University of Zurich[UZH]

Thesis December 1, 2020

Driving Argument Mining with the Help of the Crowd

Crowdsourcing Argumentative Annotations in Scientific Papers

Joachim Baumann
of Winterthur ZH, Switzerland

Student-ID: 15-715-220
joachim.baumann@uzh.ch

Main Advisor: Florian Ruosch
Second Advisor: Cristina Sarasua

Prof. Abraham Bernstein, PhD
Institut für Informatik
Universität Zürich
http://www.ifi.uzh.ch/ddis

Abstract

Huge volumes of human knowledge are available in many different data sources, many of which contain thoughtful and well-reasoned arguments in the form of natural language text. Along with recent advances in machine learning (ML) techniques, researchers have increasingly started to investigate possibilities to automatically extract argumentative components and the relations between them — a process which is called *Argument Mining* (AM). As an emerging research area, one of the major challenges in AM research is the lack of annotated datasets. These datasets are needed as training data as well as for benchmark experiments. With a focus on scientific publications, we implemented a system that could be used to create gold standard datasets for AM with the help of hundreds of thousands of ordinary workers (i.e. the crowd). Our system includes two types of tasks, one for the annotation of argument components and one for the annotation of argumentative relations that hold between those components. To evaluate and improve our system, we conducted experiments for both of these task types with 70 participants on the crowdsourcing platform Amazon Mechanical Turk. Detecting argumentative components and relations is a very complex task, especially for untrained, non-expert crowdworkers. We found that by introducing a quality assurance filter mechanism, it is possible to detect high-performing workers and also to detain workers who are expected to perform poorly from participating. In this way, it is possible, to some extent, to steer the quality of the crowd-annotated dataset, in exchange for money and time — money, because workers need to complete the task that will determine whether they will be filtered out or not, and time, because filtering out workers results in a smaller workforce, meaning it could take longer for all annotation tasks to be completed by the crowd. Our work denotes another step towards an effective interaction between researchers and the crowd in the field of AM and, thereby, decisively contributes to an emerging research area.

Zusammenfassung

Riesige Mengen menschlichen Wissens sind in vielen verschiedenen Datenquellen verfügbar. Die meisten davon enthalten durchdachte und gut begründete natürliche Sprachargumente. Die Fortschritte der letzten Jahre im Bereich von Machine Learning (ML) haben dazu geführt, dass Forscher zunehmend damit begonnen haben, Möglichkeiten der automatischen Extraktion argumentativer Komponenten sowie deren Beziehungen untereinander zu analysieren — diesen Prozess nennt man *Argument Mining* (AM). Der Mangel an annotierten Datensätzen ist eines der grössten Probleme der noch jungen AM Forschung. Solche Datensätze sind jedoch notwendig, einerseits zum Trainieren der ML Algorithmen und andererseits zur Durchführung von Benchmark-Experimenten. Wir haben ein System implementiert, welches zur Kreierung von Goldstandard-Datensätzen für AM verwendet werden könnte. Dabei haben wir uns insbesondere auf die Annotation von wissenschaftlichen Publikationen fokussiert. Um dieses Ziel zu erreichen, wurden hunderttausende gewöhnliche Arbeiter, auch Crowd genannt, in unser System eingebunden. Unser System beinhaltet zwei Arten von Aufgaben; eine zur Annotation von Argumentkomponenten und eine zur Annotation der argumentativen Beziehungen, welche zwischen diesen Komponenten bestehen. Zur Bewertung und Verbesserung unseres Systems, haben wir Experimente mit insgesamt 70 Crowdworkern auf der Crowdworking-Plattform Amazon Mechanical Turk durchgeführt. Argumentkomponenten und deren Beziehungen zu identifizieren ist ein äusserst Schwieriges Unterfangen, insbesondere für ungeschulte, nicht fachkundige Arbeiter in der Crowd. Im Rahmen dieser Experimente haben wir herausgefunden, dass es möglich ist, mit einem Qualitätssicherungs-Mechanismus, in Form eines Filters, besonders leistungsfähige Crowdarbeiter zu identifizieren und gleichzeitig Crowdarbeiter, von welchen erwartet wird, dass sie schlechte Leistungen erbringen, von der Teilnahme auszuschliessen. Auf diese Weise ist es bis zu einem gewissen Grad möglich die Qualität der von der Crowd annotierten Datensätze zu steuern, im Tausch gegen Geld und Zeit — Geld, da die Crowdworker dafür bezahlt werden müssen, dass sie den Filter durchlaufen und Zeit, da das Herausfiltern von Crowdworkern eine Dezimierung der Arbeitskräfte zur Folge hat, weshalb es länger dauern könnte, bis alle Annotationsaufgaben von der Crowd erledigt sind. Die vorliegende Arbeit ist ein weiterer Schritt zur erfolgreichen Interaktion zwischen Forschern und der Crowd im Bereich von Argument-Annotationen und trägt dadurch massgeblich zu dieser aufstrebenden Forschungsrichtung bei.

Contents

1

Introduction

The digital age has come along with a huge volume of human knowledge, which is included in many different data sources. Oftentimes, they contain thoughtful and well-reasoned arguments. However, we cannot always access and reuse the argumentative content. And even if we find out *what* is being expressed, for example, with techniques like opinion mining and sentiment analysis, we do not necessarily know *why* this particular opinion is being expressed. Pursuing the objective to create stand-alone tools, which automatically identify argumentative components and the relations between them from generic textual corpora, has led to a new research area called *Argument Mining*[1] (AM) (Bench-Capon and Dunne, 2007). On the one hand, these tools help to deal with the large-scale discovery, extraction, and reuse of argumentative data (Wells, 2014), and on the other hand, they identify argumentative structures, which makes it possible to determine not only *what* positions people are adopting, but also *why* they hold the opinions they do (Lawrence and Reed, 2019). In recent years, due to advances in Machine Learning (ML) methods, AM has become a very important part of Artificial Intelligence (AI) research (Lippi and Torroni, 2016b; Cabrio and Villata, 2018).

AM techniques have already been used for data from many different sources, such as scientific articles (Teufel et al., 2009; Lauscher et al., 2018b), legal documents (Mochales and Moens, 2011), Wikipedia articles (Lippi and Torroni, 2016c), user-generated Web content (Habernal and Gurevych, 2017), online product reviews (García-Villalba and Saint-Dizier, 2012), newspaper articles (Lippi and Torroni, 2016c), social media (Dusmanu et al., 2017) and even political debates and speeches (Lippi and Torroni, 2016a; Duthie et al., 2016a). As AM has been applied to different scenarios, there have also been different approaches regarding the technical implementation. Various different methods, such as Support Vector Machines (SVM), Parsing algorithms, Logistic Regression, or Recurrent Neural Networks, to name but a few, have been used (Cabrio and Villata, 2018). Overall, SVM have proved to be the most promising algorithms in different settings and also for different AM subtasks (Cabrio and Villata, 2018; Duthie et al., 2016a). The performance of these ML algorithms for AM heavily depends on the quality of data they are being trained with (Pustejovsky and Stubbs, 2012; Lippi and Torroni, 2016b).

As an emerging research area, one of the major challenges in AM research is the lack of data (Peldszus and Stede, 2013; Lawrence and Reed, 2019). More precisely, domain-specific annotated corpora, which is crucial for designing, training, and evaluating the

[1] Sometimes also referred to as *Argumentation Mining*.

algorithms is not sufficiently available (Habernal and Gurevych, 2017). What state-of-the-art approaches have in common is that they require large amounts of gold standard training data. Therefore, successful AM requires the creation of annotated datasets which are of high quality. These corpora will need to be created for different types of sources and also for different domains, because whether some content can function as an argument or not depends on the given context (Moens, 2018). For example, an AM algorithm that was trained with data from social media posts does not necessarily perform well when being used to identify arguments from scientific papers in a particular domain (Lippi and Torroni, 2016b). However, gold standard datasets are not only needed to train ML algorithms, but also to assess their performance and, hence, to be able to compare different solutions, for example, in benchmark experiments. These datasets have to be created separately for each of the above-mentioned application scenarios.

Ever since the development of the institutionalised structures of modern science in the 17th century — with the publication of peer-reviewed results of scientific work in journals and manuscripts —, the number of scientific publications has been increasing (Bornmann and Mutz, 2015). We do not expect this increase in scientific output to slow down in the near future. Consequently, it becomes more and more difficult to trawl through all of the relevant scientific articles for a specific topic. For this reason, the mining of arguments within scientific papers is gaining more and more attention in the field of AM (Stab et al., 2014; Green et al., 2014b; Green, 2015; Kirschner et al., 2015; Lauscher et al., 2018b,a; Accuosto and Saggion, 2019; Song et al., 2019). This makes the evaluation of novel approaches for the effective creation of gold standard datasets for scientific papers particularly interesting. Even though such an annotated dataset was released in a machine-readable format by Lauscher et al. (2018b), we still lack a system to generate further ground truth datasets. By utilising hundreds of thousands of ordinary workers (i.e. the crowd), crowdsourcing has proved to be an effective way to address tasks that can benefit from the use of human cognitive ability, like, for example, entity resolution, sentiment analysis or image recognition (Li et al., 2016). However, while some researchers (Ghosh et al., 2014; Nguyen et al., 2017; Stab et al., 2018; Miller et al., 2019; Lavee et al., 2019) have included crowdsourcing in their corpus creation workflow, it still remains unclear whether or not crowdsourcing is a viable approach to identify arguments in difficult-to-understand textual corpora, such as scientific publications, from scratch. In this regard, our work takes an exploratory approach and strives to shed light on the potential of the crowd to create gold standard datasets for AM. Accordingly, we raise the following research question:

(RQ) How can we design a crowd-powered system to annotate scientific publications for argument mining effectively?

We further break this RQ down into the following subtopics: design of tasks, data quality assurance method, workflow definition, structured annotations and aggregation method.

In this work, we introduce an end-to-end process to create gold standard datasets for AM, which, with the help of the crowd, enables effective and efficient human-computer

interaction. Further, we present a crowd-powered system which follows this process to annotate argumentative components and relations in natural language text.

While we focus mainly on the annotation of scientific publications — in the domain of computer graphics — our tool can easily be customised to annotate other types of text corpora from other domains. In addition to that, our tool can also easily be extended to annotate other types of components (or other types of relations between the preferred types of components), for example, research hypotheses. The versatility of our solution arises thanks to the high modularity of the implementation.

We evaluate the potential of our crowd-powered system by running two pilots (with a total of 24 workers) and four experiments (with a total of 46 workers) on Amazon Mechanical Turk (AMT or MTurk). Our empirical analysis shows that the introduction of a quality assurance filtering mechanism at the beginning of the crowdsourced AM annotation workflow, that checks with a few specific questions of varying difficulty whether the annotators are able to accurately annotate the different argumentative labels, has a positive effect on the AM annotation. We present such a mechanism so that a user of our system can, to some extent, steer the quality of the crowdsourced annotations. However, higher quality does not come for free, but this trade-off between quality and costs for crowd-powered AM annotations might differ even within the domain of scientific publications.

The remainder of this work is structured as follows: in Chapter 2, we introduce the concept of argumentation, crowdsourcing and AM and discuss the combination of the latter two. In Chapter 3, we describe the design of the experiment and the questions we want to answer. Then, we present the implementation of our tool in Chapter 4. We report our results in Chapter 5, before we evaluate and discuss them in Chapter 6. Subsequently, we acknowledge limitations and outline open questions that should be addressed in future work in Chapter 7. Finally, we draw conclusions in Chapter 8.

2

Background and Related Work

In this chapter, we introduce the theoretical background and related work. First, we start with some general information about argumentation and crowdsourcing. Then, we introduce AM, where we focus on existing corpora, AM in scientific publications and different performance measures. After introducing crowdsourcing and AM, we discuss related work where researchers have brought those two topics together.

2.1 Argumentation

The first studies on argumentation date back to the 4th century BCE where the ancient Greeks, in particular Aristotle, set an important starting point for further studies (Stab and Gurevych, 2017a). The discipline of argumentation has been examined from different perspectives, such as philosophy, psychology, communication studies, logic, computer science and many others (Habernal and Gurevych, 2017).

There exist different definitions of argumentation. According to MacEwan (1898, p. 1),

> "[...] argumentation is the process of proving or disproving a proposition. Its purpose is to induce a new belief, to establish truth or combat error in the mind of another."

Ketcham (1914, pp. 3–5) defines argumentation as

> "[...] the art of persuading others to think or act in a definite way. It includes all writing and speaking which is persuasive in form. [...] The object of argumentation is not only to induce others to accept our opinions and beliefs in regard to any disputed matter, but to induce them to act in accordance with our opinions and beliefs."

Similarly, Baker and Huntington (1925, pp. 6–7) write that

> "[...] argumentation is the art of producing in the mind of another person acceptance of ideas held true by a writer or speaker, and of inducing the other person to make a decision, or, if necessary, to perform an act in consequence of his acquired belief."

So, definitions of argumentation vary regarding the communicative ends and communicative means specified. However, they all conclude that the ultimate purpose of argumentation is to persuade others (O'Keefe, 2012; Habernal and Gurevych, 2017).

Persuasion and argumentation are related, but the relationship depends on how persuasion and argumentation are defined. However, a correct definition exists for neither of those two terms and when it comes to the formal description of arguments and argumentation, not even argumentation theorists agree (O'Keefe, 2012). In this regard, van Eemeren et al. (2014, p. 29) stated that there is no universally accepted theory of argumentation and that "[...] the current state of the art in argumentation theory is characterized by the coexistence of a variety of theoretical perspectives and approaches, which differ considerably from each other in conceptualization, scope and theoretical refinement".

There needs to be some means by which the desired effects of argumentation is being communicated, such as some kind of argumentative writing or discourse (Walton, 1998; O'Keefe, 2012; Habernal and Gurevych, 2017). For this goal to be achieved, an argument can be the means to convince someone whether a particular claim is true or not, which then, in turn, can convince the opposite to believe in that claim (Damer, 2012). The term *argument* here does not refer to a dispute or disagreement and, at the same time, is not just an opinion. An argument is a supported claim whereas an opinion is an unsupported claim. According to Damer (2012, p. 13), "An argument is a group of statements, one or more of which, the premises, support or provide evidence for another, the conclusion". This group of statements, such as premise or claim, are often referred to as *argument components* (Habernal and Gurevych, 2017). The argument components are those statements that give a reason why the conclusion should be true. In addition to argument components, relationships that exist between those components, such as support or attack relations, are called *argumentative relations* (Stab and Gurevych, 2014). Together, argument components and argumentative relations form the argumentation structure (Peldszus and Stede, 2013).

Argumentation theories tend to agree that an argument can be decomposed into various interrelated components (Miller et al., 2019). As the study of argumentation is a comprehensive and interdisciplinary research field, there have been many proposals for modelling argumentation and also for classifying these components. Many of the different theoretical frameworks of argumentation that have been proposed are extensively outlined by Bentahar et al. (2010), along with its respective advantages and limits. These so-called argumentation schemes, which intend to be easily understandable, should allow a user to analyse and recognise arguments (Walton et al., 2008). However, the most widely used — also in the context of AM (Kirschner et al., 2015) — is the model of Toulmin (2003). As can be seen in Figure 2.1, Toulmin (2003, pp. 90–99) replaces the old concepts of "premise" and "conclusion" in his model with new, more specific concepts of "claim", "data", "warrant", "backing", "qualifier" and "rebuttal". The lines and arrows stand for implicit relations between the components. According to Toulmin (2003, pp. 90–99), ...

... a **claim** is an assertion or a conclusion which is presented.

... **data** are the facts which are used as foundation for the claim.

... **warrants** are general, hypothetical statements, which can act as inference from the claim to the data.

... **backing** is a set of information that assures trustworthiness of a warrant. Backings are needed when a warrant is challenged.

... a **qualifier** shapes the degree of certainty regarding the claim stated in virtue of the warrant.

... a **rebuttal** describes a situation in which the claim might be defeated.

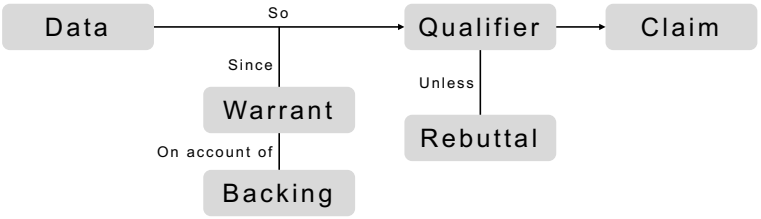


Figure 2.1: Model of an argument, as specified by Toulmin (2003, p. 97)

Other complex argumentation schemes, such as that of Walton (1998), have been presented over the years but are less relevant in the scope of this work as early on we decided to use a simplified version of Toulmin's model. A particular example of an argument based on the model of Toulmin (2003) is visualised in Figure 2.2. The simplified version of Toulmin's 2003 model is described in detail in Section 3.2.2.

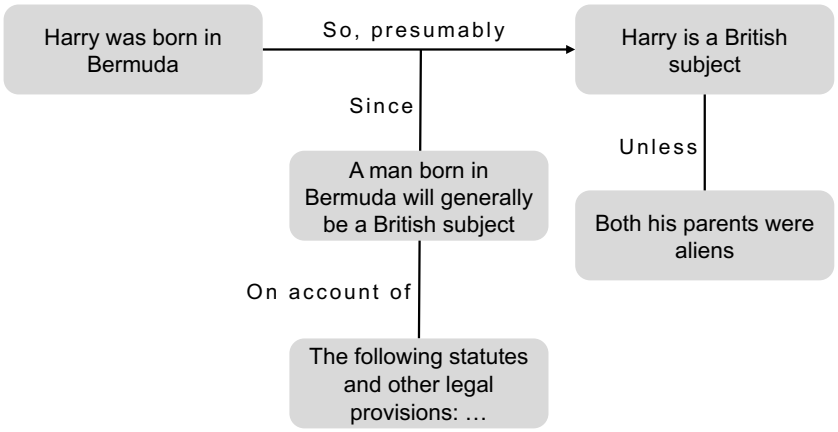


Figure 2.2: Example of an argument based on the Toulmin model, as specified by Toulmin (2003, p. 97)

2.2 Crowdsourcing

Crowdsourcing is a subfield of collective intelligence, which has been defined as the idea that, together, many people together can achieve great solutions disregarding of the intelligence of the individuals (Quinn and Bederson, 2011). Howe (2006) coined the term crowdsourcing in the year 2006. On his website he defines it as "[...] the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call" (Howe, 2009). The terms crowdsourcing and human computation overlap in the sense there are applications that could be a replacement for a human role as well as for computer roles (Law, 2011). Social computation and crowdsourcing are not completely separated, as they both rely on a group of humans, mediated by technology, to solve a task.

In recent years, crowdsourcing has become known to be a good solution to break down a manual, time-consuming task into more manageable tasks which are completed by different workers who are distributed over the Internet. For this reason, numerous crowdsourcing platforms exist where everyone has the possibility to set up a task to be completed by the crowd, the most widely used being AMT[1] or Appen[2] (formerly Figure Eight and Crowdflower). The persons who set up these so-called microtasks (also known as Human Intelligence Tasks or HITs) are called requesters. The ones actually solving the tasks are referred to as crowdworkers (sometimes just called workers or, in the case of AMT, turkers). A study by Difallah et al. (2018) in 2018 revealed that, in the case of AMT, there are more than 100,000 workers available, that at any given time there are more than 2,000 active workers, and that most of the workers are from the USA (75%) or from India (16%).[3]

2.3 Argument Mining

Looking at argumentation in natural language from a computational linguistics perspective has led to a new field called *Argument Mining* (Bench-Capon and Dunne, 2007; Green et al., 2014a). One of the very first to use the approach which is now called AM was Teufel et al. (2009). Then, Mochales and Moens (2011) followed with different methods to detect arguments from legal texts. Ever since these early approaches, AM has become one of the most promising research areas in AI and has seen a growing community with more and more funded projects (Cabrio and Villata, 2018). AM has also seen a rapid growth in conferences: ACL workshops on the topic are held annually after the first took place 2014[4], with this year's workshop[5] being the seventh edition.

[1] *https://www.mturk.com.*

[2] *https://appen.com.*

[3] The demographics survey of Difallah et al. (2018) is ongoing and their results are available at *http://demographics.mturk-tracker.com.*

[4] *http://www.uncg.edu/cmp/ArgMining2014.*

[5] *https://argmining2020.i3s.unice.fr.*

Today, a clear definition of AM is still missing (Wells, 2014). However, researchers within this field usually focus on "[...] analyzing discourse on the pragmatics level and applying a certain argumentation theory to model and analyze textual data at hand" (Habernal and Gurevych, 2017). In order to successfully mine arguments, an interdisciplinary approach is needed; natural language processing provides the basis for identifying argumentative structures, knowledge representation and reasoning can then help to reason about the retrieved arguments, and Human-Computer Interaction provides the design of good supportive tools (Cabrio and Villata, 2018). In many contexts, such as in user-generated content on the web and also in scientific writing, much more data is available now than was available just a few years ago (Bornmann and Mutz, 2015). Due to this information overload, it can be quite difficult to make sense of it, which is why automatic processing of such data is becoming more and more important. In the case of web data, AM can provide arguments to a given topic of interest and also find evidence for the given controversial topic. Another possibility is that AM helps to reveal argumentation flaws in such data (Habernal and Gurevych, 2017). Regarding the information seeking perspective in scientific papers, AM can satisfy the goal of automatic analysis of a given article by, for example, identifying the article's contribution in relation to background material (Teufel and Moens, 2002). Even though opinion mining and sentiment analysis have been very successful in many tasks related to natural language processing, these existing techniques fail to identify more complex structural relationships between concepts (Lawrence and Reed, 2019). AM strives to solve this difficult problem by turning unstructured text into structured argument data to understand why people have a certain opinion. To do that, reasons have to be searched as opposed to just opinions and sentiments (Lippi and Torroni, 2016b).

Any AM system has to deal with different strictly interrelated tasks. Stab and Gurevych (2017a) provided a categorisation by splitting AM into the following three subtasks:

- **Component identification**[6] focuses on the identification of argument components and their boundaries in text units. Hence, in this task, the annotator (human or machine) has to decide for each component[7] whether it is argumentative or not.
- **Component classification**[8] addresses the identification of the type of argument components (i.e. to classify them into different types, such as claims and premises, among others).
- **Structure identification**[9] involves predicting the relations that hold between the argument components identified in the previous subtasks. Different types of such argumentative relations can, for example, be support or attack relations.

[6] Also referred to as segmentation or boundaries detection.

[7] Also referred to as segment or Argumentative Discourse Units.

[8] Also referred to as sentence classification. Conducting component identification and classification together is also referred to as component extraction or component detection.

[9] Also referred to as relation identification, relation extraction or relation prediction.

There is no standard that specifies the boundaries for component identification (Stede and Schneider, 2018). Such boundaries could be an entire sentence, a part of a sentence, two parts of a sentence combined, etc. The types into which argument components are classified during the component classification task depend on the argumentation scheme used. If the model created by Toulmin (2003) is used, these types would be claim, data, warrant, backing, qualifier and rebuttal. As component identification and classification are the first steps of AM, most of the research efforts concentrate on these two steps.

Lippi and Torroni (2016b) and Cabrio and Villata (2018) provide an extensive overview of the different methods that have been applied to all three AM subtasks in the past. Cabrio and Villata (2018) found that overall SVM perform best for all subtasks. However, Lippi and Torroni (2016b) emphasise that there is no one best algorithm for all use cases and that to positively impact the performance, one should rather put effort into conceiving good features. In general, we can see from existing AM systems that for component classification, even though there have been other approaches, SVM are by far the most used method. For component identification and structure identification, there does not seem to be a method which is favoured by researchers.

No only for automated systems, but also for human annotators, mining argument in natural language text i is a very difficult task. Identifying and understanding the argumentative components and relations in natural language text is challenging (Moens, 2018). Stab et al. (2014) reported that the ambiguity of argumentation structures is the biggest challenge due to the fact that there are often several possible interpretations. This can make it impossible to identify one correct interpretation. In addition to that, a lot of the argumentative content is not expressed explicitly but has to be read between the lines (Peldszus and Stede, 2013). Also, whether some content can function as an argument or not heavily depends on the context (Moens, 2018). Thus, AM faces subtasks which are complicated (and controversial) even for humans (Mochales and Ieven, 2009; Habernal et al., 2014). The structure identification subtask is particularly difficult due to high-level knowledge representation and reasoning issues (Cabrio and Villata, 2018; Lippi and Torroni, 2016b). Hence, when mining arguments in natural language text, even expert human annotators might disagree in some cases, which makes it an incredibly difficult task for machines, despite the recent advances in ML methods.

Stede and Schneider (2018) proposed some ideas on what can be done with arguments once they have been mined. They conclude that visualisation, summarisation and evaluation are the most important techniques to be applied to mined arguments. Visualising arguments that have been gathered and analysed can help to improve qualitative error analysis for the mining techniques and also to provide input to human analysts. For the creation of good summaries, the identified arguments can be clustered, and the different aspects raised by the variety of arguments would then need to be identified.

There are many different possible applications scenarios for AM (Stede and Schneider, 2018). Group decision-making often depends on finding and comparing different claims. Schneider (2014) suggested that AM can be used to foster open source and open knowledge projects, especially for sense-making and decision-making, as an individual or as

a group. Here, sense-making refers to the process of making sense of large or differing information to be able to explain it. Further, AM could be used for systems to support practical reasoning for policy-making or public discourse on topics of public concern, as explored by Kriplean et al. (2012) and Walton (2015). Another promising application of AM is the active support of argument exchange on the web. In this regard, human debaters could be actively supported by suggesting suitable arguments to them (Rosenfeld and Kraus, 2016). On the one hand, AM has already proven to help rank essays, while scoring them appears to be more difficult (Ong et al., 2014). On the other hand, AM might also be helpful in providing qualitative feedback to students on their writing, for example, by checking if arguments are sufficiently supported (Stab and Gurevych, 2017b). One has to keep in mind that here we have briefly introduced just some of the promising application scenarios of AM. There are many more, and due to extensive research in recent years, AM will most likely find its way into new areas in the future.

2.3.1 Corpus Creation for Argument Mining

Natural language datasets are referred to as corpora, and a single set of data annotated based on some specification is called an annotated corpus. Any attempt at AM which uses some kind of ML and AI techniques requires a corpus to be used as data to train a predictor (Lippi and Torroni, 2016b). For this reason, datasets annotated with argumentative components and relations are a highly sought after resource because state-of-the-art approaches are based on ML techniques and therefore do require large amounts of gold standard training data. These approaches often make use of (un)supervised ML techniques and thus their success heavily depends on the quality of the data they are being trained with (Pustejovsky and Stubbs, 2012; Lippi and Torroni, 2016b). Accordingly, we not only require many different ground truth datasets, but we also need them to be of high quality. In order to create them, researchers typically hire a group of expert annotators who are asked to annotate the text at hand (Lippi and Torroni, 2016b). Even in a setting with easy specifications, annotating datasets is an expensive and time-consuming task, so the difficulties AM entails do not help to keep the costs low (Pustejovsky and Stubbs, 2012).

Experts need a tool to be able to annotate argument components and argumentative relations in natural language. Over time, various tools have been used to facilitate the annotation process and the visualisation of annotations. Some tools are available as WebApps (such as OVA2[10] or BRAT[11]) and others are stand-alone solutions which need to be downloaded (for example, NOMAD[12] or AVIZE[13]). Figure 2.3 shows screenshots

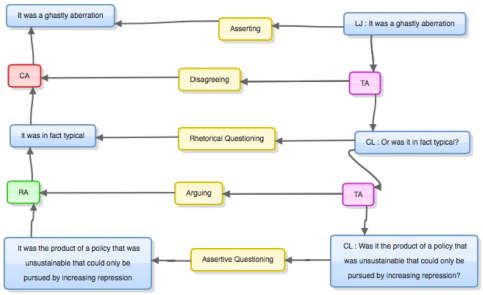
[10]Initially published by Janier et al. (2014), the second version (OVA2), which was developed at ARG-tech, is available at: *http://ova.arg-tech.org*.

[11]BRAT was developed by Stenetorp et al. (2012) and is available (online or for download) at: *http://brat.nlplab.org*.

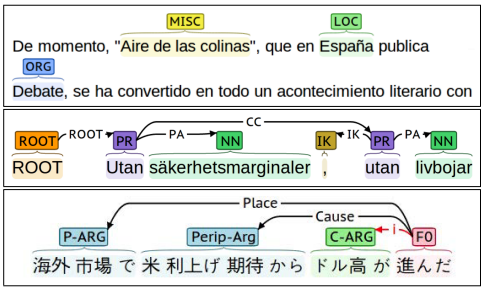
[12]NOMAD was developed by Petasis (2014) and is available for download at: *https://www.ellogon.org/index.php/annotation-tool/nomad-annotation-tool*.

[13]AVIZE was developed by Green et al. (2019) and is available for download at: *https://github.com/greennl/AVIZE*.

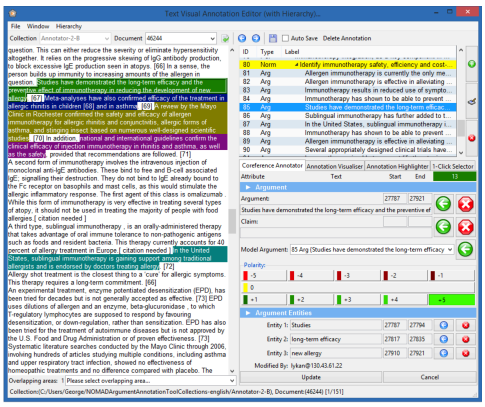
of the mentioned tools. As can be seen, these tools' user interfaces are often designed to visualise the argumentative structure and the offered functionalities are not always intuitive. These tools are optimised for expert annotators, and some of them were even designed for use in a group setting. NOMAD, for example, allows for communication amongst annotators during the annotation process. For these reasons, many annotation tools are highly customisable. Therefore, inexperienced annotators might be discouraged due to the information overload.



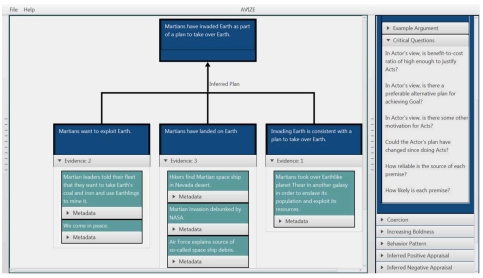
(a) The OVA2 annotation tool, as specified by Janier et al. (2014, p. 464)



(b) The BRAT annotation tool, as specified by Stenetorp et al. (2012, p. 102)



(c) The NOMAD annotation tool, as specified by Petasis (2014, p. 1934)



(d) The AVIZE annotation tool, as specified by Green et al. (2019, p. 51)

Figure 2.3: Screenshots of a selection of currently existing annotation tools

In Table 2.1 we provide a short overview of recently released datasets. Apart from the domain of the dataset, we also list the achieved IAA (which we will further describe in Section 2.3.3) and the subtask(s) for which each dataset was created. For older contributions, we refer to Lippi and Torroni (2016b), Cabrio and Villata (2018), and Lawrence and Reed (2019).

While much work has been done in other domains — such as legal documents, user-generated web content, newspaper articles, social media and politics — annotated datasets for tasks related to AM on scientific publications are scarce. Currently, there are only three publicly available corpora of scientific papers that have been argument-annotated,

namely those released by Kirschner et al. (2015), Lauscher et al. (2018b) and Accuosto and Saggion (2019).

Consistency in the way the annotators proceed when annotating the text with the different labels is very important (Pustejovsky and Stubbs, 2012). For that to be the case, detailed and easy-to-understand guidelines are needed, which annotators will then follow when annotating the text. There is currently no consensus amongst AM researchers on what such guidelines should look like. Different researchers typically work out their own guidelines and some of them, for example, Stab and Gurevych (2014)[14], Kirschner et al. (2015)[15], Teruel et al. (2018a)[16], and Lauscher et al. (2018c)[17], have also made them available online. The fact that not everyone uses the same argumentation scheme complicates the goal of having consistent guidelines, as they heavily depend on the underlying argumentation scheme. The guidelines influence the result of the annotation to a great extent as they often also contain specific instructions, for example, whether punctuation at the beginning or at the end of an argumentative statement should also be annotated or not. Therefore, guidelines are usually refined iteratively. Teruel et al. (2018b) suggest that annotators discuss conflictive examples to be able to create guidelines that are as clear as possible with the final goal of a lower level of disagreement between annotators.

[14]Annotation guidelines available at: *https://www.informatik.tu-darmstadt.de/media/ukp/data/fileupload_2/argument_annotated_news_articles/ArgumentAnnotatedEssays-1.0.zip*.

[15]Annotation guidelines available at: *https://www.informatik.tu-darmstadt.de/media/ukp/data/fileupload_2/argument_annotated_news_articles/guidelinesLinkingThoughts_English.pdf*.

[16]Annotation guidelines available at: *https://github.com/PLN-FaMAF/ArgumentMiningECHR/blob/master/docs/guidelines-annotating-argumentation(1).pdf*.

[17]Annotation guidelines available at: *http://data.dws.informatik.uni-mannheim.de/sci-arg/annotation_guidelines.pdf*.

Dataset	Domain/Source	URL	IAA	CD		SI
				CI	CC	
Accuosto and Saggion (2019)	Scientific Papers	*http://scientmin.taln.upf.edu/argmin/sciabb_argmin_annotations.tgz*	F1 = 0.38 - 0.69	✓	✓	✓
Skeppstedt et al. (2018)	Microtexts	*http://angcl.ling.uni-potsdam.de/resources/argmicro.html*	F1 (macro) = 0.64 - 0.78		✓	✓
Teruel et al. (2018b)	Legal	*https://github.com/PLN-FaMAF/ArgumentMiningECHR*	κ (CI sentence level) = 0.77 - 0.84, κ (CI token level) = 0.59 - 0.84, κ (CC) = 0.48 - 0.56, Accuracy (SI) = 0.10 - 0.19	✓	✓	✓
Lauscher et al. (2018b)	Scientific Papers	*http://data.dws.informatik.uni-mannheim.de/sci-arg/compiled_corpus.zip* Guidelines: *http://data.dws.informatik.uni-mannheim.de/sci-arg/annotation_guidelines.pdf*	F1 (CD) $\approx$ 0.20 - 0.60, F1 (SI) $\approx$ 0.03 - 0.47	✓	✓	✓
Park and Cardie (2018)	User-generated Web Content	*https://facultystaff.richmond.edu/~jpark/data/cdcp_acl17.zip*	α (CC) = 0.65, α (SI) = 0.44		✓	✓
Stab et al. (2018)	Web Content (Controversial Topics)	*https://www.informatik.tu-darmstadt.de/ukp/research_6/data/argumentation_mining_1/ukp_sentential_argument_mining_corpus/index.en.jsp*	Many different performances depending on setting and method		✓	✓
Rocha and Cardoso (2017)	News Article (Opinion Articles)	*http://corpora.aifdb.org/ArgMine*	F1 (CI) = 0.48 - 0.89, F1 (SI) = 0.58 - 0.81	✓	✓	✓
Walker et al. (2017)	Legal Judgements	*https://github.com/LLTLab/VetClaims*	Not reported		✓	
Stab and Gurevych (2017b)	Student Essays	*https://www.ukp.tu-darmstadt.de/data/argumentation-mining/insufficiently-supported-arguments/*	Many different performances depending on setting and method.		✓	
Kirschner et al. (2015)	Scientific Papers	available on request	F1 Score (weighted average) = 0.5559			✓

Table 2.1: Recently released datasets for AM (sub)tasks. The acronyms stand for: Inter Annotator Agreement (IAA), Component Detection (CD), Structure Identification (SI), Component Identification (CI), Component Classification (CC)

2.3.2 Mining Arguments in Scientific Publications

Ever since the development of the institutionalised structures of modern science in the 17th century — with the publication of peer-reviewed scientific results — the number of scientific publications has been increasing (Bornmann and Mutz, 2015). Scientific papers aim at publishing evolving knowledge in different disciplines, and for this endeavour, arguments are the means through which results are communicated (Stab et al., 2014). Therefore, mining arguments in scientific papers can foster the representation and discovery of knowledge. Likewise, researchers assume that identifying argument components in scientific publications helps to create a summary of the text (Stab et al., 2014). Recent work by Accuosto and Saggion (2019), identified that argumentative components and relations in scientific texts have even been used to predict the acceptance of papers at computer science conferences.

As a forerunner of AM, Teufel and her colleagues performed a rhetorical-level analysis of scientific articles and introduced the definition of *argumentative zoning* (AZ) (Teufel and Moens, 1999; Teufel et al., 1999, 2009). The AZ approach aims at the detection of important sentences based on their likelihood of being argumentative, with the final goal of automatic text summarisation. However, AZ looks at the rhetorical level of articles and does not includes fine-grained argument components on a token level or relations between these argument components. A few years later, Blake (2010) introduced a framework that differentiates between implicit and explicit claims, observations, correlations and comparisons to find out how authors communicate findings in empirical studies in the biomedical domain. In contrast to Blake (2010), several other researchers (Stab et al., 2014; Green et al., 2014b; Green, 2014, 2015, 2016; Kirschner et al., 2015; Lauscher et al., 2018b,a; Accuosto and Saggion, 2019; Song et al., 2019) were then interested in identifying claims and their relations in a text instead of finding out how they are constructed. However, the only ones to release a publicly available annotated corpus of scientific papers are:

- Kirschner et al. (2015): 24 scientific papers with a total of 2,743 sentences from the domain of educational psychology and developmental psychology, in German.
- Lauscher et al. (2018b): 40 scientific papers with a total of 10,789 sentences from the domain of computer graphics, in English.
- Accuosto and Saggion (2019): 60 abstracts with a total of 327 sentences from the domain of computational linguistics, in English.

All three datasets were created using a different argumentation scheme and were published in different formats. The only dataset of full scientific papers in English, which were argument annotated in a machine-readable format, is the one released by Lauscher et al. (2018b).[18] The reason for this is that it is quite difficult and also costly to create such a dataset as we still lack a general system to generate further ground truth datasets without the need to hire expert annotators. In the future, further ground truth

[18]Details on the format applied by Lauscher et al. (2018b) can be found in Section 3.2.1.

datasets will be crucial for advancing AM in scientific publications. On the one hand, they are needed to train ML algorithms, and on the other, they would enable benchmark experiments.

2.3.3 Performance Measures

The so-called Inter Annotator Agreement (IAA) scores are used to compare individual annotators (Artstein, 2017). Thus, the IAA can be used to assess how clear the annotation instructions are (Pustejovsky and Stubbs, 2012). Accordingly, a high IAA score indicates that the annotation instructions are clearly defined and vice versa. This is necessary but not sufficient to obtain correct annotations (Artstein, 2017). However, calculating the IAA is not trivial because experts do not agree on the way it is calculated, on how the score is interpreted and even on how reliable the different measures are (Wacholder et al., 2014). Nevertheless, various metrics have been proposed to solve the problem of evaluating the performance of AM, some of which we briefly introduce here:

- **F1 Score**: The F1 Score is also referred to as F-Measure, F Score or just F1. The F1 Score is an accuracy measure which compares predicted labels against the true labels (also referred to as the gold standard). As defined by Van Rijsbergen (1979), it is the harmonic mean of precision (P) and recall (R). The precision measures how many labels were accurately identified — or, in other words, how many relevant labels were retrieved in comparison to all of the retrieved labels. It is defined as the number of true positives (TP) divided by the total number of positives in the prediction, which is the sum of TP and false positives (FP):

$$P = \frac{TP}{TP + FP} \tag{2.1}$$

  The recall measures how many relevant labels were identified — or, in other words, how many relevant labels were retrieved in comparison to all of the relevant labels. It is defined as the number of TP divided by the total number of positives in the gold standard data, which is the sum of TP and false negatives (FN):

$$R = \frac{TP}{TP + FN} \tag{2.2}$$

  Finally, the output of the F1 Score computation is a number between 0 and 1, which is used as a metric to evaluate the performance of a prediction (e.g. of an algorithm) — with 1 being perfect accuracy in comparison to the gold standard data:

$$F1\,Score = \frac{2 * P * R}{P + R} \tag{2.3}$$

  Depending on the context, an advantage of the F1 Score can be that the number of negatively identified labels does not need to be known. However, the F1 Score does not tell us how exactly this value was reached and whether precision is lower than

recall or not. Further, as reported by Powers (2015), the fact that if one label type exists much more often than others in the gold standard, then an annotator who always simply guesses this category can reach a higher F1 Score than someone who actually tries to make honest annotations but also makes some mistakes. But still, the F1 Score is a widely used performance measure, mainly because it is a metric that is easy to use in terms of quickly telling how good a prediction's performance is (Pustejovsky and Stubbs, 2012). However, there has also been criticism, in particular, because of doubly penalised annotation mistakes when the F1 Score is used for measuring the performance of AM (Duthie et al., 2016b).

- **Kappa Measures**: Cohen's Kappa (Cohen's κ) measures the agreement between two annotators that are annotating the same thing (Cohen, 1960). By doing that, it takes into account the possibility of chance agreement. Cohen's κ has been defined as:

$$Cohen's\ \kappa = \frac{P_o - P_e}{1 - P_e} \tag{2.4}$$

  where P_o is the relative observed agreement amongst annotators (which is identical to the accuracy), and P_e is the hypothetical probability of chance agreement. Cohen's κ can range from 0 to 1 — with 0 meaning that there is a random agreement and 1 meaning that there is complete agreement.

  Fleiss' Kappa (Fleiss' κ) is a variation of Cohen's κ which is used for more than two annotators (Fleiss, 1971). The base equation for Fleiss' κ is essentially the same as for Cohen's κ: the actual agreement and the expected chance agreement are calculated and compared. However, the way these are calculated differs. Instead of assuming that all labels are annotated by the same annotators, as is the case in Cohen's κ, Fleiss' κ just assumes that all labels are annotated the same number of times, which is why it is not restricted to exactly two annotators.

- **Krippendorff's Alpha (α)**: Krippendorff's α was introduced by Krippendorff (2018) in 1989. Krippendorff's α is a reliability coefficient to measure the agreement of annotators. In comparison to the Kappa measures, it observes the expected disagreement:

$$\alpha = 1 - \frac{D_o}{D_e} \tag{2.5}$$

  where D_o is the observed disagreement and D_e is the disagreement one would expect if the annotation were done by chance. While Krippendorff's α seems to be a promising measure, because it can be used for any number of annotators (Antoine et al., 2014), there are also drawbacks, such as the dependency on a large sample size (Zhao et al., 2013).

- **CASS**: Duthie et al. (2016b) identified the challenges related to agreement measures in the field of AM and, as a result, proposed a technique for combining metrics that cover different parts of AM, named the Combined Argument Similarity Score (CASS). According to Duthie et al. (2016b), in the case of AM, κ measures

or the F1 Score can over penalise differences in argumentative structures. More specifically, Cohen's κ can penalise doubly or also too harshly in cases where two annotations are almost but not exactly the same. For this reason, Duthie et al. (2016b) created the CASS, which separates the calculation into three parts: segmentation similarity (S), propositional content relation (P) and dialogical content relation (D). While the CASS focuses on evaluating segmentation of a text, it does not take the labels of an annotation into consideration.

All three steps make sure that near misses (two segments that differ by a margin as small as one word) are penalised, but not as heavily as full misses (two segments that do not overlap at all). S considers a minimum edit distance, scaled to the overall segmentation size to account for that. Performing the same calculation with the F1 Score or Cohen's κ would result in a heavily penalised segmentation. M is the sum of all propositional content calculations plus the sum of all dialogical content calculations divided by the total number of calculations, n. P compares relations by calculating the agreement between each of the individual labels which are annotated within an argument analysis. Thereby, it does not penalise on this basis of different segmentation. D makes sure to not doubly penalise by checking whether each relation is contained in an argument map or not. Accordingly, the equation of the arithmetic mean M looks as follows:

$$M = \frac{\sum P + \sum D}{n} \tag{2.6}$$

Finally, the CASS measure is defined as the arithmetic mean combined with the segmentation similarity:

$$CASS = 2 * \frac{M * S}{M + S} \tag{2.7}$$

As reported by Duthie et al. (2016b), in comparison to κ measures and the F1 Score, the CASS effectively handles errors of segmentation. Even if the CASS technique for combining metrics covering different parts of the argument mining task seems promising, as of yet, it has only been applied to argument data in the Argument Interchange Format (AIF) and a publicly available general implementation does not yet exist.

- **UCP Porto Metric**: The UCP Porto metric was introduced by Sá (2019). Their intention was to create a highly flexible metric including the following three levels:

  1. **Textual matching**: Initially, the metric calculates the number amount of correct nodes (each node corresponds to an annotated argument component). In order to do that, precision, recall, and the F1 Score of the identified components are computed and the resulting F1 Score is then adjusted based on a configurable grading scale (Sá, 2019, p. 67).

2. **Usage of nodes**: In this level, the metric examines whether the nodes were used correctly by evaluating the role that each node holds in the annotated structure with regard to different possible roles a node can perform.
3. **Relation matching**: On the third and final level, the metric evaluates whether the relations connecting the nodes are correct or not. In order to do that, three different steps are performed on this level for each possible relation combination (to compare the annotated relations with the relations in the ground truth data):
   a) Relation Matching: The first step checks whether the nodes at both ends of the relation are the same.
   b) Support/Attack Check: The second step checks whether the relation type is correct, given that the nodes have been found to match in the first step. While this step has been called *Support/Attack Check* by Sá (2019), the different possible types of relations depend on the used argumentation scheme and are therefore not limited to just support or attack relations.
   c) Convergent/Linked Check: Finally, in the third step, the metric checks whether the relation is part of the same argument type with regard to the argumentation diagram, given that the compared relations have been found to match in the first two steps. These types are dependent on the argumentation scheme used and might include, for example, convergent or linked arguments.

   In order to calculate the final grade, the three steps are weighted as follows for each relation combination: $0.2 * \mathit{Relation\ Matching} + 0.4 * \mathit{Support/Attack\ Check} + 0.4 * \mathit{Convergent/Linked\ Check}$. And then the Level 3 grade can be calculated by dividing the sum of all gold standard annotation relation's grades by the total number of relations in the gold standard annotation.

As can be seen in the description of the Level 3 grade, only relations that exist in the gold standard data are considered. However, this formula is actually used to prevent annotated relations which are not present in the gold standard from being doubly penalised, as this has already been done in previous levels of the UCP Porto Metric. So for this level's grading, FP and true nagatives (TN) are filtered out, that is, only those relations that are present in the gold standard data are considered.

Finally, the UCP Porto Metric calculates the final grade of the annotation by adding up the grades of all three levels after each level's result has been weighted:

$$\begin{aligned} \mathit{UCP\ Porto\ Metric\ Final\ Grade} = {} & 0.4 * \mathit{Level\ 1\ grade} \\ & + 0.4 * \mathit{Level\ 2\ grade} \\ & + 0.2 * \mathit{Level\ 3\ grade} \end{aligned} \tag{2.8}$$

All weights used for the calculate of the final grade of the third level and equation

2.8, as well as the grading scale used on level 1, are based on suggestions by Sá (2019) but are designed to be configurable.

2.4 Annotating Arguments with the Help of the Crowd

Crowdsourcing is becoming increasingly popular as it provides a good solution to cope with difficulties related to the volume of data and also because it facilitates the completion of many annotations in a short amount of time (Dumitrache et al., 2018). When it comes to crowdsourcing text annotations, crowdworkers perform best when identifying persons or locations (Demartini et al., 2017). Identification of argumentative components and relations, however, is something that is found to be difficult even for experts (Mochales and Ieven, 2009; Habernal et al., 2014). So this task is incredibly difficult in comparison with typical crowdsourcing tasks. On top of that, embedding crowdsourcing in workflows is often very time consuming and challenging for programmers or application developers (Marcus and Parameswaran, 2013). Nevertheless, some researchers have used crowdsourcing in one way or another within their AM corpus creation workflow, some of which we introduce here.

Ghosh et al. (2014) explore the feasibility of using crowdsourcing to solve the problem of lack of argument-annotated corpora in online interactions. They conducted two AMT experiments, with a total of 10 workers, to find out if that way they can manage to obtain finer-grained annotations of basic argument components. In the first experiment, workers had to decide whether they agree with an argumentative relation or not. In the second one, workers had to annotate argument components. Based on these two experiments, Ghosh et al. (2014) demonstrated that crowdsourcing is indeed a suitable approach, especially to refine argument components which are found to be easier to annotate by experts.

Nguyen et al. (2017) used crowdsourcing as a means to mine arguments which could then be used as training data for an automatic AM algorithm. For their analysis, Nguyen et al. (2017) considered web documents, such as news entries or articles found on Bing[19], for five different topics, namely vaccine, processed food, genetically modified food, death penalty, and globalisation. They focused on designed low complexity tasks. Crowdworkers were expected to detect candidate claims and evidence for these claims on a sentence level in short paragraphs. Since the topics were defined in advance, they could ask specific questions which could be answered in Boolean questions, which helped to keep the overall complexity low, such as:

> *Does Segment S1 support the above claim?*
> *Which of the following segments expresses a claim about the keyword Globalisation?*

They found that crowdsourced argument identification does not work equally well for all topics. They concluded that crowdsourced argument extraction for topics that do

[19] *https://www.bing.com.*

not converge with the expertise and background of crowdworkers are more expensive. They reported that arguments extracted from paragraphs of topics of everyday life, such as death penalty or globalisation, are more meaningful than those obtained from other paragraphs, such as vaccine, processed food, or genetically modified food.

Stab et al. (2018) showed that crowdworkers are able to annotate arguments in arbitrary web texts quickly and reliably thanks to their rather easy-to-use, simplified, annotation scheme — they differentiate between argument component types but do not consider relations between them. They did, however, restrict themselves to topics that can be related to keywords, similar to the approach of Nguyen et al. (2017). Also, they only allowed for arguments on a sentence level. Soon after, Miller et al. (2019) continued investigating the combination of crowdsourcing and AM and reported on the achieved annotation reliability for argument-annotated product reviews. They found that the performance of the application of their initial argumentation scheme could be improved by splitting it up in three consecutive crowdsourcing steps. In the first step, crowdworkers are asked to identify the major claim; in the second step, workers annotate further claims; and in the end, workers are asked to find relations between these claims. They report that the crowd-powered solution substantially agreed with expert annotators.

According to Lavee et al. (2019), labelling all possible combinations of sentences and claims in a long debate speech does not scale with a crowdsourcing solution. Therefore, they give crowdworkers the possibility to access the full speech (text and audio). This has the advantage that they can see the full context, which makes a speech easier to understand. By hand-picking a group of high-performing annotators based on the number and quality of previous annotations, they managed to obtain high-quality annotations.

According to Nowak and Rüger (2010), crowdsourcing data annotations is quick and cheap. This opens up opportunities for large-scale annotation projects. In general, at least three annotators are needed to make sure there is always the possibility to break a tie if two annotators disagree on an argument annotation (Wacholder et al., 2014; Dumitrache et al., 2018). Crowdsourcing platforms, which are well-established entities in the research community, promise to solve this recruitment problem because as a requester, you can ask almost as many crowdworkers as you want to work on your task.

Even though some researchers have combined crowdsourcing with AM, to our knowledge, no one has done it to identify arguments in scientific publications. Hence, it remains unclear whether or not crowdsourcing is a viable approach to annotate argumentative components and relations in difficult to understand textual corpora, such as scientific publications, from scratch, with the final goal being to curate ground truth datasets.

3

Experimental Design

In this chapter, we describe the experimental design of our work. We start off by recapitulating the RQ and breaking it down into smaller subtopics. We then outline the general set-up of our study. Finally, we discuss the analysis design, including the stated hypotheses and the metrics based on which we evaluate them.

3.1 Goals

This work aims to shed light on the potential of the crowd to create gold standard datasets for AM. With a focus on scientific articles, we set out to answer the following research question:

(RQ) How can we design a crowd-powered system to annotate scientific publications for argument mining effectively?

Our main goal is to find out whether crowdsourcing is a viable approach for the generation of further ground truth datasets that could be used in empirical evaluations for AM. Therefore, we build a crowd-powered system which we can test to finally improve upon it. As this has not yet been done for scientific articles, there are a number of unanswered questions, which is why we further break this RQ down into the following five subtopics and associated questions:

(RQa) *Design of Tasks*: What types of tasks can we define to obtain accurate annotations of argumentative components and relations? In which of these tasks do crowd workers perform better?

Since AM can be split up into different subtasks (Stab and Gurevych, 2017a), there are various ways in which the problem of annotating arguments with the help of the crowd can be tackled. RQa reveals that the idea is to generate tasks for the annotation of argumentative components and relations. However, what exactly these tasks contain and how they are designed is part of our iterative approach to implementation and evaluation, which should enable us to answer this question.

(RQb) *Data Quality Assurance Method*: Does the profile of crowd annotators influence the quality of results? What task assignment mechanism and aggregation method can be defined to increase annotation performance and efficiency?

This question concerns the profile of crowdworkers, regarding the requirements a requester can define on the crowdsourcing platform. Further, the question aims at finding mechanisms to assign HITs to those workers who produce high-quality output and also at elaborating on methods to aggregate worker answers, resulting in annotations which are likely to be correct.

(RQc) *Workflow Definition*: How can we interweave machine and human computation optimally?

This question concerns the creation of an end-to-end process from the raw data via a crowd-powered solution to the final output in the form of an argument-annotated corpus.

(RQd) *Structured Annotations*: What vocabularies should be used/extended to annotate argumentative components and relations?

This question is closely intertwined with RQa, as the vocabularies used in the crowdsourcing task depend on how this task is designed. As we are dealing with argument annotation, which is not something that has been studied extensively in crowdsourcing literature, we intend to elaborate on the formulation of and instructions for such a task.

(RQe) *Aggregation Method*: How can we combine the results from the crowd and save them so that they are reusable in the future?

While this question might sound quite familiar to 3.1, there is a difference. Even though both questions regard the aggregation of worker answers, 3.1 aims not only at combining the annotator answers but also at the practicality of the format used when it comes to preparing them for future reuse.

After having found out during the pilots that the accuracy of crowdsourced annotations varies considerably for different workers, we reflected on potential improvements of our system. Two major problems we identified were that not every worker approached our task with the same vigour and that not everyone was willing to invest the intended amount of time, based on which we also set the reward, when working on our task — even though we clearly communicated the expected time required to complete the task. As a result, some workers' answers were much better than others, but still, we gave the same reward to each of them.

Workers who simply try to give an answer as quickly as possible, no matter whether it is correct or not, to get the reward with minimal effort, are often referred to as spammers (Demartini et al., 2017; Alonso, 2019). One way to reduce the risk of including these faulty answers in the final corpus, is an effective aggregation method, as described in the previous section. But, this does not solve the inefficiency problem we face. A

worker who finishes a task, which is supposed to take 30 minutes to complete, in two minutes is rewarded in full even if we end up not using any of their annotations. Another problem we identified was that even among benevolent workers, namely, those who tried to solve the task according to the instructions and without spamming, there are quality differences. The easiest solution to this problem would be to reject workers who provide bad answers, as suggested by Young and Young (2019). However, first of all, as a requester, we felt a responsibility to act fairly towards workers. Second of all, in the final system, where we do not know the correct solution in advance, it might not always be easy to differentiate between benevolent workers and workers who try to get the reward with minimal effort.

For these reasons, we intended to extend our system with filters to identify benevolent, high-ability workers at an early stage, which would allow us to curate argument annotated datasets not only effectively, but also efficiently. We came up with the idea to include a mechanism in our system to filter out workers who were going to annotate the scientific papers inaccurately. With this mechanism, we would only need to pay a filtered-out worker for the time they worked until they were filtered out, and only high-performing workers would actually annotate the text. Hence, if such a mechanism was successful, we could reduce the costs and increase the annotation quality at the same time. Accordingly, we state the following two hypotheses:

(H1a) *Spammer Filter*: Introducing a quality assurance filtering mechanism at the beginning of the crowdsourced AM annotation workflow that checks, with one basic question per annotation type, whether the annotators understand the difference between the argument component types (argumentative relation types respectively), influences the AM annotation accuracy positively, defined as the average F1 Score of aggregated annotations (which is computed based on crowdsourced solution and the ground truth solution by Lauscher et al. (2018b)).

In H1a we hypothesise that a short test with just one basic question per annotation type is enough to filter out spammers. Filtering out spammers should then lead to more accurate annotations by the crowd.

(H1b) *Ability Filter*: Introducing a quality assurance filtering mechanism at the beginning of the crowdsourced AM annotation workflow that checks, with a few specific questions of varying difficulty, whether the annotators are able to accurately annotate the different argument component types (argumentative relation types respectively), influences the AM annotation accuracy positively, defined as the F1 Scores of aggregated annotations (which is computed based on crowdsourced solution and the ground truth solution by Lauscher et al. (2018b)).

While this hypothesis is quite similar to H1a, there is an important difference. Here, workers do not just have to answer one basic question per annotation type. Instead, they have to annotate argument components (argumentative relations respectively) in short paragraphs. Based on how well a worker annotates these short paragraphs, we can assess whether this worker has the ability to accurately annotate paragraphs of scientific papers.

With our research question, including the five subtopics, and these two hypotheses in mind, we decided to divide our implementation into two main parts, namely, the annotation of argument components and the annotation of argumentative relations.

We pursued the creation of a modular framework, which is why we decided to build a new system without relying on an already existing annotation tool, so as not to restrict ourselves and also to enable an easy extension of the framework in the future. In addition to that, we decided early on to seek support from the crowd on AMT, as it is a well-established platform in the research community.

3.2 System Design

In this section, we describe the overall set-up of the experiments we conducted within the scope of this work. First, we introduce the dataset and the argumentation scheme we used in these experiments. Second, we describe how we designed the HITs. Third, we present the workflow of our crowd-powered system. And finally, we describe the experiments we conducted on AMT.

3.2.1 Dataset

In Sections 2.3.1 and 2.3.2, we introduced the three currently existing datasets of argument-annotated scientific publications which are publicly available. The dataset from Kirschner et al. (2015) contains articles in German, and their argumentation scheme is targeted at argumentation structures. Hence, it does include different argumentative relations (support, attack, detail and sequence), but the argument components are not further differentiated. Instead, in their view, every sentence is an argument component. The dataset released by Accuosto and Saggion (2019) includes annotated argumentative components and relations. However, it contains only abstracts of scientific papers and is also quite small, containing only 60 abstracts with a total of just 327 sentences. The dataset of Lauscher et al. (2018b), in comparison, contains a total of 10,789 sentences from 40 papers. Our framework is targeted at the annotation of argument components and argumentative relations in scientific publications in their entirety, which are written in English. Accordingly, we decided to build on the work of Lauscher et al. (2018b) as their dataset is the one that best conforms to our intentions.

Lauscher et al. (2018b) enriched the Dr. Inventor Corpus (Fisas et al., 2016) with fine-grained argumentative components and relations. The Dr. Inventor Corpus has been annotated for several rhetorical aspects by Fisas et al. (2016) and consists of 40 scientific articles — with a total of 10,789 sentences — from the domain of computer graphics, in English. Lauscher et al. (2018b) hired four annotators (three of which were non-experts and all of which were trained in advance) to extract argument components and argumentative relations from all 40 papers. The annotation process was conducted in five iterations. After each iteration, the annotators discussed cases in which they dis-

agreed. Based on these discussions, they created and continuously refined the annotation guidelines, which were then published in their final form by Lauscher et al. (2018c). In addition to that, the IAA was measured for every iteration as F1 Score — they explain that it does not differ much from Cohen's κ in a situation with many negative instances. In particular, the used two slightly different F1 Scores: one based on a weak calculation and one on a strict calculation. For the strict calculation, annotated components have to be correct in span and type, and annotated relations have to be correct for nodes, type and direction. The weak calculation allows for components (and nodes respectively) to be correct if they coincide by at least half of the length of the shorter component. The performance for the annotation of argument components was always higher (about 23%) than that of the annotation of argumentative relations. The annotation performance increased in every iteration in comparison to the previous one for both components and relations, with just one exception (iteration 3) — a reason for that is not given in the paper. In the first round, F1 Scores were approximately 20% - 31% (lower bound corresponds to strict F1 Score calculation and upper bound corresponds to weak F1 Score calculation) for components and 3% - 6% for relations. In the fifth and final round, F1 Scores were as high as 60% - 74% for components and 34% - 47% for relations.

Lauscher et al. (2018b) released the annotated corpus as a textfile which contains both the annotated argument components and the annotated argumentative relations. In the textfile, each line corresponds to one annotation. The annotated argument components are in the following format:

ID Type Start End Text

The ID starts with a "T" followed by a number. Type corresponds to the argument component type (*own claim*, *background claim* or *data*), Start and End are specified as the character indices based on the entire paper and Text corresponds to the actual content of the annotation. Argumentative relation annotations are provided in a similar format:

ID Type Arg1:ID1 Arg2:ID2

For argumentative relations, the ID starts with an "R" followed by a number. Type corresponds to the argumentative relation type (*supports*, *contradicts* or *parts of same*), ID1 is the ID of the first argument component (i.e. the relations source) and ID2 is the ID of the second argument component (i.e. the relations target). In addition to that, a separate file containing the entire paper is provided.

3.2.2 Argumentation Scheme Selection

To annotate arguments in the Dr. Inventor Corpus, Lauscher et al. (2018b) started based on the Toulmin (2003) model. After a preliminary study, they ended up using a simplified version of Toulmin's model. In general, a claim is an assertion or a hypothesis representing the opinion of an author (Lauscher et al., 2018c). Their new model, however, neglects component types which hardly ever occur in scientific papers and instead

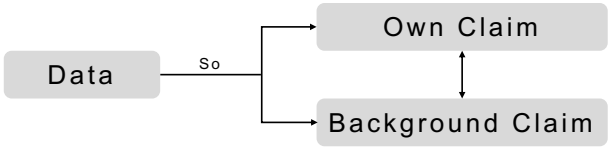


Figure 3.1: Model of an argument, as specified by Lauscher et al. (2018b, p. 41) on the basis of Toulmin (2003)

differentiates between two different types of claims — own claims and background claims — as they have a distinct role in scientific papers. Figure 3.1 visualises the components of the adjusted Toulmin model — the arrows indicate frequent relations, however, there are no restrictions regarding the way two argument components can be related. They further define that an argument component is at most one sentence and often even only a part of a sentence.
They then decided to complement their argument components with argumentative relations on the basis of Dung (1995). Their final argumentation scheme[1] consisted of the following three argument components:

1. *Own Claim*: An *own claim* is an author's statement which relates to his own work presented in the paper.

2. *Background Claim*: A *background claim* is a statement which is not related to an author's own work but rather is about a general belief within the domain.

3. *Data*: A *data* component is a fact which is used to support or contradict a claim, such as a reference, a measurement or an example. *Data* is sometimes also referred to as evidence, ground, premise or precondition.

These components are complemented by the following four relations:

1. *Supports*: A *supports* relation a directed relationship between a and b which exists if a functions as evidence to strengthen the proposition of b. Typically, a *supports* relation exists from a data component to an own claim component; however, other combinations might also be possible.

2. *Contradicts*: As a counterpart to a *supports* relation, a *contradicts* relation is a bi-directional relationship between a and b which exists if a and b contradict each other.

3. *Parts of Same*: A *parts of same* relation is used to mark argument components which actually belong to the same component which was split into two or more parts. This relation is bi-directional and intra-component, that is, it exists only inside a single component.

[1]For a comprehensive definition, including various examples, of all argumentative components and relations, we refer to the detailed annotation guidelines by Lauscher et al. (2018c).

4. *Semantically Same*: A *semantically same* relation is used to mark two different components which coincide content wise. Such a relation exists if an author of a scientific paper repeats the same argumentative statement twice, for example to emphasise an important claim. This relation is bi-directional and combines multiple instances of an argument component.

Contrary to *supports* and *contradicts* relations, the two relations *semantically same* and *parts of same* are non-argumentative. Still, in the remainder of this paper, we will continue using the wording *argumentative relations* for all relation types, including the two non-argumentative types, because all four relation types are needed to identify the argumentative structure of a text.

3.2.3 HIT Design

Even though there is much literature available on how to design a crowdsourcing task, this is not the case for the specifics of how to set up a HIT for the annotation of argumentative components and relations. Accordingly, the HIT design represents a major part of this work. However, the selection of the argumentation scheme has an impact on the design of the HITs. In accordance with the chosen scheme, it was clear from the beginning that the HIT(s) must somehow allow for the annotation of argument components and also for the annotation of relations between those components. As component extraction and relation extraction are often performed individually, we decided to implement two individual tasks. This decision is in line with the idea of crowdsourcing to break down a large project into smaller subtasks, and also in line with our goal of not overloading crowdworkers with too much technical information at once (Demartini et al., 2017).

We started by designing mock-ups for both task types, which can be found in Appendix B.1. Figure B.1a visualises the instructions for the argument component annotation HIT and Figure B.1b shows the HIT itself. We did not want to restrict the annotators, for example, by asking them single-choice questions, but rather allow them to use all of their skills to complete the task. By that, we intended to profit from the crowd's knowledge to the full extent. Therefore, we designed the HIT is a way which allows the workers to freely choose which part of the given text they want to annotate. Thereby, argument component annotations should be performed on a token level, as a component can be a sentence or just a part of a sentence, but never just a part of a token (Lauscher et al., 2018c). All annotations should then be visualised directly in the text, indicating the argument component type, as can be seen in the mock-up.

Similarly, we designed the HIT for the argumentative relation annotation task (whose mock-up is shown in the appendix in Figure B.1c) by just providing the underlying argument components and letting the workers do the rest. In particular, as can be seen in the mock-up, workers should be able to select any two argument components and annotate them with the corresponding relation type. All annotations should then be visible in the table on the right-hand side of the text.

To make sure that the worker still understands what they have to do, we elaborated

on instructions which on the one hand provide definitions for the different annotation labels, and on the other hand guide the workers by providing a manual of different steps which should be followed to do the annotations. The mock-up we designed for the instructions just includes the first part of the argument component annotation task's instructions to give an idea of how they should be implemented. In addition to that, we included an attention check in the instructions.[2] An attention check is a tricky but still very straightforward question, which intends to make sure that workers are paying close attention and do not speed trough. These attention checks are promising as they have both no costs and no benefits for high-reputation workers, and at the same time, they are identified by only 66% of the spammers (workers who are less likely to pay close attention and also less likely to yield high-quality data) (Peer et al., 2014). According to Sheehan and Pittman (2016), attention checks make sure that the obtained data is valid and of high quality. Last but not least, we included an informed consent which had to be accepted by the participating workers, towards the end of the instructions, which included some information about content, payment, and the time needed to complete the HIT.

3.2.4 Workflow Design

Figure 3.2 visualises the workflow we designed for our system. The input for the system is a scientific paper. During the preprocessing this paper is prepared so that it can then be annotated by the crowd. After the requester has created the HIT, it becomes available on AMT and workers can accept it. Workers who have accepted the HIT first have to go through the detailed instructions.[3] They have to complete two different types of filters; Our system should filter out spammers after the *spammer filter*.[4] Then, the *ability filter* should filter out those workers who are non-spammers but do not have the ability to accurately annotate argument components (argumentative relations respectively) in scientific papers.[5] Only workers who pass both filters can then go on to annotate the paragraphs.[6] A paragraph is a part of the preprocessed input paper which was selected for crowd annotation by the requester. Finally, as soon as all paragraphs have been annotated, the worker continues to the finish page where HIT can be submitted after having filled out the finish page survey.[7] Workers who completed the HIT are being paid by the requester. As soon as the crowd has completed all tasks, their answers are aggregated to the final dataset containing an annotated scientific paper.
We decided to build our own annotation tool for two reasons: First, even though annota-

[2] A screenshot of the attention check can be found in Appendix B.2.

[3] The instructions of the final system can be found in the appendix, see Figure B.3 for the argument component annotation task and Figure B.9 for the argumentative relation annotation task.

[4] The design of the *spammer filter* can be found in the appendix, see Figure B.6 for the argument component annotation task and Figure B.11 for the argumentative relation annotation task.

[5] The design of the *ability filter* can be found in the appendix, see Figure B.7 for the argument component annotation task and Figure B.12 for the argumentative relation annotation task.

[6] Detailed descriptions regarding the design and the implementation of the paragraph annotation can be found in Section 4.3.

[7] The design of the survey on the finish page can be found in the Appendix C.1.

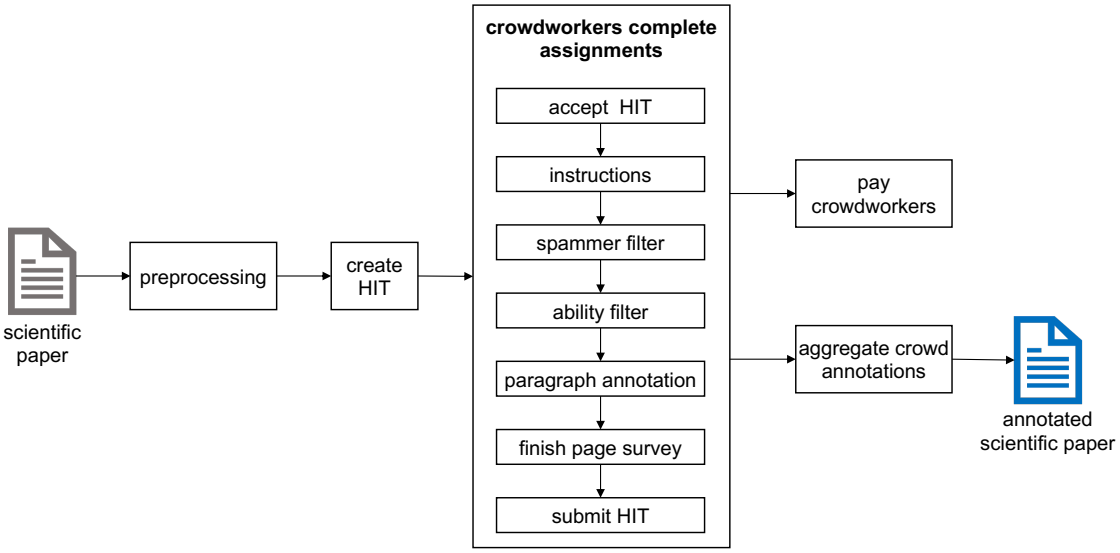


Figure 3.2: Workflow design

tion tools have been built deliberately to be used in the field of AM (Janier et al., 2014; Stenetorp et al., 2012; Petasis, 2014; Green et al., 2019), they have not been optimised for crowdsourced AM tasks and therefore do not comply with design recommendations for crowdsourcing tasks. In particular, these existing solutions offer too many functionalities and are not easily understandable. Further, these existing tools are either available as stand-alone solutions or as WebApps. Considering the fact that availability, accessibility and payment are crucial for crowdsourcing tasks, the integration of an existing annotation tool into a crowdsourcing platform would not be straightforward (Marcus and Parameswaran, 2013). Second, setting up our own annotation tool, including the embedding of the tool into AMT, has the advantage that it can be built in a modular way. This allows the system to be adapted in the future to set up new types of crowdsourced annotation tasks.

In a crowd-powered system, at least three annotators are needed to be able to break ties when two solutions are annotated the same number of times (Wacholder et al., 2014; Dumitrache et al., 2018). One of the most widely used techniques to do this in crowdsourcing is majority voting, as it is very easy to implement (Li et al., 2016). Majority voting is error-prone because in a situation where multiple workers just give the same answer to every question, for example, by always selecting the first answer possibility, wrong answers could mistakenly be considered to be true (Hovy et al., 2013). In addition to that, aggregating argument annotations is more challenging than the aggregation of answers from a conventional crowdsourcing task (Hung et al., 2013).

Hovy et al. (2013) developed the Multi-Annotator Competence Estimation (MACE) system to aggregate answers based on the trustworthiness of annotators[8]. MACE fol-

[8]MACE is a Java-based implementation which is available for download at: *https://www.isi.edu/*

lows an unsupervised learning approach to identify trustworthy annotators for whom neither the correct annotation nor the annotator quality has to be known in advance. By calculating trustworthiness parameters for annotators based different IAA measures, MACE realises if and when an annotator is spamming. This information is then used to aggregate worker annotations by predicting labels. To do that, MACE relies on expectation maximisation (Dempster et al., 1977). In this way, MACE achieves aggregated annotations which are likely to be correct.

While majority voting weights each annotator's answer equally, MACE takes a more sophisticated approach by considering different IAA measures to calculate trustworthiness parameters to learn whom to trust and when. In this way, MACE realises if and when an annotator is spamming and uses this information to aggregate worker annotations to a result which is likely to be correct.

Finally, we decided to aggregate crowdworkers' annotations with MACE (Hovy et al., 2013), as previously done by other researchers in the field of AM (Stab et al., 2018; Miller et al., 2019). In our case, in the argument component annotation task, this aggregation is done for each token, since workers perform annotations on a token level. In the argumentative relation annotation task, aggregation is done for each possible relation between two components.

After the annotations are aggregated, they have to be transformed to a machine-readable format. We chose to use the same format applied by Lauscher et al. (2018b) (see Section 3.2.1) because it is easily readable (for humans and machines) and, most importantly, because it includes the start and end character indices based on the entire paper. Including this information is crucial to be able to put several annotated parts of a paper back together in order to build an annotated corpus. In addition to that, based on the information included in the output, the annotations can easily be transformed to any other format desired, for example, to conduct benchmark experiments.

3.2.5 Experiments on AMT

Based on the mock-ups described above, we followed an agile methodology to implement our system, including the annotation tool which should be accessed by the crowdworkers through AMT. For the design and implementation of our HITs, we followed the general crowdsourcing best practises proposed by Sheehan and Pittman (2016) and Alonso (2019). We decided to run manual pilots and iteratively implement the system based on the gained insights in a first step, and test hypotheses to evaluate our system in a second step. We pretested all HITs with family and friends before running them in the productive AMT environment. Based on these tests, we obtained an estimate of how long, on average, it takes people to finish. Further, we incorporated feedback, such as design improvements or instruction clarifications, based on the informal feedback we received.

Throughout all of our AMT experiments, we always paid the crowdworkers based on

publications/licensed-sw/mace.

the estimated time to complete the HIT at hand. We set the hourly reward at $12. For all of the HITs we created (including the ones for the evaluation of the hypotheses), we made sure that only workers who had never done this type of HIT with the same paragraphs before could participate.

In addition to the reward we pay workers, AMT also charges a 20% fee on the reward (and bonus payment if this is used) that is paid to the workers.[9] Further, AMT charges an additional 20% fee on the reward paid to workers for HITs with ten or more assignments.[10] When requiring workers to be masters, AMT charges an extra fee of 5% on the reward paid to workers.

In total, 24 crowdworkers participated in the pilots. Crowdworkers were paid $6.80 for the argument component annotation HIT and $6.20 for the argumentative relation annotation HIT. The assignment duration (which is the maximum time a worker has to complete the task after having accepted it) was set to 90 minutes. Considering that all of our tasks were estimated to take approximately 30 minutes to complete, allotting a maximum time of 90 minutes should give the workers plenty of time to complete the HIT.

We decided to empirically evaluate the two hypotheses in two separate consecutive experiments, both of which contained a separate HIT for both task types (argument component annotation and argumentative relation annotation). The first experiment should evaluate hypothesis H1a and the second experiment should evaluate hypothesis H1b. The second experiment, which should evaluate hypothesis H1b, would only be started after the first experiment has been completed and its results have been analysed. That way, we left the option open to make use of the *spammer filter* when running the second experiment, in case hypothesis H1a would be substantiated in the first experiment.

For the experiment to evaluate hypothesis H1a, we decided to start with just four participants per HIT type. Based on the performances of these participants we would then perform a power analysis to get the sample size which is needed to obtain meaningful results. We defined the significance level to be 0.05 for the power analysis as well as for the assessment of the statistical significance of both experiments. We decided that we would perform the experiment for the evaluation of hypothesis H1a based on the sample size needed to achieve a power of 0.8. We split up the payment for this experiment into a fixed and a variable part (which is paid in the form of a bonus payment that is calculated based on a worker's achieved performance). For the argument component annotation task, we set the base reward to $2.20 and the maximum bonus payment (which is paid to a worker with a performance of 1.0) to $3.60. For the argumentative relation annotation task, we set the base reward to $2.20 and the maximum bonus payment to $3.00.

We decided to run the experiment to evaluate hypothesis H1b with ten workers per HIT type. For the argument component annotation task, we set the base reward to $2.40 and the maximum bonus payment (which is paid to a worker with a performance

[9] More information regarding the fees charged by AMT can be found at: *https://www.mturk.com/pricing*.

[10] For this reason, we created all of our HITs with less than ten assignments.

of 1.0) to \$5.60. For the argumentative relation annotation task, we set the base reward to \$2.10 and the maximum bonus payment to \$4.90. We decided to pay higher rewards in this experiment because the crowdworkers need more time to complete the *ability filter* than they need to complete the *spammer filter*.

Finally, we further planned to build our final system based on the insights gained during the experiments and also depending on whether the results supported the Hypotheses or not.

3.3 Evaluation Design

In this section, we describe the design of our analysis. First, we describe our method to aggregate answers from different annotators. Then, we introduce the performance metrics that we defined to assess our system. Further, we discuss the hypotheses that motivate our study. And finally, we say something about the data we selected for this study.

3.3.1 Annotation Aggregation in Experiments

For both task types (argument components and argumentative relations) in all of the experiments conducted, we aggregated annotations of three workers at a time for all possible combinations of workers. This means that every data point that resulted from the aggregation of crowd-annotations in the experiments represents the combination of three workers: for the experiment to evaluate hypothesis H1a, we only consider combinations of worker who are in the same group, as we intend to compare the means between two groups. For the experiment to evaluate hypothesis H1b, we take the average attempts the three workers needed, and for the performance we always take the average of those three workers' annotation performance.

We chose to aggregate worker groups of three for the hypothesis evaluation to be as similar as possible to the final system we had in mind. For the final system, we chose to have every paragraph annotated by three different workers as there is a need for at least three annotators to in order to break ties when two solutions are annotated the same number of times (Wacholder et al., 2014; Dumitrache et al., 2018). In our opinion, this approach balances the trade-off between accuracy and costs effectively, as having more annotators increases the accuracy (Stab et al., 2018), but also increases the costs.

3.3.2 Performance Metrics

Measuring the performance of the result of the aggregated annotations is not straightforward. There are various possible metrics and experts disagree as to which are best suited (Wacholder et al., 2014). Further, the metrics that are best suited might also depend on the context. In the following, we describe the metrics we use to evaluate the performance of the crowd in the pilots and in the experiments for the two hypotheses.

For datasets which were automatically created with a trained algorithm, performance is usually measured by calculating how accurately it labelled the dataset (Pustejovsky and Stubbs, 2012). In order to do that, the F1 Score (Van Rijsbergen, 1979) is the most widely used metric as an estimate of how well the algorithm will perform for unseen data in a multi-class setting. For human annotators, however, the quality of an annotated corpus is usually measured by the degree of agreement among the annotators, based on an IAA metric (see Section 3.3.2). An IAA calculation is a good performance approximation when the true solution is not known because it considers that an annotator's choice might always be random to some extent. However, the IAA might be misleading as it simply measures whether all of the annotators understood the instructions and applied them in the same way (Pustejovsky and Stubbs, 2012). So, having a high IAA score does not necessarily mean that the annotations are correct.

Even though the IAA measures the agreement amongst two or more annotators and is therefore believed to allow for conclusions regarding the clarity of instructions and the quality of obtained corpus, this might not be exactly the case in our situation, for two reasons. First, if all workers are lazy and do not annotate anything, the IAA would result in perfect agreement, even though the results do not bring us even one step closer to our goal. Second, even if the crowdworkers are not lazy, but there is something that is just very unclear (in the instructions, the design or just in the context or content of one specific example), they might all agree on the wrong annotation. This would, however, not be captured by an IAA even though it should lead to a lower performance score.

Even though we use the help of crowdworkers to annotate the text instead of letting an algorithm predict the annotations, we can still use the F1 Score to evaluate the performance of the crowdsourced annotations by comparing it to the ground truth annotations. The ground truth data that we hold our results against are the annotations published by Lauscher et al. (2018b). While the ground truth can certainly contain errors, it is still considered correct for our purpose, considering that it was created iteratively by several trained annotators (one expert and three non-experts).

There are several ways an F1 Score can be calculated in the multi-label case:

- Micro: The micro-averaged F1 Score is calculated globally by counting the total TP, FN and FP.
- Macro: The macro-averaged F1 calculates metrics for each label, and then finds their unweighted mean. This does not consider label imbalance.
- Weighted: Here, metrics are calculated for each label, to then find their average weighted by support, which is the number of true instances for each label. This changes the macro-averaged approach to take label imbalance into account.

In our setting, for the argument component as well as for the argumentative relations task, a macro-averaged F1 Score does not make sense as it does not take label imbalance into account — in our dataset there is some imbalance which is why we decided not to

use macro averaging. So if, for example, there is a paragraph in which many *own claim* components have to be annotated in opposition to just one *data* component, then we do not want to give this one *data* component the same weight as we give all the other *own claim* components. A weighted-averaged F1 Score may result in scores that are not between precision and recall. For these reasons, we chose to evaluate our results with a micro-averaged F1 Score (which in the multi-label case is equal to accuracy, micro-averaged precision, and also micro-averaged recall).

Other evaluation metrics which were developed specifically to be applied to AM, such as the CASS (Duthie et al., 2016b) or the UCP Porto Metric (Sá, 2019), aim at the calculation of a score which represents the full annotation of arguments in text. This means that even if the annotation has been performed in various steps, they compute a combined score for the argumentative component and relation performance. That way, the result of the computed argument component score serves as an input for the argumentative relation score computation. Thereby, they make sure not to penalise wrong component boundaries twice. In our case, however, we decouple the annotation of components completely from the annotation of relations. For this reason, we cannot just apply one of these methods, but have to decide on two individual performance metrics: one for argument components and one for argumentative relations.

In the argument component annotation task we designed, crowdworkers annotate sentences, fragments or tokens but never parts thereof. Also in the ground truth data annotations always include entire tokens and never parts thereof. Therefore, we decided to also calculate the performance on a token level. Calculating it on sentence level would not go hand in hand with the idea to let workers choose annotation boundaries on token level. Calculating the performance on a character level is also not desired as this would penalise the annotation of very short tokens in comparison to the annotation of long words consisting of many characters. Hence, we calculate the proportion of correctly classified tokens out of all tokens.

Regarding the argumentative relation annotation task, a metric that is perfectly suitable for our scenario does not exist. Therefore, we had to draw upon several of the existing measures to calculate the crowdworkers' performance. As the CASS does not consider the annotation type and an implementation is missing anyway, it is not suitable in our case. Instead, we followed the UCP Porto Metric's approach, However, as the UCP Porto Metric has been elaborated in a slightly different context and with a different underlying argumentation scheme, we had to adjust it to make sure that it would fit our scenario. Additionally, contrary to the UCP Porto Metric's project, we divided argument component annotation and argumentative relation annotation in two separate tasks. For the argumentative relation annotation task, we relied on the ground truth data from Lauscher et al. (2018b) as the argument components between which relations had to be identified. For this reason, the UCP Porto Metric's level 1, *textual matching*, was not relevant for us. Due to our argumentation scheme, the second level also could not be incorporated into the metric. As introduced in Section 2.3.3, the UCP Porto Metric's third level contains three steps. As with level 2, the third step of level 3 was not applied due to the difference in the underlying argumentation scheme. The third

level's steps 1 and 2, however, were applied. In particular, we used an F1 measure that we split into two parts, in line with steps 1 and 2 of the third level of the UCP Porto Metric, which we called *F1 nodes* and *F1 nodes and types*:

- *F1 nodes* calculates an F1 Score of all possible relations between the existing argument components (i.e. nodes) without considering the type or direction of the relations. Hence, the F1 Score is calculated with binary labels (1: relation exists, 0: relation does not exist), which is why FN are "filtered out" automatically as they are not considered by the F1 Score calculation anyway.

- *F1 nodes and types* calculates an F1 measure of all possible relations between the existing argument components and thereby considers the type as well as the direction (except for symmetric relation types, of course). However, in a multi-label case, the normal F1 measure would not calculate the performance as desired as all possible relations are considered, including FN. For this step, the UCP Porto Metric considers only relations whose nodes really exist in the ground truth, because wrongly annotated relations which do not exist in the ground truth are penalised in previous levels (Sá, 2019). For our case, however, the setting is slightly different, which is why we decided to filter out TN (i.e. relations which would be possible but exist neither in the ground truth nor in the annotated solution) before calculating the multi-label micro-averaged F1 Score. Neglecting this step would lead to a very high *F1 nodes and types*. Because, not filtering out TN means that every relation between two components which neither exists in the ground truth nor in the annotated data would be considered to be a correct annotation. For this scenario, the resulting F1 Score would be very high. Following the UCP Porto Metric (namely, just considering relations which exist in the ground truth) might also result in a very high F1 Score — the UCP Porto Metric actually accounts for that but in previous levels which we do not make use of. We chose to filter out the TN as it rewards correctly identified relations to the same extent that identified relations which do not exist in ground truth are penalised.

On the basis of the UCP Porto Metric's suggestion, we then calculate the final score for the argumentative relation annotation performance (*F1 total*) by weighting the two scores as follows:

$$F1\ total = \frac{F1\ nodes + 2 * F1\ nodes\ and\ types}{3} \tag{3.1}$$

While the computation of the argument component annotation performance, in terms of a multi-label F1 Score, is straightforward, the computation of the *F1 total* is more complex. Several special cases have to be considered to appropriately compute it. Symmetric relations are considered to be correct irrespective of the direction in which they were annotated. But at the same time, they should only be counted for one direction to not wight them stronger than the *supports* relations, (which are directed relations). Further, including one or the other, of two components which are related with a *parts of same* relation, should both be correct. And again, when checking different possible

annotations which are all correct, one has to make sure to not count a single annotation more than once. A detailed description of our implementation can be found in Section 4.5.

3.3.3 Evaluation of the Hypotheses

For hypothesis H1a, we extended our system to include a *spammer filter*[11] mechanism, which consists of one simple question per annotation type. The HIT[12] we designed for the evaluation of this hypothesis contains four parts, which were to be completed by the crowdworkers using a specific approach. First, the worker had to go through the detailed instructions. Second, the worker had to answer the simple questions, i.e. the *spammer filter*.[13] Third, the worker had to annotate argument components (or argumentative relations respectively) in three paragraphs of a scientific paper. Fourth, the worker could give voluntary feedback before submitting the HIT (in contrast to the pilots we did not include the survey for the experiments).

After having run the experiment, we evaluated the result with a two-sample t-test comparing the annotation performance of the *passed* group with the performance of the *failed* group (Welch, 1947). Those workers who did not answer all questions correctly were added to the *failed* group and those workers who did answer all questions correctly were added to the *passed* group. The null hypothesis for the two-sample t-test is that the means of the two groups, *passed* and *failed*, are equal. The alternative hypothesis is that the two means are not equal. The null hypothesis and the alternative hypothesis are the same for both HIT types, argument component annotation and argumentative relation annotation.

For hypothesis H1b, we extended our system to include an *ability filter*[14] mechanism, which consists of nine annotation tasks of varying difficulty (easy, medium and difficult), three per annotation type. The HIT[15] we designed for the evaluation of this hypothesis was structured similarly to the one we set up for the experiment concerning hypothesis H1a. Though instead of answering the simple questions the workers had to solve the specific annotation tasks, i.e. the *ability filter*.[16] The idea was to be able to detect workers who are good at annotating argumentative relations in scientific papers. For the sake of the experiment, no worker was filtered out.

Instead of assigning workers to two groups, we intended to find a causal effect between the ability to accurately annotate the different argument component types (argumentative relation types respectively) and the accuracy in the paragraph annotation by

[11] In the implementation, the wordings *spammer filter* and *filter step 1* are used interchangeably.

[12] The design of this extension can be found in the appendix, see Figure B.6 for the argument component annotation task and Figure B.11 for the argumentative relation annotation task.

[13] To avoid confusing the workers, since no one was actually filtered out for this experiment, we called this step *pop quiz*.

[14] In the implementation, the wordings *ability filter* and *filter step 2* are used interchangeably.

[15] The design of this extension can be found in the appendix, see Figure B.7 for the argument component annotation task and Figure B.12 for the argumentative relation annotation task.

[16] As in the first experiment, we also called this step *pop quiz*.

running a linear regression. We defined the ability to accurately annotate the different annotation types as the total attempts needed to complete the nine annotation tasks during the *ability filter* (since we aggregated worker combinations of three, as described below, we then took the three workers' average number of attempts and average F1 Score for each combination). The null hypothesis for the performed regressions is that the coefficient associated with the independent variables is equal to zero. The independent variable is the total number of attempts needed to complete the *ability filter* (as we aggregate combinations of three workers, the independent variable is then the number of attempts these three workers needed to complete the *ability filter*, on average). So, in other words, the null hypothesis is that the total number of attempts needed for the *ability filter* has no effect on annotation performance. The alternative hypothesis is that the coefficient is not equal to zero. In other words, the alternative hypothesis is that there is a relationship between the dependent variable and the annotation performance. The null hypothesis and the alternative hypothesis are the same for both HIT types, argument component annotation and argumentative relation annotation.

3.3.4 Data Selection

To reduce costs, we selected six paragraphs from one paper from the argument-annotated ground truth dataset. We chose the paper we used for our empirical analysis because it is of average difficulty compared to the other papers and because of the number of annotated argumentative components and relations. However, there were no significant differences anyway, as the 40 papers contained in the corpus are quite similar in the sense that they are all technical papers from the computer graphics domain, which are quite difficult to understand for someone without a computer graphics background. We chose the paragraphs to be used for our empirical analysis based on the fact that they contain various types of argumentative component and relation types. Three of the paragraphs[17] (which we used in the pilots as well as in the experiments) were extracted from the chapters *1 Introduction* and *2 Previous Work*. The other three paragraphs[19] (which we used in the pilots only) were extracted from the chapter *4 Determining Pre-Conditions*, in which the authors describe the preconditions for their implementation.

For budget reasons, we decided to let the crowd annotate only three paragraphs during the experiments to evaluate the hypotheses. The content of paragraphs 16-18 was very specific because the authors described technical details regarding their implementation. Paragraphs 2-4, however, were less specific because they contained more general elements of a scientific paper covering the introduction and previous work. For this reason, we decided to let workers annotate paragraphs 2-4 during the hypotheses experiments.

In all 40 papers of the dataset, there are just 44 *semantically same* relations. These are few in number compared to the 5,790 *supports* relations, 696 *contradicts* relations and 1,298 *parts of same* relations. More than half of the papers, including the one we

[17]Namely, paragraphs 2, 3 and 4 (as specified in the file *A11_ToBeAnnotated.json*[18] which was created during the preprocessing, as described in Section 4.2) from the paper *A11.txt*, see Lauscher et al. (2018b).

[19]Namely, paragraphs 16, 17, and 18.

Paragraph Name	**Sentences**	**Tokens**	**AC**	**AR**
paragraph_2	9	247	9	3
paragraph_3	10	229	4	1
paragraph_4	9	224	19	11
paragraph_16	9	232	17	10
paragraph_17	9	253	13	6
paragraph_18	10	225	6	1

Table 3.1: Overview of the paragraphs used in the experiments. The acronyms stand for: Argument Components (AC), Argumentative Relations (AR)

chose, do not contain any *semantically same* relations. We implemented our argumentative relation annotation HIT so that only the argumentative relation types which are actually contained in the paper could be annotated (*supports*, *contradicts* and *parts of same*) to avoid confusing the workers with an annotation type which does not even occur in any of the paragraphs. However, the annotation tool can easily be extended to include other annotation types in the future. The chosen paragraphs contain all three argument components which we introduced in Section 3.2.2, hence, we added them all as possible annotation labels for the argument component annotation HIT. Table 3.1 shows summary statistics for the six paragraphs we chose.

4

Implementation

This chapter covers the implementation of the crowd-powered system to annotate arguments in scientific papers. First, we outline our framework in general. Second, we describe how our system preprocesses the raw input. Third, we present the annotation tool we implemented. Fourth, we show how our system interacts with the crowdsourcing platform AMT. Then, we describe how we implemented the performance metrics to evaluate the hypotheses as outlined in Section 3.3. Finally, we present the final output generated by our system.

4.1 Framework

We built a framework that enables the annotation of argumentative components and relations in scientific publications with the help of the crowd. The user of the system, which in AMT terms is the requester, has to decide which part(s) of the paper the crowd should annotate. Then, the user can set the crowdsourcing platform's parameters, such as the reward for participants, profile requirements of the participants, or duration of the task. Finally, the crowdworkers' annotations are aggregated by the framework to a single argument-annotated file in a machine-readable format.[1]

As visualised in Figure 4.1, the system is composed of the following parts: First and foremost, the raw textfile containing a scientific paper is preprocessed, using a Jupyter Notebook[2], and then serves as an input to the annotation tool. We implemented our annotation tool in the form of a Python-based WebApp with the lightweight web application framework Flask[3], which we deployed to a Hobby Dyno on Heroku[4]. The Flask app is composed of different Blueprints. The annotation tool[5] itself and the admin

[1] We used the same format as Lauscher et al. (2018b) which is described in Section 3.2.1.

[2] This Jupyter Notebook can be found on GitLab: *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/blob/master/app/data/preprocessing.ipynb*.

[3] *https://flask.palletsprojects.com*.

[4] *https://www.heroku.com*.

[5] The annotation tool blueprint was named "textannotation" and can be found on GitLab: *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/blob/master/app/textannotation/textannotation.py*.

area[6] are both blueprints. These blueprints represent the back-end of the Flask app and are written in Python. Both blueprints are registered in the Flask app's __init__.py file[7]

A blueprint contains so-called routes which render different templates. These templates are all based on Flask's integrated template engine Jinja2[8]. As such, the front-end can profit from Jinja2's powerful template inheritance methodology and also from the functionality to pass variables from the back-end to the front-end directly when the front-end template is being rendered. But, in general, the templates are written in Hypertext Markup Language (HTML) and JavaScript (JS). The templates of the admin area blueprint are organised in a separate directory.[9] The textannotation blueprint's templates were organised in two directories, one for the argument component annotation task[10] and one for the argumentative relation annotation task[11]. A specification of the functionality of all of the templates contained in these directories can be found in Appendix A.3. For the management of the elements in the annotation tool as well as in the admin area we used jQuery[12]. For the design of these elements we used Bootstrap[13].

We decided to integrate our argument annotation tool After deploying the Flask app to Heroku[14], it was possible to directly embed the argument annotation tool in AMT, as is described in more detail in Section 4.4.3.

The admin area blueprint allows a user to request work on AMT and also to monitor HITs and assignment. Crowdworkers do not interact in any way with the admin area.

By pressing the submit button at the end of the annotation task, the results are being sent to AMT. As this submit button lies within the annotation tool, this functionality had to be implemented with a form which then sends a request to AMT. This saves the worker answers to AMT, from where they can be download to be saved locally. To download and save the worker answers, we implemented a Jupyter Notebook.[15] The qualitative analysis of the answers retrieved by the crowd is implemented with another Juypter Notebook.[16] We implemented the performance analysis of the crowd with Python scripts and R scripts, all of which can be found on Gitlab.[17] The aggregation of all worker answers

[6] The admin area blueprint can be found on GitLab: *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/blob/master/app/admin_area/admin_area.py*.

[7] The __init__.py file can be found on GitLab: *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/blob/master/app/__init__.py*.

[8] *https://jinja.palletsprojects.com*.

[9] The admin area's templates can be found on GitLab: *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/tree/master/app/templates/admin_area*.

[10] *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/tree/master/app/templates/textannotation*.

[11] *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/tree/master/app/templates/relationannotation*.

[12] *https://jquery.com*.

[13] *https://getbootstrap.com*.

[14] Step-by-step instructions on how to set up and deploy the Flask app to Heroku can be found in Appendix A.2.

[15] *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/blob/master/MTurk/save_worker_answers.ipynb*.

[16] *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/blob/master/MTurk/qualitative_answer_analysis.ipynb*.

[17] *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/tree/master/*

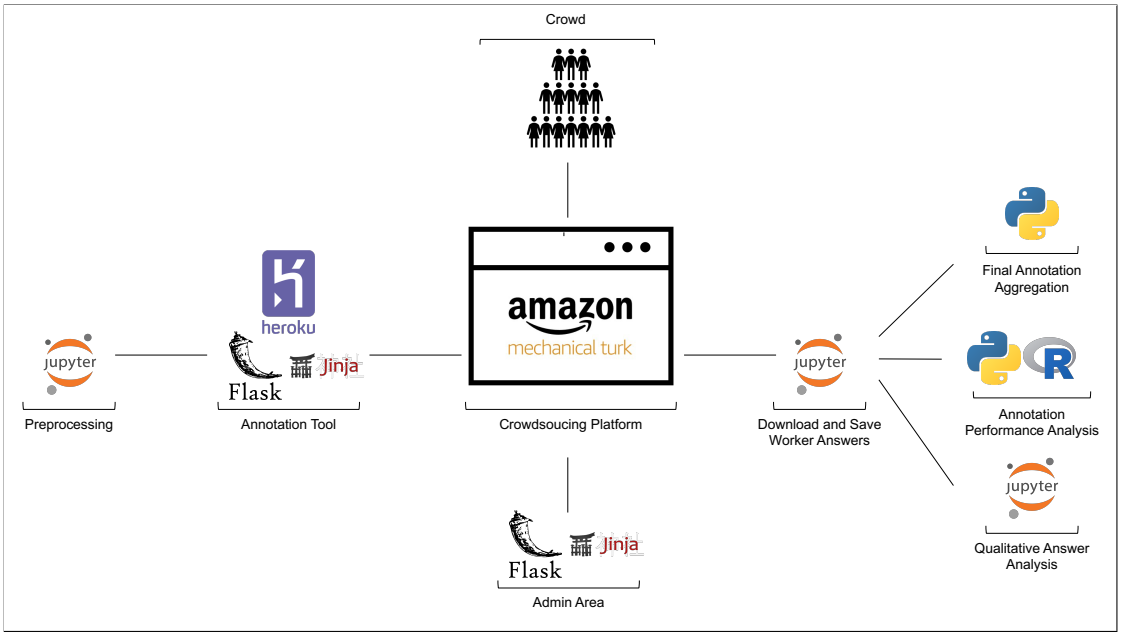


Figure 4.1: Visualisation of the framework's components

for the final system is done with a separate Python script and will be described in more detail in Section 4.6.[18]

Technical instructions regarding the set-up, the usage and the deployment of the framework can be found in Appendix A. In the following, we describe various parts of our framework in more detail.

4.2 Preprocessing

The input for our system is a plain textfile, which contains an entire scientific publication.[19] We implemented a Jupyter Notebook to preprocess the text in such a file to prepare it to be annotated by crowdworkers with our annotation tool.[20] To preprocess the data, the user has to provide the path to the textfile containing the scientific paper, and (if desired) the user can provide certain spans (i.e. character indices of for start and end of each span) which should not be considered for preprocessing. This is helpful to

AnswerAggregationAndPerformanceEvaluation.

[18]This script can be found on GitLab: *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/blob/master/AnswerAggregationAndPerformanceEvaluation/final_aggregation.py*.

[19]The scientific paper we used for the experiments can be found on GitLab: *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/blob/master/app/data/batch1/entire_paper/A11.txt*.

[20]This Jupyter Notebook has to be started manually before deploying the app to Heroku. It can be found on GitLab: *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/blob/master/app/data/preprocessing.ipynb*.

prevent certain parts of a scientific paper (such as the title page with affiliations, the paper's references or the appendix) from being annotated by the crowd. In our case, we chose the scientific paper *A11.txt* from Lauscher et al.'s (2018b) corpus and we chose to exclude everything before the start of the first chapter (which included meta-data, title, authors and affiliations, abstract and conference proceedings information). The notebook then automatically split the remaining part of the paper into paragraphs of at least 200 tokens each.[21] Paragraphs are set up in such a way that they only contain full sentences; this is why the specification of 200 tokens is not a hard limit — paragraphs usually contain a few tokens more to make sure that sentences are not cut, but never less than 200 tokens. We used nltk's (Bird et al., 2009)[22] *PunktSentenceTokenizer* and *TreebankWordTokenizer* to split the paper into sentences and tokens.

The result of the preprocessing step is a JavaScript Object Notation (JSON) file[23] which can be used as an input to the annotation tool. This JSON file contains the following metadata: original filename; number of characters, tokens, and sentences in the original file; number of characters that were excluded from the original file; and the spans which were excluded from the original file. In addition to that, this JSON file contains the created paragraphs, each of which contains the following metadata: start and end character index in line with the original paper and the total number of characters, tokens, and sentences the paragraph contains. Further, each paragraph contains information about the contained tokens, including the start and end character index, again, related to the original paper. This information is important to be able to aggregate the crowdsourced annotations and to create an output which can be reused.

To split a paper into paragraphs, we followed a tumbling window approach, that is, the preprocessing algorithm tumbles over the paper's text to create non-overlapping paragraphs. As we describe in more detail in Section 4.4, these paragraphs can then be used to specify which part of a scientific paper the crowdworkers should annotate.

4.3 Annotation Tool

In this section, we present the implementation of our annotation tool for each of the implemented HITs, namely, the argument component annotation HIT and the argumentative relation annotation HIT.[24]

In general, we present the implementation of the final system. However, in some situ-

[21] We chose for each paragraph to contain at least 200 tokens, but this is configurable.

[22] *https://www.nltk.org*.

[23] *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/raw/master/app/data/batch1/A11_ToBeAnnotated.json*.

[24] In addition to those two HITs, we implemented a further HIT to compensate crowdworkers who dedicated time to work on a HIT which they then, for whatever reason, did not submit. We designed this HIT based on a blog article by Amazon Mechanical Turk (r 10). This HIT was developed after a technical issue made submission impossible for crowdworkers when we created a HIT for the first time in the productive AMT environment at the beginning of the pilot studies.

ations we also mention the design we used during the pilots or during the experiments[25] as this is important to understand how we developed our tool further based on the insights we gained.

Creating a new annotation tool gave us the ability to choose which data we wanted to collect. First and foremost, we designed the tool so that the annotated paragraphs could later on be aggregated to an an annotation file covering an entire scientific paper. For that to be possible, a token's character index in the entire paper must be preserved during preprocessing and also during the annotation on the paragraphs. That was important to be able to produce a machine-readable output. Additionally, it allowed us to track the annotators' behaviour. We tracked various things in the form of a logger, such as the time they needed for the different parts of the HIT or whether they looked at the provided examples or not. However, this data did not influence the payment a worker received. Because what we eventually want are good-quality annotations and just because a worker leaves our task open for a long time or clicks on the example button many times does not necessarily imply better quality.

In line with the idea of keeping crowdsourcing tasks as simple as possible and letting a crowdworker do one thing at a time (Alonso, 2019), and also based on the chosen argumentation model, we decided to build two separate tasks: one for the annotation of argument components and one for the annotation of argumentative relations. In the following, we will present the design of both of these tasks in more detail.

4.3.1 Argument Component Annotation

In this section, we present the HIT we implemented for the annotation of argument components in natural language text. First, we describe the design of the component annotation in the tool. Then, we explain how we extended our tool to comply with the requirements of the two experiments we ran to evaluate hypotheses H1a and H1b.

Argument Component Annotation HIT Design

With regard to the classification of AM subtasks by Stab and Gurevych (2017a), this HIT includes both component identification and component classification. The argument component annotation HIT is designed to traverse the following three steps: instructions, paragraph annotation and finish page.

First, the crowdworker has to go through the detailed instructions. A screenshot of this HIT's complete instructions can be found in Appendix B.3. We followed Lauscher et al.'s (2018c) annotation guidelines to set up the instructions. The instructions contain *Examples* buttons to enable workers to look at specific examples for an annotation type. Screenshots of these examples are accessible in the Appendix B.3.

Second, after having read the instructions, the worker continues to the annotation step. There, the worker is shown the first paragraph which has to be annotated. As

[25] More details on how we performed the pilots and the experiments can be found in Section 3.2.5.

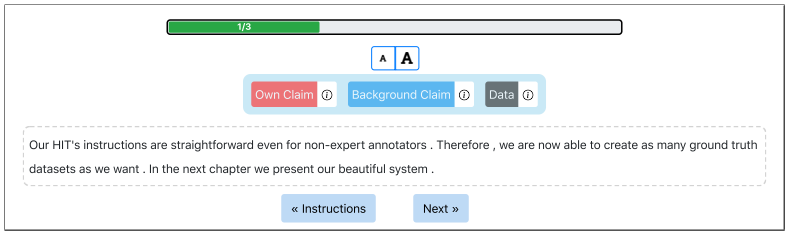
described in the HIT instructions, for each sentence, the annotator should decide whether the sentence is argumentative or not. If yes, the worker should decide which type of argument component is represented in this sentence and which are the exact boundaries. Figure 4.2 visualises a step-by-step example of an argument component annotation. A progress bar on the top of the page shows the worker how many of the paragraphs have already been annotated. Underneath it, the worker can adjust the font size to allow for different screen sizes, ensuring every worker can read the task content clearly. Above the paragraph lie buttons for the different annotation labels. Every label is accompanied by an info button which allows the annotator to look at examples for a specific label. Navigation buttons at the bottom of the page allow the crowdworker to go back and forth between the different steps, or to the previous or next paragraph in case more than one paragraph has to be annotated. The already made annotated are thereby always preserved.

Figure 4.2a visualises the initial page a worker sees after having read the instructions. As can be seen in Figure 4.2b, a text span is highlighted as soon as the worker selects it. Even selecting only a part of the word highlights the entire word as we only allow annotations on a token level. So-called *forbidden tokens* (see Section 4.4.1) are automatically trimmed from the selection in case they lie at the beginning or at the end of the worker's selection, to mitigate the risk of crowdworkers choosing wrong annotation boundaries which do not conform with the annotation instructions. Generally, the component type buttons are disabled as it does not make any sense make an annotation when nothing is selected. Selecting one or several tokens enables the component type buttons. After having selected a text span, the worker has to press the preferred component type button to continue with the annotation. As shown in Figure 4.2c, the crowdworker then has to fill out a modal to complete the annotation. In particular, the worker has to specify how certain they are that the annotation is correct, select some keywords, and provide an explanation describing what led the worker to their decision. The certainty and the keyword selection are mandatory, while the written explanation is voluntary.[26] Figure 4.2d shows the completed annotation in the paragraph. And finally, as can be seen in Figure 4.2e, crowdworkers can also edit or delete an already completed annotation.

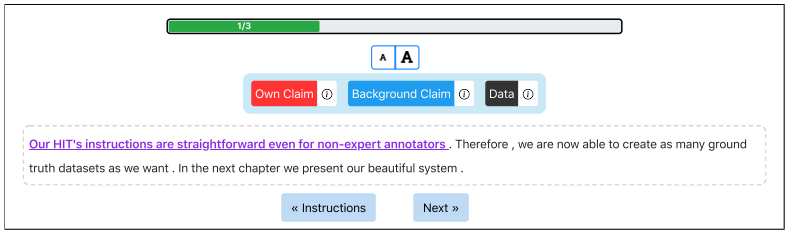
Third, after having annotated all paragraphs, the worker continues to the finish page. As can be seen in Figure B.5 (see Appendix B.3), this page is quite simple in the final system, as it just contains a text box to give the crowdworker the ability to provide feedback, and the submit button.[27] As soon as the worker presses the submit button, the HIT is submitted, the data is sent to AMT and the worker lands again on AMT, where they see the currently available HITs.

[26] During the pilots, also the written explanations was mandatory. But, as it takes a lot of time for the workers to provide a written explanation and we do not automatically process this information, we decided to make the written explanation voluntary in the final system.

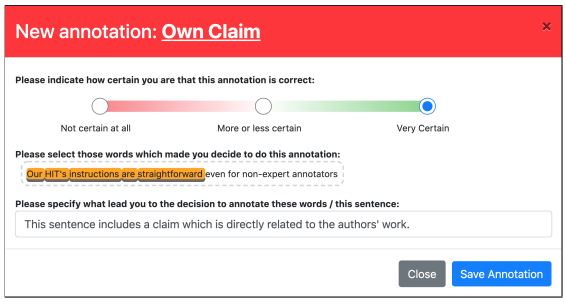
[27] In the pilots, the finish page included a more extensive survey to get detailed feedback from the crowdworkers. A screenshot of the extensive survey can be found in Appendix C.1; see Figures C.1 and C.2.



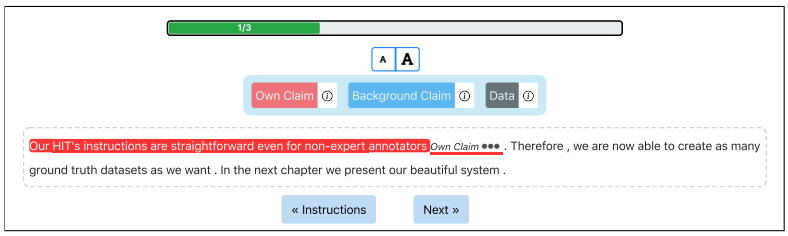
(a) The initial page of the paragraph annotation



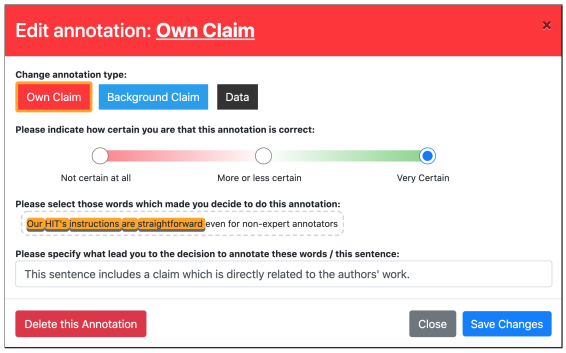
(b) The worker's selection is highlighted



(c) The worker selects a component type and fills out the form



(d) The annotation is displayed in the text



(e) The existing annotation can be edited or deleted

Figure 4.2: A step-by-step example of an argument component annotation

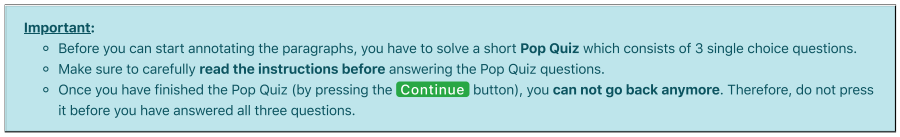


Figure 4.3: An infobox to inform workers about the pop quiz for experiment H1a

Spammer Filter for the Annotation of Argument Components

To evaluate hypothesis H1a, we extended our existing system to include a pop quiz[28] which could eventually function as a *spammer filter*. Even though during the experiments we did not filter out any workers, to be able to investigate the effect of the filter answers on performance, the goal was to use it to filter out spammers in the final system. Namely, we made two changes to our existing system. First, we added an infobox to the instructions informing the crowdworker that a pop quiz has to be completed before starting the annotation of the paragraphs. This infobox is displayed in Figure 4.3.

Second, we added the pop quiz right after the instructions. It included three single-choice questions, one for each argument component type (*own claim*, *background claim* and *data*), containing examples which were formulated based on the annotation guidelines provided by Lauscher et al. (2018c). A screenshot of the pop quiz, including the correct solutions, can be found in Appendix B.3. While a worker could always go back to look at the instructions, they could only start annotating the paragraphs after the pop quiz had been finished. Workers only had one attempt to solve the pop quiz before it disappeared. They were not informed whether they passed or failed the quiz. They were just asked to begin annotating the paragraphs.

The JSON file, which determines the content of the *spammer filter* questions including the answers, can be found on GitHub.[29] We use the same file for both task types.

Ability Filter for the Annotation of Argument Components

To evaluate hypothesis H1b, we extended our existing system to include a preliminary step consisting of nine short annotation tasks, which could eventually function as an *ability filter*. The goal of this filter is to have only those workers annotate our paragraphs who are good at annotating argument components in scientific papers. For the crowdworkers, we called this step a pop quiz. As we did in the *spammer filter*, we conducted two changes in our system in the *ability filter*. First, we added an infobox, which is shown in Figure 4.4, and second, we added nine short annotation tasks to be completed before annotating the three paragraphs.

The nine questions were formulated based on the annotation guidelines provided by

[28] To not confuse workers with a *spammer filter* that does not really filter out anyone, we named it pop quiz in the instructions. As we did not include the *spammer filter* in the experiment to evaluate H1b (the reason for that will be described in Section 5.3), we also called the *ability filter* pop quiz in the HIT's instructions.

[29] *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/blob/master/app/data/filterStep1.json*.

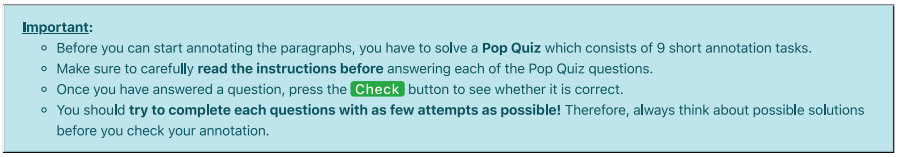


Figure 4.4: Infobox to inform workers about the pop quiz for experiment H1b

Lauscher et al. (2018c). For each component type, we formulated three questions of varying difficulty. The *easy* questions contained only one annotation, and the worker was told the type of the searched argument component. The *medium* questions contained zero, one or more than one annotation and the worker was not told the type of the searched argument component (in case there wasn't one). The *difficult* questions contained one or more annotations and followed a structure which was described as a special annotation case in the underlying annotation guidelines. All nine questions, including the correct answers, can be found in Appendix B.3

We provided immediate and dynamic feedback to each of the worker's answers. In case of a correct answer, the worker could directly continue with the next question. In case of an incorrect answer, the worker was told that the answer was incorrect and that they should try again. Depending on the number of already used attempts, we helped the worker to solve the question by providing a hint. The more incorrect attempts a worker gave to a question, the more specific the information contained in the hints became. Some examples of the hints we provided are visible in Figure B.8, which can be found in Appendix B.3. Eventually, after ten[30] incorrect attempts, the worker was shown the correct solution.[31] At this point, the worker can either skip the question by pressing the *skip* button or provide the correct annotation (based on the displayed solution) to continue to the next question.

For these short annotation tasks, we intended to use the same design as for the paragraph annotation, to avoid confusing the workers. However, to keep things as simple as possible, we exchanged the *edit* button for a *delete* button, as the workers' annotations are removed anyway as soon as they check whether their provided answer is correct. The JSON file, which determines the content of the *ability filter* questions including the answers, can be found on GitHub.[32] As can be seen in this JSON file, we use a list named `worker_IDs_who_passed` to keep track of those workers who have already passed the filter. We use the same file for both task types and workers have to be added to this list manually.

[30] We chose ten because in our opinion, a lower number of attempts would lead to the solution being provided too early and a higher number would protract the pop quiz. However, this number is configurable in the config file; see Section 4.4.1.

[31] With the exception that workers who provided exclusively empty solutions were not shown the correct answer. We wanted to exclude the risk of a worker finishing the pop quiz without trying to annotate the text even once, as this would result in a worker who may not even know how to annotate the text.

[32] *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/blob/master/app/data/filterStep2.json.*

4.3.2 Argumentative Relation Annotation

In this section, we present the HIT we implemented for the annotation of argumentative relations in natural language text. First, we describe the design of the relation annotation in the tool. Then, we explain how we adjusted our argumentative relation annotation tool to comply with the requirements of the experiments regarding the *spammer filter* and the *ability filter*.

Argumentative Relation Annotation HIT Design

With regard to the classification of AM subtasks by Stab and Gurevych (2017a), this HIT corresponds to the structure identification. Like the argument component annotation HIT, the argumentative relation annotation HIT is designed to traverse the instructions, paragraph annotation and finish page.

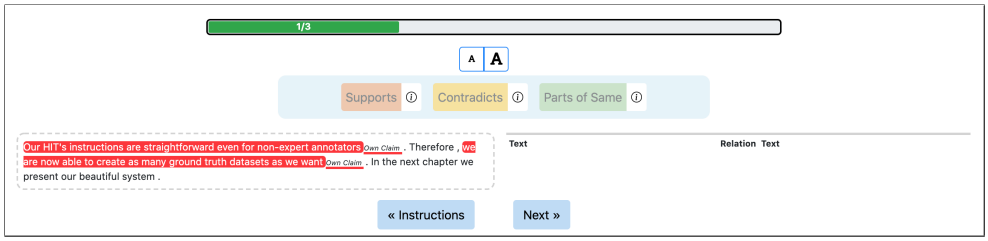
A screenshot of the argumentative relation HIT's complete instructions can be found in Appendix B.4, and examples of the different relation types can be found in Appendix B.4. Again, we followed Lauscher et al.'s (2018c) annotation guidelines to set up these instructions.

For the second step, the paragraph annotations, the worker is shown a text in which the relations are already highlighted. It is then the crowdworker's task to look at these components and decide whether they are somehow related and if so, to annotate the relation with the appropriate type.

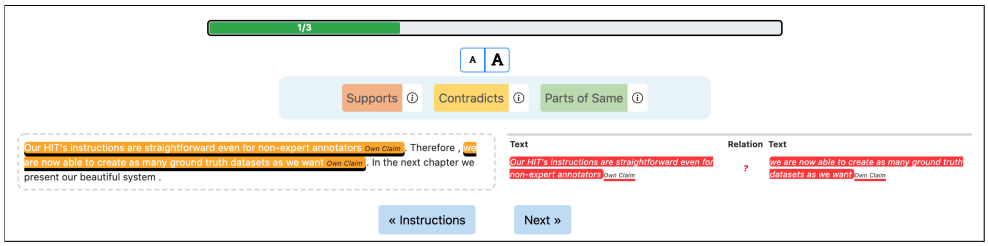
Figure 4.5 visualises a step-by-step example of an argumentative relation annotation. Figure 4.5a visualises the initial page a crowdworker sees when starting the annotation of the first paragraph. As soon as an argumentative relation between two argument components has been identified, those two components have to be selected, which makes them appear tentatively in the table on the right side of the text, as can be seen in Figure 4.5b. The **?** in the table's middle column indicates that the argumentative relation type still has to be defined to complete the annotation. So the worker has to select the preferred relation type button. Then, similar to the argument component annotation HIT, the modal has to be filled out before the argumentative relation can be saved, as shown in Figure 4.5c. Figure 4.5d visualises the completed annotation in the table. Then, crowdworkers can also edit or delete an already completed annotation, see Figure 4.2e.

After having annotated all paragraphs, the worker gets to the finish page, which looks similar to the one of the argument component annotation task (see Figure B.5 in Appendix B.3) with the only difference being that the questions are about relations and not about components.

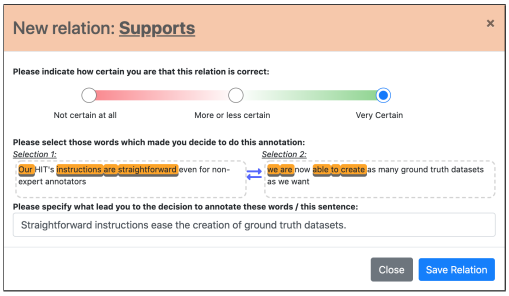
In the pilots, the finish page included a more extensive survey, as was also the case for the argument component annotation task. A screenshot of the extensive survey can be found in Appendix C.1; see Figure C.2.



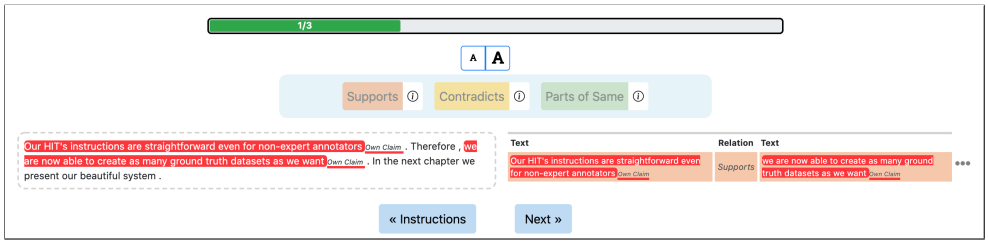
(a) The initial page of the paragraph annotation



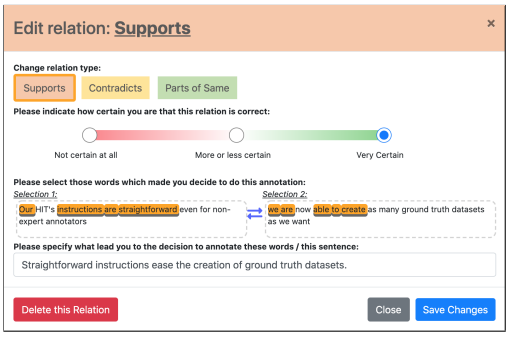
(b) The worker's selection is highlighted



(c) The worker selects a component type and fills out the form



(d) The annotation is displayed in the text



(e) The existing annotation can be edited or deleted

Figure 4.5: A step-by-step example of an argumentative relation annotation

Spammer Filter for the Annotation of Argumentative Relations

In line with the argument component annotation HIT *spammer filter*, we also extended the argumentative relation annotation task design for the evaluation of hypothesis H1a. Accordingly, we added the infobox to the instructions and implemented the pop quiz, that is, the *spammer filter*, to be completed after the instructions but before the paragraph annotation. This HIT's pop quiz, including the correct solutions, can be found in Appendix B.4. It includes one single-choice question per relation type. Also, for this HIT, workers could only start with the annotation of the paragraphs after the pop quiz had been finished, and they had just one attempt to answer the single-choice questions. They were not informed whether they had passed or failed the quiz, but once completed, the quiz could not be repeated.

Ability Filter for the Annotation of Argumentative Relations

As with the argument component HIT, we extended the argumentative relation annotation HIT with a pop quiz, that is, with an *ability filter*, to be able us to evaluate hypothesis H1b. The argumentative relation annotation HIT's *ability filter* was set up in the same manner as for the argument component annotation HIT. There were nine short annotation tasks of varying difficulty.

We formulated the question for the short annotation tasks based on the guidelines provided by Lauscher et al. (2018c). We selected examples from these guidelines which included both argumentative components and relations. However, we slightly reformulated them so that the solutions to the tasks could not easily be found online. For each relation type, we formulated three questions of varying difficulty. The examples we prepared were short and therefore only contained a few argument components. Therefore, most of these tasks were very easy, as the number of possible annotations was limited by the number of existing argument components in a paragraph. The *easy* questions contained only one relation which had to be found between two highlighted components, and the worker was told the type of the searched relation. Hence, the easy questions were straightforward, but in our opinion, still useful in helping the worker to understand from the beginning how the annotation task worked technically. The *medium* questions contained one relation between at least three highlighted argument components, and the worker was told the type of the searched relation. The *difficult* questions contained one or more relations and the worker was not told the type of the searched relation (or relations). All nine questions, including the correct answers, can be found in Appendix B.4.

The design is similar to the paragraph annotation; however, to keep the pop quiz as simple as possible, the *edit* button was replaced with a *delete* button. Also for this HIT's *ability filter*, we provided immediate and dynamic feedback to each of the worker's answers. Regarding the given hints, the functionality was implemented in the same way as in the argument component HIT (see Figure B.8 for specific examples). As this HIT's examples were easier, however, we set the number of wrong attempts needed before seeing the correct solution to eight[33] (but this is also easily configurable).

[33]In some examples there are only very few possible solutions due to the limited number of argument

4.4 Interaction with AMT

In this section, we discuss how our annotation tool interacts with AMT. To do this, we first say a few words about the configuration of a HIT. Then, we describe the admin area: the place where most interaction with AMT takes place. Last but not least, we show how our annotation tool is embedded in the AMT environment, that is, how the crowd interacts with our tool.

4.4.1 HIT Configuration

To facilitate running batches of HITs on AMT, we implemented a configuration file (config file), which allows the requester to specify the details of each run. This config file can easily be extended for future batches and even for the annotation of other labels.

Before creating HITs in the productive AMT environment, the Flask app, including the updated config file, must be deployed to Heroku.[34].

4.4.2 Admin Area

After the HIT has been configured and deployed, the requester can interact with AMT using the admin area we implemented to facilitate all of the tasks that are necessary to create and monitor HITs and assignments. After logging in to the admin area (with the password specified in the config file and the personal Amazon Web Services (AWS) access key[35]) it offers the following functionalities:

- *Configure and create HITs*: HITs are created by filling out a form. Figure 4.6 visualises the form including the configuration possibilities a requester has. Most importantly, the form allows the selection of a batch, as specified in the config file. Thereby, many HITs with similar content can be created. In addition to that, the paragraphs which should be annotated can be selected. Further, all specifications which are required by AMT to create a HIT can be set, such as title, reward, the duration of the HIT and also the number and duration of assignments. As a part of these specifications, the required profile of workers can be chosen, as can be seen in the form. There are several worker specifications which are predefined by AMT (such as whether a worker is a master[36] or not, the total number of

components. For this reason, we decided to reduce the number of wrong attempts needed before seeing the solution to eight.

[34]For technical instructions regarding the deployment of the system and the set-up of a HIT, see Appendix A.

[35]To set up an AWS account, visit *https://aws.amazon.com*. To set up an AMT requester account, visit *https://requester.mturk.com*. After having set up both accounts, they also need to be linked together at *https://requester.mturk.com/developer*.

[36]According to Amazon, masters are workers who "[...] have consistently demonstrated a high degree of success in performing a wide range of HITs across a large number of Requesters" (Amazon Mechanical Turk, 2018). When Amazon introduced the master qualification, they stated that "Master Workers are selected based on a number of attributes, including total assignments submitted, total earnings, tenure and diversity of work type" (Amazon Mechanical Turk, c 17). However, Amazon keeps the exact

lifetime HITs approved or the percentage of total lifetime HITs approved) as well as the existing manually created worker qualifications.[37] We used the "exclude" functionality (which is at the end of the form visualised in Figure 4.6) to make sure that workers who participated in a HIT of the same type before could not participate again. In this way, we could make sure that we always had new workers in all experiments.

- *Monitor HITs and Assignments*: To see how many assignments are still available to be accepted by crowdworkers and how many have already been completed, the admin area offers a requester the possibility to monitor HITs. Further, HITs can also be deleted or expired (an expired HIT becomes no longer available for new workers to accept but workers who are have already accepted it can continue with their assignment.). In addition to that, the admin area makes it possible to pay workers, either by accepting an assignment or by sending a bonus payment.

- *Create Additional Assignments*: After a HIT has been created, more assignments for the existing HIT can be added.[38]

- *Manage Worker Blocks*: Workers can be blocked, which makes sure that they cannot work for the requester who blocked them anymore. Even though workers can also be unblocked, it might negatively influence their AMT reputation (Amazon Mechanical Turk, n 26). In addition to that, Amazon tracks such blocks and may ban workers with too many blocks from working again on AMT (Sheehan and Pittman, 2016). So even if these suspensions are handled by Amazon, as a requester, we felt a responsibility to act fairly towards workers, which is why we chose qualifications over blocks for the execution of the experiments. Worker blocks were still used, however, to prevent workers from working for us again if the work they submitted was so bad that it was clear that they did not even try to solve the task, as suggested by Young and Young (2019).

- *Manage Worker Qualifications*: The admin area includes two possibilities for creating qualifications: either manually or automatically when creating a HIT. Workers can be associated with those qualifications. This gives the requester the possibility to make sure that workers, who are associated with a specific qualification, can be excluded from accepting a specific HIT. This allowed us, for example, to make sure that a worker participating in the control group of an experiment, did not participate in the treatment group. Another possible configuration for a requester would be to specify that only workers who are associated with a qualification can

mechanism used to choose workers to be assigned the master qualification a secret.

[37] More information regarding qualifications can be found on AMT's API reference: *https://docs.aws.amazon.com/AWSMechTurk/latest/AWSMturkAPI/ApiReference_QualificationRequirementDataStructureArticle.html*.

[38] Note that for HITs which were created with less than ten assignments, additional assignments can only be added if this does not extend the HIT to have ten or more assignments. More details on this are available on AMT's API reference:*https://docs.aws.amazon.com/AWSMechTurk/latest/AWSMturkAPI/ApiReference_CreateAdditionalAssignmentsForHITOperation.html*.

work on a HIT. Overall, using qualifications to manage experiment participators is preferred to using worker blocks as it does not influence a worker's reputation.

- *Consult Account Balance*: This page shows the currently available account balance, which helps the requester to keep track of the budget.

To access AMT from the admin area, we used Boto3[39] which is the AWS software development kit for Python.

4.4.3 Embedding of the Annotation Tool in AMT

There are various ways that questions and answers can be passed between requesters and workers on AMT. Since the traditional HIT templates provided by AMT are quite restricted, we decided to host the HIT content on our own website using a so-called "ExternalQuestion"[40, 41]. When creating a HIT as an "ExternalQuestion", the HIT is rendered inside an inline frame (iframe) on AMT, as can be seen in Figure 4.7. A blue border frames the embedded WebApp so that our annotation tool is distinguishable from the AMT website.

While working on the HIT, the worker's answers are saved in the browser's local storage. As data which is stored in the local storage does not have an expiration time, we implemented this functionality ourselves — making sure that it expires after 30 days (in case the worker submits one of our HITs). In addition, the key of local storage data is signed with the assignment's ID to make sure that a worker can work on several of our HITs at the same time. When a worker presses the submit button, which is part of our tool and therefore lies within the iframe, all of the worker's answers, including the annotations of the different paragraphs, are sent to AMT. In addition to that, as soon as a worker submits a HIT, the local storage is cleaned — that is, the data of the submitted HIT and all data that has already been there for more than 30 days is deleted from the worker's local storage. This is necessary for cases in which a worker starts and then aborts one of our tasks, as the local storage is only cleared when submitting the HIT. This implementation makes it possible for workers to refresh the page without losing any data (i.e. the status of their pop quiz, including all questions and answers, and all paragraph annotations and feedback are not lost on page refresh), and at the same time, the worker's local storage does not get flooded as it is cleaned after every submit. At the same time, this a worker to resume with the previous annotations if a HIT is aborted and then accepted again.

[39] *https://boto3.amazonaws.com/v1/documentation/api/latest/index.html*.

[40] See AMT's API reference for more details regarding "ExternalQuestion": *https://docs.aws.amazon.com/AWSMechTurk/latest/AWSMturkAPI/ApiReference_ExternalQuestionArticle.html*.

[41] The design of the HIT we implemented to compensate crowdworkers who did not submit the HIT was much less complex. For this reason, we used the so-called "HTMLQuestion" for this HIT. More information can be found at: *https://docs.aws.amazon.com/AWSMechTurk/latest/AWSMturkAPI/ApiReference_HTMLQuestionArticle.html*.

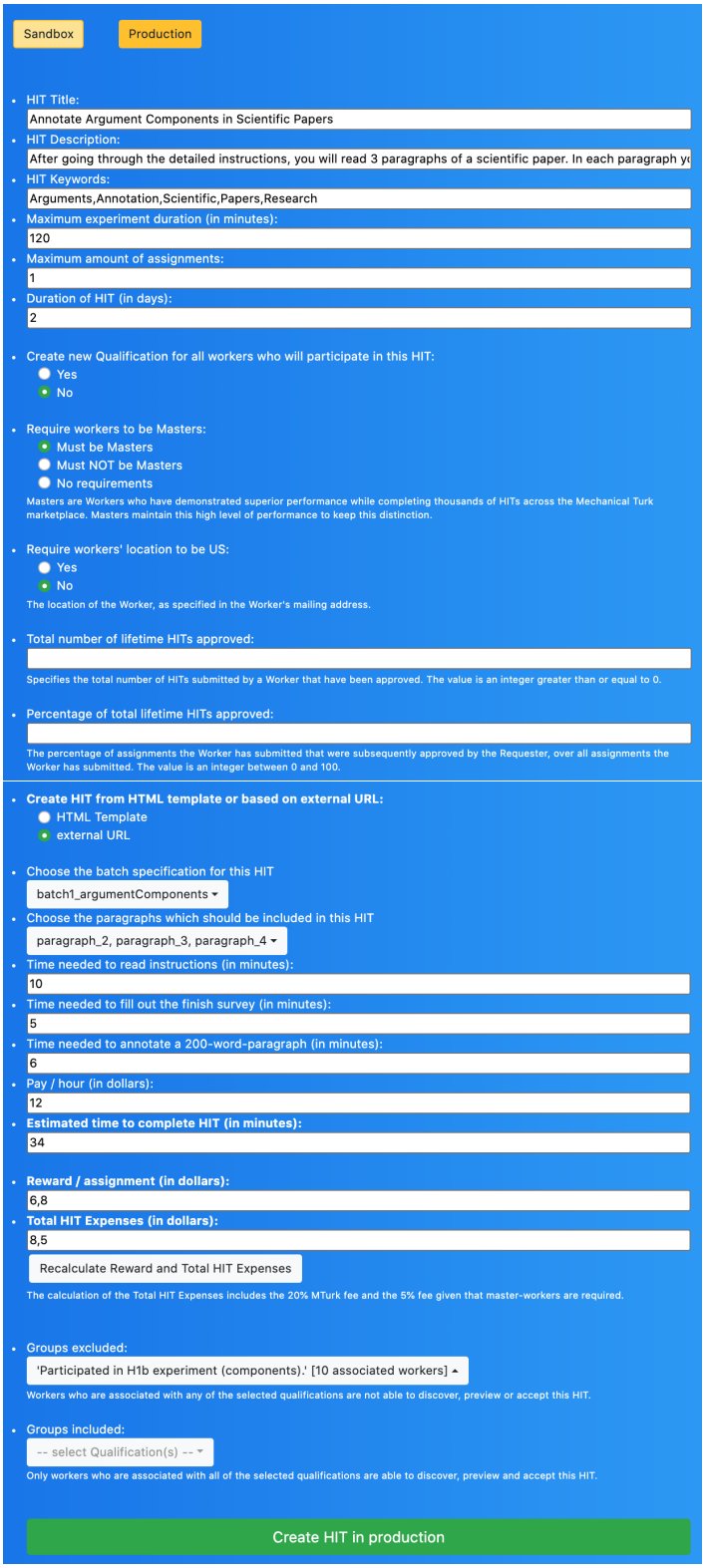


Figure 4.6: Form to create an AMT HIT in the admin area

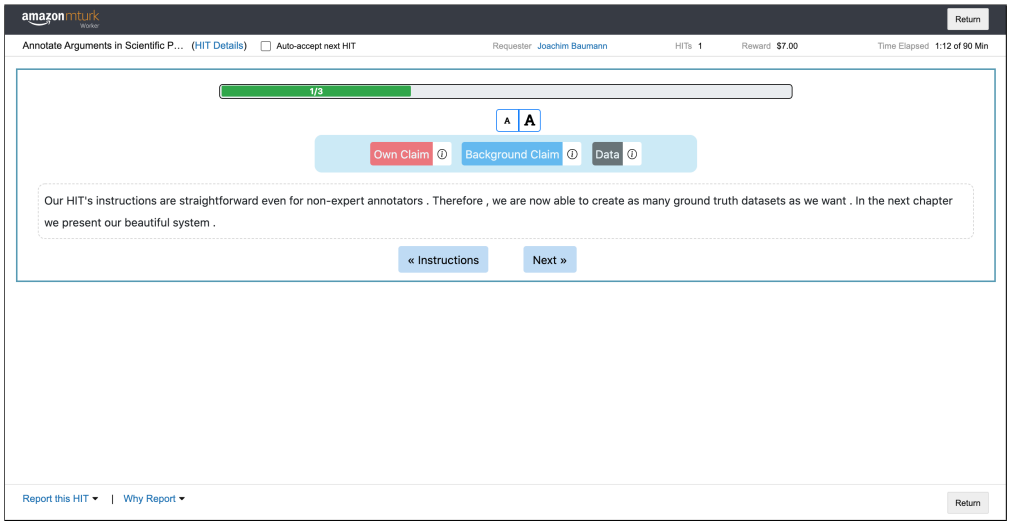


Figure 4.7: A HIT embedded in AMT

4.5 Data Analysis

As soon as a worker submitted a HIT, they can be download using a Jupyter Notebook.[42] The user must provide the HITId and the Notebook then automatically saves as JSON data so that we can further process them.[43] We implemented another Jupyter Notebook[44] for qualitative analysis of a worker's submission.

To compute the performance of argument component annotations, we first had to transform the crowdsourced annotations and the ground-truth annotations by Lauscher et al. (2018b) into the same format, which is described in detail in Section 3.2.1. Then, the computation of the micro-averaged multi-label F1 Score was straightforward.[45]

We wrote Python scripts to aggregate worker answers and to evaluate their performance.[46]

The computation of the performance of argumentative relation annotations was a bit more complicated.[47] For every paragraph, we first computed all possible relations —

[42].

[43]Downloaded worker answers are saved here (in the respective subfolder depending on the type of HIT it is: *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/tree/master/MTurk/WorkerAnswers/production*.

[44]*https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/blob/master/MTurk/qualitative_answer_analysis.ipynb*.

[45]Check the following Python script to see how this was done: *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/blob/master/AnswerAggregationAndPerformanceEvaluation/worker_answer_aggregation_components_H1b.py*.

[46]We used separate scripts for all experiments and for each HIT type. All scripts can be found in this directory: *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/tree/master/AnswerAggregationAndPerformanceEvaluation*.

[47]Take a look at the following Python script to see how this was done: *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/blob/master/*

which are all permutations of two distinct argument components, i.e. $n^2 - n$ — between the existing argument components and saved them in an object.

We did that by computing permutations of length two. This made sure that every combination of two components was considered twice, for both possible directions. This was necessary, as *supports* relations are asymmetric. Second, for every crowdsourced relation annotation (for every ground-truth relation respectively) we updated the object containing all possible relations to keep track of all annotations (ground truth as well as crowd annotations). Asymmetric relations (i.e. *supports* relations) were only updated for the exact components and direction for which they were annotated. Symmetric relations (i.e. *contradicts* and *parts of same* relations), however, had to be considered correct in any of the two directions (e.g. *a* contradicts *b* is the same as *b* contradicts *a*). Then, we also had to adjust the performance calculation for all *parts of same* relations.[48] Finally, when computing the performance, we made sure that symmetric relations were only counted once (i.e. for one direction), and that the reverse direction was omitted, while making sure that the same direction was considered for the ground truth as well as for the crowdsourced solution.[49] In addition to that, we also made sure that the relations we added during the performance calculation adjustment for *parts of same* relations were only counted once while at the same time allowing different, but still correct, annotations by the crowd.

4.6 Output of the Final System

We automated the aggregation of worker annotations by calling MACE from within the Python script (Hovy et al., 2013).[50] Then, they are saved in the same format applied by Lauscher et al. (2018b) (see Section 3.2.1). We chose this format because it includes the start and end character indices based on the entire paper. This makes it a machine-readable format which can be transformed into any desired format. As outlined in the Section 4.5 above, also for the aggregation of the crowd's annotations in the final system

AnswerAggregationAndPerformanceEvaluation/worker_answer_aggregation_relations_H1b.py.

[48] Assume that a paragraph contains four argument components (*a*, *b*, *c*, and *d*). Further assume that the ground-truth contains two argumentative relations (*a* supports *c* and *c* parts of same *d*) and that the crowdsourced annotations contain two argumentative relations (*a* supports *d* and *c* parts of same *d*). Note that the two *supports* relations are not the same, as the one in the ground truth supports *c* and the one in the crowdsourced annotation supports *d*. However, as *c* and *d* are part of the same argumentative statement (even though they are two argument components), the crowdsourced annotation is correct regarding the ground-truth data, which is why it should result in a performance of 1, measured in terms of *F1 total* as described in Section 3.3.2. Ensuring this calculation behaviour is what we mean by "adjust the performance calculation for all *parts of same* relations". We achieved this behaviour by adding the same relations to all other components which are part of the same argumentative statement, with the same initial relation source and type.

[49] Every combination of two argument components results in two possible directions. To ensure that symmetric relations are only counted once, we only consider the first occurrence and omit the second occurrence of the reverse relation of the same symmetric type.

[50] *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/blob/master/AnswerAggregationAndPerformanceEvaluation/mace_runner.py*.

we had to make sure to treat symmetric relations accordingly.

In our case, the crowd does not proceed exactly as the annotators hired by Lauscher et al. (2018b) did. The difference is that while they annotated both, argument components and then the argumentative relations between those exact components, the crowdworkers we employed annotated the argumentative relation based on the ground truth components by Lauscher et al. (2018b). For this reason, aggregating the annotation of both tasks does not make sense in our case. For this reason, we implemented a Python script which separately creates the output file containing the annotated scientific paper for both task types.[51] However, this script is prepared so that it can be applied also to a situation where the crowd annotates argumentative relations based on previously identified components. As a showcase of this aggregation to a final annotated scientific paper, we ran the script for three random worker answers from our experiment for both task types. The resulting files can be found on GitLab.[52]

[51]This script can be found on GitLab: *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/blob/master/AnswerAggregationAndPerformanceEvaluation/final_aggregation.py*.

[52]The scientific paper annotated with argument components (which is just a showcase and not an actual dataset) can be found on Gitlab: *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/blob/master/AnswerAggregationAndPerformanceEvaluation/annotated_scientific_paper_components.ann*.
The scientific paper annotated with argumentative relations (which is just a showcase and not an actual dataset) is also available on GitLab (the argument component IDs correspond to the component IDs from the ground truth dataset as these are the ones the workers saw when annotating the relations): *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/blob/master/AnswerAggregationAndPerformanceEvaluation/annotated_scientific_paper_relations.ann*.

5

Results

In this chapter, we report the results we obtained from the empirical analysis of our system. First, we introduce baseline performance scores. Second, we present the results of the pilots we ran. And third, we show the results of the two hypotheses, H1a and H1b.

5.1 Baseline Performance

As we implemented our own performance metrics (see Section 3.3.2) there is the need to somehow put them into perspective. For that, we introduce baseline performances, which are the performances an annotator would achieve by being lazy and not annotating anything. The argument component annotation HIT's baseline performance, measured in terms of F1 Score, is not necessarily 0 as we consider multi-label annotations. One can consider a "non-annotated token" as a token which is deliberately annotated with the label 'none' by the annotator. Only in the case where, in the ground truth, every token is annotated with something else than 'none', the baseline performance would be 0. Therefore, depending on how many argument components are contained in a paragraph, F1 Scores can be quite high even if the annotator does not annotate anything. More specifically, without annotating anything, an annotator would achieve an average F1 Score of 0.40 (0.42 for paragraphs 2-4 and 0.39 for paragraphs 16-18) for the six paragraphs we used for our experiments. Baseline performances for all paragraphs are visualised in Figure 5.1. *F1 Score with type* corresponds to the F1 Score as defined in Section 3.3.2. *F1 Score without type* corresponds to the annotator's F1 Score if we just consider the boundaries of an annotated argument component but not the type — so we just check if the annotation is 'none' or something else (own claim, background claim or data).

For the argumentative relation annotation HIT the baseline performance is 0, measured in $F1total$, because the calculation does not consider FN, as described in more details in Section 3.3.2.

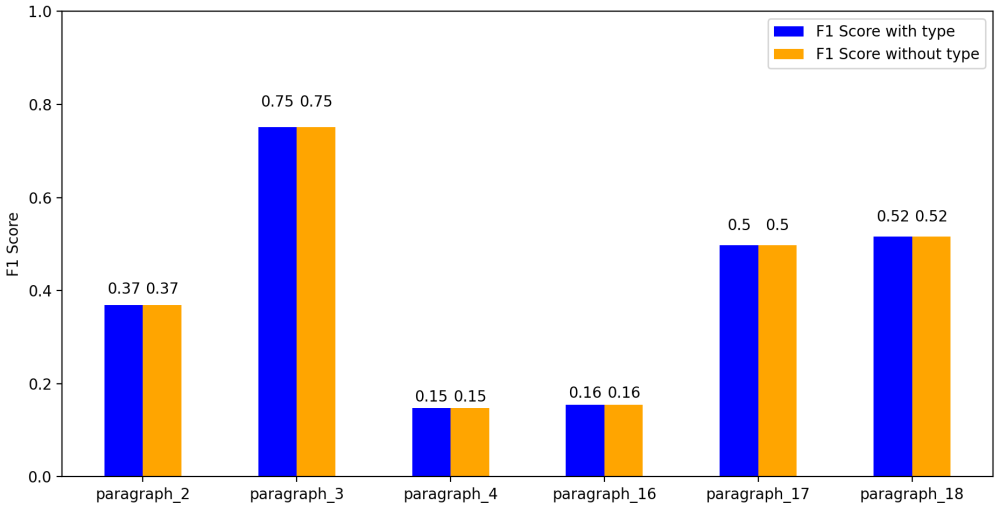


Figure 5.1: Baseline performance, in terms of F1 Scores, for the six paragraphs used in the experiments

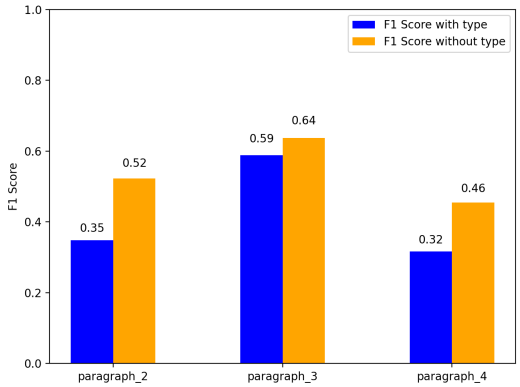
5.2 Pilots

We ran two pilot studies with a total of 24 crowdworkers on AMT. In the first pilot, ten workers annotated argument components in paragraphs 2-4. In the second pilot, four workers annotated argument components in paragraphs 16-18 and ten workers annotated argumentative relations (six of them in paragraphs 2-4 and four of them in paragraphs 16-18).

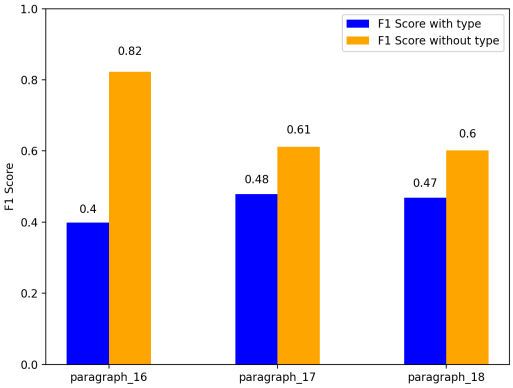
In the first pilot study, four out of ten workers failed the attention task. However, not all workers who passed the attention task achieved a good annotation performance and not all workers who failed the attention task were spammers. Six out of ten workers did not annotate anything.

For the second pilot study, we let workers annotate components in paragraphs 16-18 and, in addition to that, we also set up a HIT for the annotation of argumentative relations. However, in contrast to the first pilot study, we only allowed master workers to participate. But still, seven workers failed the attention task. However, in comparison to the first pilot, fewer workers did not annotate anything, namely, only two (both in the argumentative relation annotation HIT). For this reason, we decided to only allow participation for master workers for all following experiments.

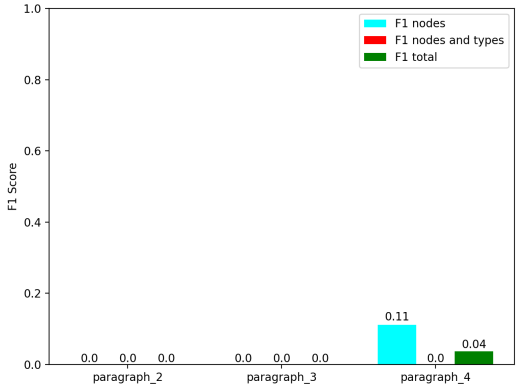
Figure 5.2 shows the performances achieved by the crowd in the two pilots. As we can see in the two Subfigures, 5.2a and 5.2b, crowdworkers achieved quite high F1 Scores for the annotation of argument components. However, considering that 60% of all workers did not annotate anything in the first pilot, these F1 Scores have to be taken with a grain of salt. Especially in paragraph_3, the crowdsourced annotations had quite a high F1 Score due to the fact that this paragraph contains only four argument components.



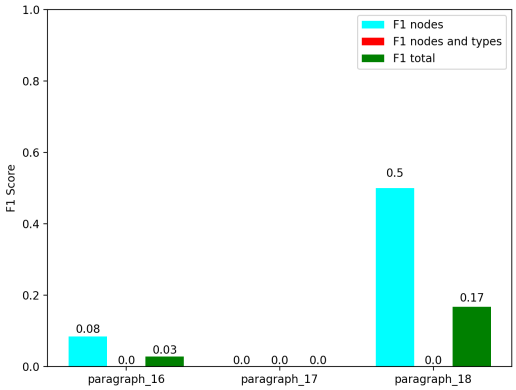
(a) Pilot 1: Argument component annotation performance (paragraphs 2-4)



(b) Pilot 2: Argument component annotation performance (paragraphs 16-18)



(c) Pilot 2: Argumentative relation annotation performance (paragraphs 2-4)



(d) Pilot 2: Argumentative relation annotation performance (paragraphs 16-18)

Figure 5.2: Crowd performance in pilots

As we can see in Subfigure 5.2b, master workers were not only less lazy, but they also performed better than the non-masters in pilot 1.

Figure 5.2 further shows that the crowdworkers had problems when identifying the correct argumentative relations (see Subfigures 5.2c and 5.2d). In paragraphs 2-4, none of the workers identified any of the 15 argumentative relations correctly. There are a total of 17 argumentative relations in paragraphs 16-18, and the crowd was only able to identify two of them correctly.

The crowdworkers' answers to the survey on the finish page can be found in Appendix C.1. Therefore, we only summarise the feedback by pointing out interesting insights without reporting on all answers the crowdworkers provided. This survey was only used during the pilot studies. As described in Section 3.2.5, a total of 24 workers participated in the pilot studies. However, we only analyse the survey answers of those workers who did at least one annotation, namely 16 out of 24 (eight per task type).

In general, workers reported that it was (at least more or less) clear what the task

is about. Therefore, we did not make any major changes to our instructions for later experiments. Five out of eight workers said that it was very difficult to understand the difference between *own claims*, *background claims* and *data*. For the relation HIT, only one out of eight workers said that it was very difficult to understand the difference between *supports*, *contradicts* and *parts of same* relations. Still, two workers (one in the component and one in the relation HIT) mentioned in the feedback that more examples would have been helpful. Therefore, we decided to add more examples to all of the argument component types and also to all argumentative relation types. Workers mostly agreed (five out of eight workers in both HIT types) that seeing more context would have been helpful to accurately annotate the given paragraph. Ten out of sixteen workers said that the payment they received was appropriate for the work they did, and six were undecided. In the argument component annotation HIT, half of the workers reported having needed more time than indicated. In the argumentative relation annotation HIT only three workers said that it took them more time than indicated to properly annotate the relations in all paragraphs. Two said that they needed less time, and three stated that they needed just as much time as indicated.

After the two pilot studies, we decided not to use an extensive survey on the finish page but rather just one text box where workers can provide feedback to the requester. An overview of all of the feedback the crowdworkers provided, by means of the text box on the finish page, throughout all of the experiments can be found in the appendix in Table C.1. Even though most workers provided feedback, it was often not very useful. However, we still wanted to give the workers the opportunity to easily give some feedback in case they felt the urge to do so. It does not necessarily increase the workers' time needed to complete the task, as providing feedback is voluntary.

We qualitatively analysed the time workers needed to complete the HIT based on the logger we implemented. The logging messages proofed what we already expected based on annotations: those workers who did not annotate anything finished our HIT very quickly. In general, they spent less than three minutes looking at all three paragraph that had to be annotated. Workers who did annotate something, however, spent much more time working on the HIT. We found that our estimates[1], based on which we defined a HIT's reward, were mostly accurate. Namely, workers needed roughly five to ten minutes to go through the instructions and then around five to ten minutes for each paragraph. However, workers did not need the full five minutes to go through the survey; in general, they needed around three minutes. Still, the total time needed, and hence the reward we set, seemed to be accurate (except for spammers), which is why we set the price for the later experiments based on the same time estimates. Additionally, we found that workers who did not annotate anything also did not look at any of the provided examples. Half of the workers who did annotate something also looked at the provided examples, mostly on the instructions page and not on the annotation page. Still, we decided to keep the design as is so that workers can check the examples of a

[1]We estimated that workers would spend ten minutes going through the detailed instructions, five minutes filling out the survey on the finish page, six minutes annotating argument components in a paragraph of 200 tokens and five minutes annotating argumentative relations in a paragraph of 200 tokens.

specific annotation label directly on the annotation page.

The assignment duration was set to 90 minutes. As no worker complained about not having had enough time to complete the HIT, we also set the assignment duration to 90 minutes for all of the following experiments.

We could see a tendency to maximise profit by spending as little time as possible per question in the first pilot but not so much in the second pilot. We believed that this was most likely due to the fact that we only allowed master workers to participate in the second pilot. Therefore, we continued to require the master qualification for workers participating in the later HITs.

After the two pilot studies, we decided to get rid of the attention task as it did not help us to tell good and bad workers apart. Instead, we expected that telling them apart would be made possible by the implemented filters in line with the two hypotheses, H1a and H1b.

5.3 H1a: Spammer Filter

We extended our system as described in Section 4.3.1 to run an experiment to evaluate hypothesis H1a. As described in detail in Section 3.3.1, we hypothesised that the introduction of a *spammer filter* would allow us to detect spammers at an early stage, with the ultimate goal of only having non-spammers annotate the paragraphs.

As described in Section 3.2.5, we planned to start with just four participants per HIT type to have just enough data to compute the necessary sample size to obtain meaningful results by performing a power analysis. Of the first four workers who participated in the argument component annotation HIT, none answered all filter questions correctly. Therefore, we created an additional assignment in order to have at least one worker in both groups, *failed* and *passed*. The fifth worker[2] answered all filter questions correctly. However, based on these fife participants, the *filter* group actually performed better than the *passed* group. The same was true for the argumentative relation annotation task. As the data indicated an effect which opposed our hypothesis, we decided not to obtain a power analysis, instead we performed it post-hoc.[3] We chose to perform the two experiments with a total of 14 participants each because of budget and time constraints. While this worked out fine for the argument component annotation HIT, we found ourselves waiting a few days for our argumentative relation annotation HIT to

[2]This worker was actually the sixth worker. The original fifth worker did not answer all questions correctly. With the goal of having at least one participant in both groups, to be able to perform the power analysis, and while keeping an eye on the available budget, we excluded the original fifth worker from annotating the paragraphs. As a result, we also did not obtain any performance results for this worker, which is why we do not include this worker in our results regarding the evaluation of Hypothesis H1a for the argument component annotation HIT.

[3]When we later evaluated the results, we still performed a power analysis, as then the effect pointed in the right direction (i.e. *passed* group performance was higher than *failed* group performance for both task types. We measured a power of 1.0 for the argument component annotation task and a power of 0.87 for the argumentative relation annotation task.

be completed. Ultimately, we were forced to end the experiment early (at a time when we still had just twelve participants for the HIT regarding the argumentative relations).[4] For this reason, a total of 26 master workers participated in the experiment to evaluate the Hypothesis H1a on AMT, 14 of which annotated argument components and 12 of which annotated argumentative relations in paragraphs 2-4 (i.e. the same paragraphs we had already used in the pilots).

For both HIT types (i.e. argumentative component and relation annotation), we divided the workers into two groups, depending on whether they answered all questions correctly or not. Those workers who answered everything correctly were put into the *passed* group and all the others into the *failed* group. As described in Section 3.3.3, we performed a two-sample t-test to evaluate this hypothesis (Welch, 1947). As described in Section 3.3.1, for both HIT types, we aggregated the annotations of three workers for each combination of workers in the same group and for each paragraph to calculate the performance. For this experiment, each worker was paid a base reward and, in addition to that, a variable bonus payment which was based on their individual performance. *This was explicitly indicated in the instructions.* For the argument component annotation task, the base reward was \$2.20 and the maximum bonus payment (which is paid to a worker with a performance of 1.0, measured in terms of F1 Score, as described in Section 3.3.2) was \$3.60. Workers were paid the percentage of the maximum bonus which was equal to their performance ($performance * maximumreward$). For the argumentative relation annotation task, the base reward was \$2.20 and the maximum bonus payment was \$3.00.

For the argument component annotation HIT, seven workers answered all three *spammer filter* questions correctly, which is why both the *passed* and the *failed* group were equally large, with several workers each. One worker answered two questions correctly, five workers answered just one of the questions correctly, and only one worker did not answer any question correctly. After the aggregation of the annotations for both groups, we had 105 data points to compare.[5] One data point is equal to the performance of the aggregated annotations of three workers (who are all in the same group) for one paragraph. Figure 5.3 shows boxplots of both groups for the argument component annotation HIT. We can see that the average performance of aggregated answers is 4.53% higher for the *passed* group than for the *failed* group. Namely, averages are 0.482 for the *passed* group and 0.4367 for the *failed* group. However, as shown in Table 5.1, the two-sample t-test revealed a p-value of 0.103. Hence, the difference in means we saw in Figure 5.3 is not statistically significant and we cannot reject the null hypothesis for the argument component annotation task. Average performances per paragraph for this experiment can be found in Appendix C.2; see Figure C.3a.

[4] Later we realised that the reason why we had to wait so long for more workers to participate was that we created additional assignments for an already existing HIT. Apparently, crowdworkers on AMT are very quick to accept and work on newly created HITs but are not interested in HITs which have already been on the platform for a few days. Therefore, from then on, we always created a completely new HIT instead of adding additional assignments to an already existing HIT.

[5] Due to the large sample size, we did not test for normality (Posten, 1984).

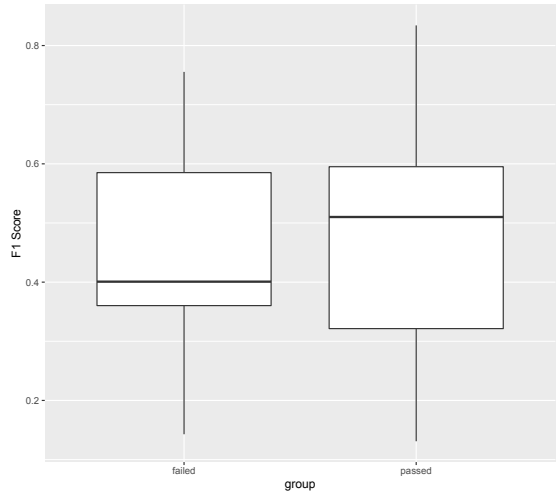


Figure 5.3: Performance by *spammer filter* group for crowdsourced argument component annotations

Mean of group *failed*	**Mean of group *passed***	**P value**
0.4367	0.482	0.103

Table 5.1: Two-sample t-test by Welch (1947): performance by group for crowdsourced argument component annotations

We further computed the performance of both groups similarly, but with one difference in that we did not consider whether an annotation's type was correct (we call this metric *F1 without type*). The results can be found in Table 5.2. So, for every token, we just checked whether it had been annotated or not. For the calculation of the *F1 without type*, the *failed* group achieved a performance of 0.528, the *passed* group achieved a performance of 0.587 and the two-sample t-test resulted in a p-value of 0.043.

Of the twelve crowdworkers who participated in this experiment's argumentative relation annotation HIT, only one correctly answered all three *spammer filter* questions. Three workers answered two questions correctly, three workers answered just one of the questions correctly and five workers did not answer any question correctly. After the aggregation of the annotations, we had 495 data points for the *failed* group and just three

Mean of group *failed*	**Mean of group *passed***	**P value**
0.5284	0.5954	0.0216

Table 5.2: Two-sample t-test by Welch (1947): performance (without considering annotation type) by group for crowdsourced argument component annotations

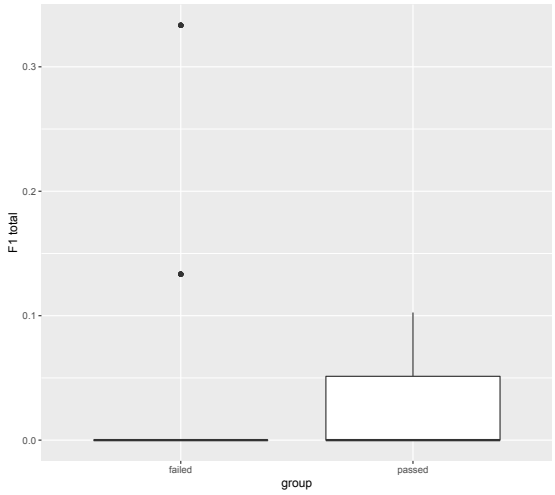


Figure 5.4: Performance by *spammer filter* group for crowdsourced argumentative relation annotations

Mean of group *failed*	**Mean of group *passed***	**P value**
0.0085	0.0342	0.531

Table 5.3: Two ample t-test by Welch (1947): performance by group for crowdsourced argumentative relation annotations

data points[6] for the passed group to compare. In general, the performances achieved were quite low in both groups. The means are visible in the boxplots in Figure 5.4 and also in Table 5.3. Even though the average performance of the *passed* group is 2.57% higher, we cannot reject the null hypothesis as the difference in means is not statistically significant (as can be seen with the high p-value of 0.531).

Average performances per paragraph for this experiment can be found in Appendix C.2, see Figure C.3b.

As we could not reject the null hypothesis for any of the two HIT types, we did not include the *spammer filter* in the experiment to evaluate the Hypothesis H1b. In addition to that, we obtained a high power for the experiments of both task types, meaning that if their was an effect we would probably have found it.

5.4 H1b: Ability Filter

We extended our system, as outlined in Section 4.3.1, to run an experiment to evaluate hypothesis H1b. As described in detail in Section 3.1, we hypothesised that the

[6] We could not aggregate any annotations as there was only one worker in the passed group. Therefore, for this group, one data point represents the annotation of one worker for one paragraph.

introduction of an *ability filter* would allow us to differentiate between with high and low cognitive ability at an early stage, with the ultimate goal of achieving a higher annotation performance by our crowd-powered system.

In total, 20 crowdworkers participated in this experiment on AMT and all of them annotated paragraphs 2-4. Ten workers annotated argument components and the other ten annotated argumentative relations. As described in Section 3.3.1, for both HIT types, we aggregated annotations of three workers for each combination of workers and for each paragraph to calculate the performance.

As in the *spammer filter* experiment, in this experiment each worker was paid a base reward and, in addition to that, a variable bonus payment which was based on their individual performance. *Again, this was explicitly indicated in the instructions.* For the argument component annotation task, the base reward was \$2.40 and the maximum bonus payment (which is paid to a worker with a performance of 1.0) was \$5.60. For the argumentative relation annotation task, the base reward was \$2.10 and the maximum bonus payment was \$4.90. Four days after having created the HITs for both task types, we found ourselves with just one completed assignment for the argument component annotation task type and two completed assignments for the argumentative relation annotation task type. We can only speculate on the reasons for this being the case. One possible explanation is that there are only a few master (workers who are interested in argument annotation tasks) and that these master workers had already worked for us in previous HITs of the same type, which prevented them from participating anymore. However, Amazon continuously analyses the workers' performances and, based on a statistical model, grants and revokes the master qualification to the highest performing workers (Amazon Mechanical Turk, 2018). Thereby, Amazon keeps the exact attributes upon which workers are rated, and the total number of active master workers in the system a secret. For this reason, we cannot know whether our explanation is true or not. Another possible explanation is that the base reward was too low in comparison with the time needed to complete the HIT to attract workers. As we did not know the exact reason for the low number of workers participating in our HITs, we chose to create new HITs with a number of changes. First, we changed the qualifications required, meaning we no longer required participating workers to be masters. Instead, we required the *percentage of total lifetime HITs approved* to be at least 95% and the *total number of lifetime HITs approved* to be at least 20,000.[7] Second, we adjusted the payment ratio to have a higher base reward and a lower variable reward, leaving the maximum possible reward unchanged. Namely, we changed the base reward to \$4.80 and the variable reward to \$3.20 for the argument component HIT. For the argumentative relation HIT, we changed the base reward to \$4.20 and the variable reward to \$2.80.

[7]Loepp and Kelly (2020) found that master workers have a significantly higher *total number of lifetime HITs approved* than non-master workers. We chose this value to be at least 20,000 as this is more than non-master workers have, on average. Loepp and Kelly (2020), however, found no statistically significant difference in the *percentage of total lifetime HITs approved* between master workers and non-master workers. We therefore, chose an adequately high percentage without setting it too high, as this again might have resulted in a situation where we would be unable to finish our experiment because of too few active workers on AMT who satisfy the requirements.

	Dependent variable:
	F1 Score
Number of attempts	-0.004^{***}
	(0.001)
Constant	0.663^{***}
	(0.041)
Observations	360
Note:	*p<0.1; **p<0.05; ***p<0.01

Table 5.4: Effect of number of attempts in *ability filter* on performance for crowdsourced argument component annotations

The ten crowdworkers who annotated argument components needed different numbers of attempts to complete the *ability filter* which consists of nine short argument component annotation tasks. On average, they needed 69.6 attempts (minimum 41 and maximum 98). Average performances per paragraph for this experiment can be found in Appendix C.3; see Figure C.4a.

Figure 5.5[8] graphically shows the effect of the number of attempts needed in the *ability filter* on annotation performance achieved. One data point corresponds to the average attempts of three crowdworkers and the performance of those three workers' aggregated annotations for one paragraph. Starting with the ten participating workers, there are 120 possible combinations to aggregate three workers' answers. As workers annotated three paragraphs, we end up with 360 data points. We further fit a linear model which is indicated with the red line. The negative slope shows that the more attempts crowdworkers need, the worse the annotation performance is.

As described in Section 3.3.3, we perform a linear regression to evaluate hypothesis H1b. In Table 5.4, the effect of the number of attempts on the dependent variable, the F1 Score, is shown. The table shows that, on average, a worker who needs ten fewer attempts to complete the argument component HIT's *ability filter* annotates argument components with a 4% higher performance. This effect is statistically significant. In other words, based on the 5% level of significance, we reject the null hypothesis for argument component annotations.[9]

The ten crowdworkers participating in the argumentative relation annotation HIT needed between 14 and 61 attempts to complete the *ability filter's* nine short argumentative re-

[8]Note that we added random noise to the plot to make it more readable. Namely, we jittered the data points by 0.1 in the x direction and by 0.01 in the y direction.

[9]The linear regression's p-value is 2.97e-09.

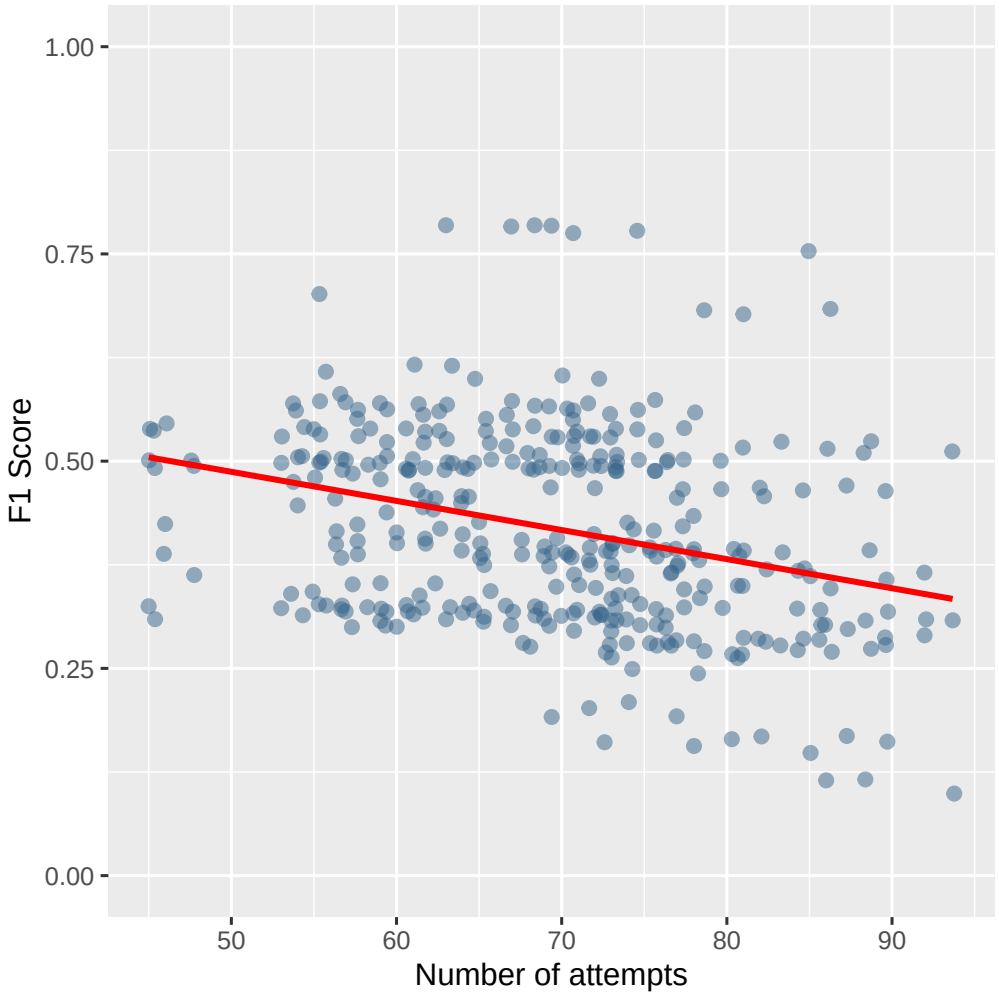


Figure 5.5: Visualisation of the effect of the number of attempts in the *ability filter* on performance for crowdsourced argument component annotations

	Dependent variable:
	F1 total
Number of attempts	-0.012^{***}
	(0.001)
Constant	0.546^{***}
	(0.052)
Observations	360
Note:	*p<0.1; **p<0.05; ***p<0.01

Table 5.5: Effect of number of attempts in *ability filter* on performance for crowdsourced argumentative relation annotations

lation annotation tasks. On average, they needed 37.3 attempts. Average performances per paragraph for this experiment can be found in Appendix C.3; see Figure C.4b.

Figure 5.6[10] graphically shows the effect of the number of attempts needed in the *ability filter* on annotation performance achieved. One data point corresponds to the average attempts of three crowdworkers and the performance of those three workers' aggregated annotations for one paragraph. Here, we have 360 data points based on the ten participants. The red line indicates the linear fit of those data points. As in the argument component annotation task, in the argumentative relation annotation task, the slope of the linear fit is negative, which means that the more attempts crowdworkers need, the worse the annotation performance is.

In Table 5.5 the effect of the number of attempts on the dependent variable, the *F1total*, is shown. The table shows that, on average, a worker who needs ten fewer attempts to complete the argumentative relation HIT's *ability filter* annotates argumentative relations with a 12% higher performance. This effect is statistically significant. In other words, based on the 5% level of significance, we reject the null hypothesis for argumentative relation annotations.[11]

[10]Note that here we jittered the data points by 0.1 in the x direction and by 0.01 in the y direction. This is why some *F1total* values are slightly smaller than 0 or slightly larger than 1.

[11]The linear regression's p-value is 2.28e-16.

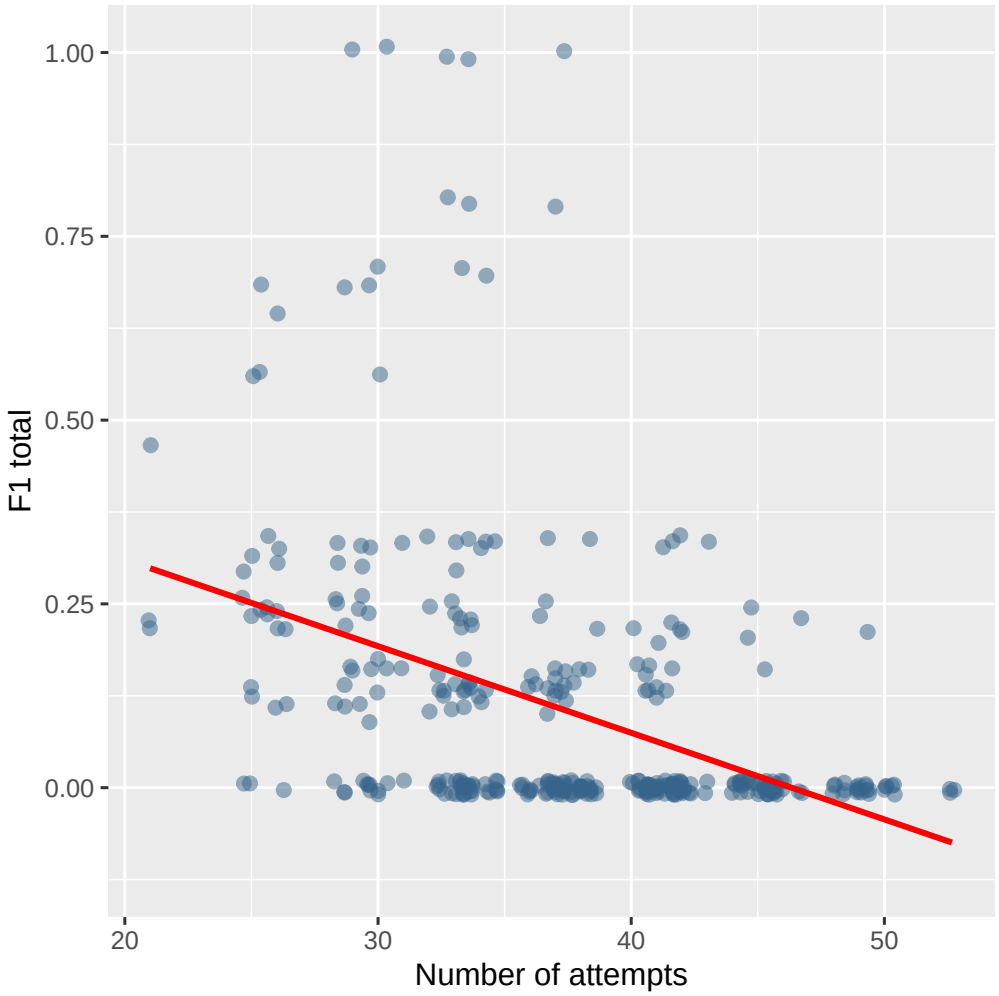


Figure 5.6: Visualisation of the effect of the number of attempts in the *ability filter* on performance for crowdsourced argumentative relation annotations

6

Discussion

In this chapter, we discuss the results described in Chapter 5. We analyse the answers of all 70 crowdworkers who participated in our experiments. The anonymised crowd answers are available on GitLab[1].

First, we discuss the results of the two hypotheses, H1a and H1b, and the implications on our final system. Then, we return to the five subquestions we outlined in Section 3.1, RQa-RQe, and discuss each of them based on the results we obtained. Finally, we conclude this chapter by answering our main research question.

6.1 Hypotheses

In this section, we discuss the results of the experiments we ran on AMT to evaluate the two hypotheses, H1a and H1b.

(H1a) Spammer Filter

As described in Section 3.1, we hypothesised that the introduction of a *spammer filter* at the beginning of the crowdsourced AM annotation workflow that checks, with one basic question per annotation type, whether the annotators understand the difference between the argument component types (argumentative relation types respectively), positively influences the accuracy of AM annotation. The results of the experiment we ran with a total of 26 master workers on AMT do not confirm this hypothesis. Even though, on average, workers who answered all questions correctly performed better in the annotation of the paragraphs, neither for the argument component nor for the argumentative relation annotation task were the results statistically significant. For this reason, we do not include a *spammer filter* in the final system.

For the argument component annotation task, 50% of the participating workers did not answer all three questions correctly. This shows that even for master workers, the quiz was not too easy. This is somewhat surprising as we tried to make the *spammer*

[1] *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/tree/master/MTurk/WorkerAnswers/production.*

filter's questions as short and clear as possible by formulating a basic statement for each of the argument components without including any special cases. For the argumentative relation annotation task, only one of the twelve participating workers answered all three questions correctly. This is very surprising as we also formulated the argumentative relation annotation *spammer filter* questions based on Lauscher et al.'s 2018c guidelines and tried to formulate them as simply as possible. For this reason, even if we had found a significant effect, the eligibility of the *spammer filter* for the argumentative relation annotation task would be questionable. This is because every worker who failed the filter would also have to be paid, which, considering that in our experiment eleven out of twelve workers were classified to be spammers, would result in very high costs for the detection of non-spammers.

We can only speculate about the low percentage of master workers answering all *spammer filter* questions correctly. As we saw in the pilots, many spammers were filtered out by enforcing the master qualification for participating workers. So, assuming that only a few of the workers participating in our experiment for H1a were spammers, we do not believe that spammers were able to pass the filter but rather that non-spammers were not able to answer the simple choice questions. However, this does not necessarily mean that crowdworkers are not able to annotate argumentative components and relations in scientific papers. It simply shows that not all non-spammers can correctly identify different types of argument components (argumentative relations respectively) — at least not by just reading the instructions, maybe after actually annotating a real sentence or paragraph they would be able to do so.

We found that when considering the *F1 Score without type*, the two-sample t-test reveals statistically significant results. However, even though passing the *spammer filter* is correlated with a higher performance (without type), this does not mean that the effect is causal. Because while the *spammer filter* checks whether a worker can correctly identify an argument component type, the *F1 without type* measures whether a worker identified the correct boundaries.

(H1b) Ability Filter

As described in Section 3.1, we hypothesised that the introduction of an *ability filter* at the beginning of the crowdsourced AM annotation workflow that checks, with a few specific questions of varying difficulty, whether the annotators are able to accurately annotate the different argument component types (argumentative relation types respectively), positively influences the accuracy of AM annotation. The results of the experiment we ran with a total of 20 workers on AMT support this hypothesis. For both task types, argument component annotation and argumentative relation annotation, the results are statistically significant at the 5% level (with p-values of 2.97e-09 for the component annotation task and 2.28e-16 for the argumentative relation annotation task). So there is an effect between the total number of attempts needed to complete the *ability filter* and the paragraph annotation performance. For the argument component annotation task, needing ten fewer attempts to complete the filter is associated with 4% higher performance, on average. For the argumentative relation annotation task, this

effect is bigger, and needing ten fewer attempts to complete the filter is associated with 12% higher performance, on average.

Comparing the average performance achieved by the crowd for the annotation of argumentative relations during the H1b experiment against the other experiments, we notice that they performed much better when the *ability filter* was in place. During the pilots, the crowd achieved performances of 0.0, 0.0 and 0.04 for the annotation of argumentative relations in paragraphs 2-4 (see Figure 5.2c). During the experiment to evaluate H1a, the crowd achieved performances of 0.01, 0.01 and 0.02 for the same paragraphs(see Figure C.3b). In contrast, performances were 0.08, 0.12 and 0.12 when evaluating the *ability filter* (see Figure C.4b). This rise in performance shows that the *ability filter* might even have a training effect on workers, because when workers start to annotate paragraphs, they have already completed nine short annotation tasks. Without the *ability filter* they would immediately begin annotation of the paragraphs after reading the instructions, without any practice.

As the results support the hypothesis, we include the *ability filter* in our final system for both the argument component and the argumentative relation annotation HIT. This filter has to be passed by crowdworkers before they can annotate paragraphs of scientific papers. We define a threshold, to differentiate whether the filter is passed or failed. The threshold defines the maximum number of attempts a worker is allowed to have before they are excluded from annotating the paragraphs. If a worker has too many incorrect attempts in the first few questions, they are directed to the finish page, where they have to submit the HIT before all short annotation tasks are even completed. Only workers who complete the entire filter using less than the defined threshold are allowed to continue to the paragraph annotation.

In addition to that, we also implemented the functionality to handle workers who have already completed the *ability filter*. On the one hand, workers who pass the filter once do not have to complete the filter again if they work on the same type of HIT.[2] On the other hand, we exclude workers who do not pass the filter by associating them with a qualification.[3]

6.2 Research Questions

In this section, we first discuss the five subquestions before we conclude the chapter by answering our initial research question.

[2]Keeping track of workers who pass the filter of a specific HIT type can be done by running a Python script, which can be found on GitLab.

[3]Excluding workers who fail the *ability filter* of a specific HIT type has to be done manually using the *Manage Worker Qualifications* page in the admin area, as described in Section 4.4.2.

(RQa) Design of Tasks: What types of tasks can we define to obtain accurate annotations of argumentative components and relations? In which of these tasks do crowd workers perform better?

Nguyen et al. (2017) included the crowd in their AM workflow. They let crowdworkers answer many single-choice questions to detect argumentative components and relations in paragraphs which were ranked with a high likelihood to contain arguments. They did this by asking the crowd specific questions concerning a predefined keyword. In comparison to Nguyen et al. (2017), we let crowdworkers annotate scientific papers for which the exact topic is not known in advance. For this reason, asking closed questions was not feasible, which is why we posted open questions. The advantage of open questions is that crowdworkers are not restrained regarding their answers. In particular, they can decide freely which token of a specific paragraph they want to annotate, they can decide the type of the argument component and they can also decide how many components they want to annotate. In the approach of Nguyen et al. (2017) crowdworkers just have to answer closed questions on sentence level. Thereby, they are able to design a less complex task, but at the same time, workers are much more restricted. Asking open questions does not restrict crowdworkers but bears the risk that they give incomplete answers in order to finish the task as quickly as possible to achieve the highest possible pay per time.

Letting workers annotate arguments in natural language text without restricting them, as would be the case if we were just to ask single-choice questions, is a highly complex task. Crowdworkers are mostly non-experts, so there is a need to make this naturally complex task as easy as possible for them. On the one hand, we cope with this high complexity by developing our own annotation tool, which is specifically designed for argument annotation by the crowd. On the other hand, we reduce complexity by splitting the annotation task into two parts: argument component annotation and argumentative relation annotation. Dividing the crowd's work into two parts makes sense, as others have also tackled different subtasks of AM without the need to look at the entire mining problem at once (Stab and Gurevych, 2017a). In addition to that, we let workers annotate only a few short paragraphs rather than having to annotate the entire paper at once, which is in line with the idea of crowdsourcing to break down a large project into smaller subtasks (Demartini et al., 2017) and also reduces the risk of assigning a very costly task to just one worker whose ability is unknown. Our task design is in accordance with Pustejovsky and Stubbs (2012), who stress that to successfully use the help of the crowd, the annotation task needs to be as easy as possible.

We point out in the task's instructions that workers should solve the annotation task step by step. For the argument component annotation task, workers should first read the entire paragraph; then identify argumentative statements, including their boundaries; and finally, annotate it with the correct component type. For the argumentative relation annotation task, workers are instructed to first read the entire paragraph, and then look at all different combinations of the highlighted argument components to identify pairs

of related components. And finally, workers should annotate the relation with the corresponding label. However, even though the workers are asked to follow the instructions exactly, the design of the task does not force them to do so. Instead, workers see a paragraph and are able to annotate freely. We believe that this approach empowers the crowd and allows workers to emphasise their skills to the fullest. Our task design does, however, enforce some behaviour: workers are forced to fill out the modal (the certainty and the keywords have to be provided, the textual explanation is voluntary) before they can save an annotation. We include this constraint to urge workers to add additional information, as just annotating some words without filling out the form might be done inconsiderately. This conforms to Alonso (2019), who found that making crowdworkers provide a reason for their answers is associated with fewer malevolent answers.

During the pilots, we realised that the design of our annotation tool allowed crowdworkers to cheat the system. In particular, for workers who tend to maximise profit by spending as little time as possible per question, it is possible to complete the task in a way that does not conform with the instructions. For example, workers can just go through the HIT without annotating anything or they can annotate something in just a few seconds without actually thinking about the correctness of the solution. In addition to that, they can also just annotate a part of the provided paragraph.

One potential solution is to reject workers who do not complete the task as they were expected to, and instead compensate them with a bonus payment for the time they invested in completing the HIT (Young and Young, 2019). However, we do not know the correct solution when using the final system to curate annotated datasets, which is why it would require manual work to look at workers' answers and decide which workers should be accepted and which should be rejected. In addition to this, from an ethical point of view, the expectation would need to be pointed out clearly so that workers know that they might be rejected if they do not strictly follow the task's instruction. This would make workers insecure regarding the probability of actually getting paid for working for us, which, in turn, might lead to workers refraining from choosing to complete the HIT.

Another possibility would be to include restrictions regarding the way a question can be answered, to eventually enforce preferred worker behaviour. For example, it could be designed that workers can only proceed to the next paragraph after they have spent a minimum amount of time working on a paragraph, thereby annotating a minimum amount of argument components (argumentative relations respectively). However, this solution is problematic for two reasons. First, deciding on how to design these restrictions depends on the question that is asked. For example, in the scope of our experiments, it would have been possible to only allow workers to proceed to the next paragraphs once they have annotated at least something. This is not possible in the final system as paragraphs might exist which do not contain any argumentative components and relations. In contrast to the experiments we did, the final system will be applied to new scientific papers where we do not know in advance how many argumentative components and relations are contained in a paragraph. Second, even if we could design restrictions which are suitable for the paragraph at hand, crowdworkers could still trick the system. For example, a worker could just quickly do some nonsense annotations to reach the

minimum number of annotations to be able to continue to the next paragraph or similarly, a worker could just work on another task while waiting until the minimum time that has to be spent on a paragraph has been passed. This is likely to happen because many workers accept more than one HIT at once. Then, they work on the ones with shorter completion times and in this way build up a queue of accepted HITs they want to complete (Sheehan and Pittman, 2016). While this would not lead to the intended result of workers spending more time working on our HIT, at the same time, it may lengthen the HIT's total completion time.

For these reasons, the final system we implemented continues to give workers the possibility to freely annotate as many argumentative components and relations in as much time as they need and we instead aim at solving the problem of undesirable worker behaviour which results in bad quality annotations on other levels.

The average crowd performance was quite low in all our experiments. In some cases the crowd was not even able to perform better than if nothing would have been annotated. As described in Section 5.1, the baseline performance for the argument component annotation task depends on the number of annotated tokens in the ground truth. During the pilot studies, the crowd's performance for the argument component annotation task was below the baseline performance for four out of six paragraphs.[4] When looking at the *F1 Score without type*, the crowd's performance was below the baseline performance for just one paragraph, namely, paragraph_3 — the reason for this is that this paragraph's baseline performance is very high (*F1 Score* of 0.75). During both of the experiments for hypotheses H1a and H1b, the crowd's performances for the argument component annotation task were below the baseline performance for just one out of three paragraphs (paragraph_3 in both experiments).[5] The same is true when looking at the *F1 Score without type*.

As described in Section 5.1, the baseline performance for the argumentative relation annotation task is always 0. During the pilot studies, the crowd's performance for the argumentative relation annotation task was 0 for two out of six paragraphs. [6] During the experiment to evaluate hypothesis H1a, the crowd's performance for the argumentative relation annotation task was never above 0.02 for all three paragraphs. [7] During the experiment to evaluate hypothesis H1b, the crowd's performance for the argumentative relation annotation task was better, namely, between 0.08 and 0.12. [8]

When comparing with the baseline performances, the average performances achieved by the crowd was quite low in all experiments and for both task types. For this reason, requesters who intend to crowdsource argument annotation in scientific publications are in desperate need of an effective quality assurance method to make sure that the crowd's result is useful and, hence, that the money invested to pay the crowd is worth it.

[4]Performances for all six paragraphs can be found in figures 5.2a and 5.2b.

[5]Performances for all three paragraphs can be found in Figure C.3a for experiment H1a and in Figure C.4a for experiment H1b.

[6]Performances for all six paragraphs can be found in figures 5.2c and 5.2d.

[7]Performances for all three paragraphs can be found in Figures C.3b.

[8]Performances for all three paragraphs can be found in Figures C.4b.

(RQb) Data Quality Assurance Method: Does the profile of crowd annotators influence the quality of results? What task assignment mechanism and aggregation method can be defined to increase annotation performance and efficiency?

Crowdworkers are diverse and they have different levels of motivation and ability (Demartini et al., 2017). Not everyone's motivation is based on the same source, and they also do not all have the same strengths and weaknesses (Difallah et al., 2018; Alonso, 2019). This was reflected in the experiments we ran on AMT. During our pilots, we quickly realised that there are big differences between workers in the sense that not everyone reads the instructions well enough, not everyone invests the same amount of time for equal tasks, and not every worker tries equally hard to actually solve the task at hand. This also led to annotations of very different quality. While some workers spent 50 minutes and identified more than 10 argument components per paragraph, of which many were correct, others spent just two minutes on the same task and did not annotate anything. Similar behaviour could be seen in the argumentative relation annotation task. We are not the first to discover this behaviour; it also poses a challenge for other crowdsourcing tasks (Alonso, 2019). In the crowdsourcing literature, workers who are just trying to finish a task as quickly as possible to get themselves a high reward for little work, are called spammers (Demartini et al., 2017; Alonso, 2019).

Instead of trying to enforce preferred worker behaviour by asking closed questions or by introducing additional restrictions to the annotation tool we implemented, we aimed at solving the problem of bad quality answers on other levels. In the scope of this work, we focus on three different approaches to assure data quality, namely, imposing profile requirements on workers, introducing a filtering mechanism and aggregating answers to a final solution which is likely to be correct.

As we have seen with the change in the number of spammers when we switched from no worker profile requirements in the first pilot to requiring master workers in the second pilot, the profile of crowd annotators does indeed influence the quality of results. Even though we do not exactly know how Amazon assigns high-performing workers with the master qualification, fewer of the workers were spammers as soon as we enforced the master qualification. However, when we changed from requiring workers to be masters to requiring them to have a *percentage of total lifetime HITs approved* of at least 95% and a *total number of lifetime HITs approved* of at least 20,000 (as described in Section 5.4), we did not see any difference (i.e. these workers did not perform particularly badly and the number of spammers did not increase). For an extensive analysis of the differences between master and non-master workers on AMT, we refer to Loepp and Kelly (2020).

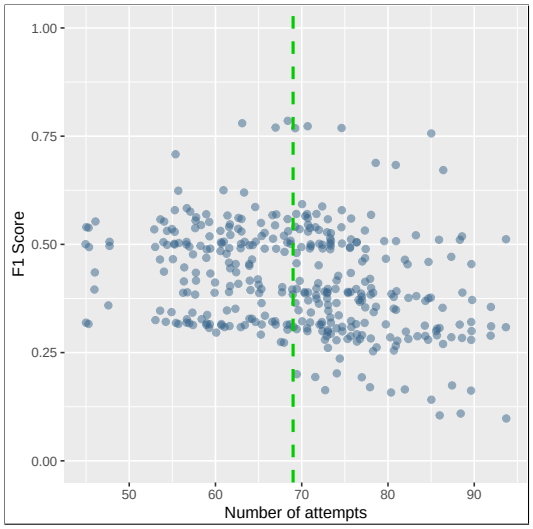
We did not find a statistically significant difference in performance between those workers who *passed* and those workers who *failed* the *spammer filter*. However, it is also possible that the *spammer filters* did not have an effect on performance simply because the questions asked are not detailed enough (as they contain just one short single-choice

question per HIT type) or because these questions are not similar enough to the actual annotation task.

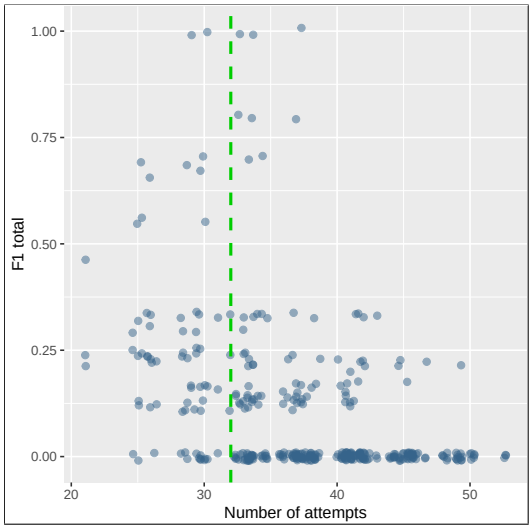
We used MACE (Hovy et al., 2013) to aggregate the crowdsourced annotations. In comparison to majority voting, which is one of the most widely used techniques for annotation aggregation (Li et al., 2016), MACE takes a more sophisticated approach by training an ML model to calculate annotator trustworthiness, based on which those annotations are considered for the aggregated annotation which is more likely to be correct. We are not the first to make use of this system. MACE has already been applied to similar aggregation problems in the past (Stab et al., 2018; Miller et al., 2019).

Based on the performance achieved by the crowd in the pilots, we can say that a crowd annotator profile selection in combination with a good aggregation mechanism is necessary but not sufficient to assure high annotation quality. According to Stab et al. (2018), hiring more crowdworkers leads to better annotation quality. However, our experiments show that to annotate argumentative components and relations in scientific papers, just hiring more crowdworkers is not efficient because the quality of the crowd's annotation varies depending on the workers' ability. Bad-quality annotations from low-ability workers cannot be used to curate a ground truth dataset. Therefore, just letting more workers annotate the same text is not efficient, as this means paying workers for answers which are not useful. Instead, we implemented an *ability filter* which aims to solve this problem for both HIT types. In contrary to the *spammer filter*, we did find a statistically significant effect on performance for the number of attempts needed to complete the *ability filter*. This shows that while spammers can be filtered out by requiring worker profiles, these worker profiles are not sufficient to filter out workers who perform badly when annotating arguments in scientific publications.

Based on the results from the experiment we ran, we suggest using a threshold of 69 for the argument component annotation task's *ability filter*. This threshold is visualised with the green dashed line in Figure 6.1a (which includes the same data points we presented in Figure 5.5). In this scatter plot, we see that none of the data points below this threshold have an F1 Score of less than 0.28. Further, the scatter plot shows that data points between 36 and 69 attempts perform similarly. There are, however, data points with really bad performance (*F1 Scores* of less than 0.25) but they are all above 69 attempts. We suggest 69 as setting a lower threshold would exclude many high-performing workers (with *F1 Scores* of more than 0.5) without excluding many low-performing workers (see the scatter plot where we can see that there are some data points with less than 69 attempts but still a low *F1 Score* of between 0.28 and 0.37). For the argumentative relation annotation task's *ability filter*, we suggest a threshold of 32. This threshold is visualised with the green dashed line in Figure 6.1b (which includes the same data points we presented in Figure 5.6). In this scatter plot, we see that above the suggester threshold there are many data points with an *F1total* of 0 (visible as the group of data points in dark blue). While there are some data points with a very low performance, even below the suggested threshold, it is not possible to choose a threshold which excludes all bad-performance data points.



(a) Argument component annotation HIT (threshold: 69)



(b) Argumentative relation annotation HIT (threshold: 32)

Figure 6.1: Suggested *ability filter* thresholds

But, these threshold suggestions have to be taken with a grain of salt. Setting a lower threshold is likely to result in a higher annotation performance. But at the same time, depending on the specified worker profile requirements, there may be many workers who are admissible to accept a HIT which is why a worker must also be lucky to be able to accept the HIT while there are still available assignments. So, even with the application of different thresholds, we cannot predict which exact worker(s) will work on our HIT. For this reason, also the exact performance is not predictable. In addition to that, the threshold might be set differently for different situations. For example, if the scientific paper is really easy to understand and the contained arguments can be identified even by low-ability workers, the threshold might be set higher. Then, the chosen threshold also depends on the situation and the preferences of the requester, that is, if the requester is willing to pay more for better annotation quality or if the low quality is acceptable in the given situation. For this reason, we implemented the thresholds so that they can easily be configured by the requester in the config file.

(RQc) Workflow Definition: How can we interweave machine and human computation optimally?

Figure 6.2 visualises the workflow based on which we implemented our final system. This workflow represents an end-to-end process that includes machine and human computation which can be used to create gold standard datasets for AM.

In comparison to the workflow design we described in Section 3.2.4, we here present the final workflow which was designed iteratively based on the insights gained during the experiments. This workflow is used for both the argument component annotation task as

well as for the argumentative relation annotation task. Batch and HIT configurations, as well as the content of the HIT (i.e. the instructions, the filter questions and the annotation tool design), differ as they are specific for each task type.

After the input paper has been preprocessed, the system user (who is also the AMT requester) can specify a batch configuration (in the config file) which has to be deployed to Heroku.[9] This configuration allows the user to create batches of HITs on AMT. For every experiment we ran, we configured a new batch in the config file.

Then, the requester can configure and create HITs (which all follow the configured batch specifications).[10] Thereby, the requester can specify various things which hold for just this specific HIT that is being created. The requester can choose the worker profile (for example, (non-)master workers), the paragraphs which should be annotated during this HIT, the reward that will be paid to workers who complete this HIT, and also worker qualifications. The requester can *exclude* workers who are associated with certain qualifications from participation. And the requester can also *include* qualifications to allow participation for those workers who are associated with a certain qualification.

While the crowd is working on currently existing HITs on AMT, the requester can monitor the progress in the admin area.[11]

As can be seen in the visualised workflow, the *spammer filter* and the survey on the finish page are not included in the final system.

The *ability filter* is included in the final system whereas the threshold of the filter is configurable (in the batch config) to give the requester the chance to govern the quality of results achieved by the crowd. Workers who do not pass the filter are compensated for their time but cannot go on to annotate the paragraphs and will not be able to work for on same HIT type again. In the figure, the *ability filter's* frame is dashed, indicating that workers who have already passed the filter do not have to do it again the next time they work on a HIT of the same type. Hence, in addition to the increased accuracy which can be achieved with the usage of the *ability filter*, it increases the efficiency of the system as a whole. However, even with a very low threshold, a 100% accurate annotation will most likely not be achieved, as not even expert annotators always agree (Mochales and Ieven, 2009; Habernal et al., 2014). Therefore, depending on the quality requirements of the problem at hand, the requester might need to postprocess the annotations to increase the quality of the final dataset. This postprocessing can be done manually by checking the solutions and adjusting or deleting incorrect annotations. For this process, the certainties, keywords and textual explanations of the annotation provided by the crowd might be helpful.

In addition to the displayed steps, the user can also block workers. This is not included in our workflow design because it is not standard procedure and should only be done in justifiable cases as it might harm a worker's reputation (Amazon Mechanical Turk, n 26).

In the relation annotation task, the paper is split in the same way as in the component

[9] Technical details of the batch configuration are described in Section 4.4.1.

[10] Details on how HITs can be configured and created can be found in Section 4.4.2.

[11] More details on HIT monitoring in the admin area can be found in Section 4.4.2.

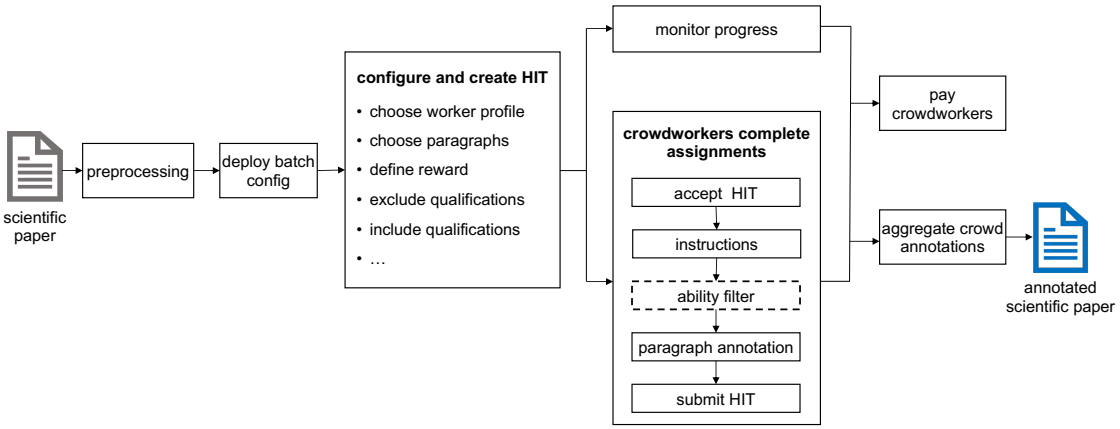


Figure 6.2: Final workflow design

task. Then, a worker subsequently annotates, for example, three different paragraphs. This means that relations can only be identified if both of the related components lie exactly within a paragraph. We analysed how many of the relations found by Lauscher et al. (2018b) are not identifiable anymore due to the preprocessing. In the six paragraphs we used for the pilots, there are no unidentifiable relations. In all 42 paragraphs from the paper, there are 15 unidentifiable relations.

Thanks to the final workflow we designed, many things are automated to reduce the human computation effort as much as possible. However, there is a need to be able to configure adjust and monitor various steps of this workflow. Therefore, we designed our workflow in a way that the user of the system can follow the workflow with just a few necessary interventions (such as creating the hit or paying the workers) while at the same time giving the user the possibility to configure as many things as possible. This optimally interweaves machine and human computation and also enables the user to run a wide range of differently configured HITs and batches of HIT.

(RQd) Structured Annotations: What vocabularies should be used/extended to annotate argumentative components and relations?

We chose the argumentation scheme according to Lauscher et al's (2018b) work. They elaborated their scheme with two expert annotators, starting from the Toulmin (2003) model. Based on this scheme, they then created detailed annotation guidelines (Lauscher et al., 2018c). We formulated the task instructions for both the argument component annotation task and the argumentative relation annotation task based on these guidelines. In particular, we explained the different types of components and relations and also provided various examples of all of them. Further, as can be seen in Appendices B.3 and B.4, we provided step-by-step instructions on how the tasks should be solved. As 2018c) created their guidelines iteratively and by discussing details and ambiguities with

(expert) annotators, there was neither the need to extend them nor to change some vocabulary. In contrary, we summarised them to only provide the information that was crucial for the crowdworkers to be able to solve the task. Thereby, we did not want to overload the workers with information and we wanted them to be able to go through the instructions as quickly as possible.

Based on the survey the workers answered during the pilots, we found that it was not always 100% clear to every worker what the task was about (for both the argument component and the argumentative relation annotation tasks). However, this was to be expected because it is not every day that crowdworkers annotate arguments in difficult-to-understand scientific papers from a domain in which most are not experts. In addition to that, as we broke down Lauscher et al.'s (2018b) 15 pages of extensive annotation guidelines into concise instructions which could be read in just a few minutes, it was not our goal to provide a detailed example for every potential case and exception that might appear when annotating arguments in a scientific paper because this is not feasible and contradicts general crowdsourcing best practises (Sheehan and Pittman, 2016; Alonso, 2019). However, in the survey, only one out of 16 workers reported that the task instructions were very unclear. For this reason, we are confident that the summaries of the annotation guidelines we presented to the crowd (for both HIT types) contain enough information for workers to understand what the task is about.

(RQe) Aggregation Method: How can we combine the results from the crowd and save them so they are reusable in the future?

It is important to have multiple annotators to create an argument-annotated corpus. First of all, because ideas can be expressed in many different ways and arguments may be ambiguous, when annotating natural language text, relying on just one annotator leads to a biased result (Wacholder et al., 2014). Second, there is a need for at least three annotators in order to break ties when two solutions are annotated the same number of times (Wacholder et al., 2014; Dumitrache et al., 2018).

As described for (RQb), we use MACE (Hovy et al., 2013) to aggregate the crowdsourced annotations on a token level. We use the output of MACE's aggregation to create the final output. Our system's final output is a text file which contains the argumentative components and relations for a scientific paper (or the part of it) that was used as the input to the crowd-powered system. For the final output, we apply the same format as Lauscher et al. (2018b), which we describe in detail in Section 3.2.1. We chose this format as it is easily readable for both humans and machines, which makes it reusable in the future. For humans, it is possible to quickly get an idea of what has been annotated as for every annotated argument component, it also includes the actual content of the annotation. For machines, it is readable as every argumentative component or relation is written on a separate line and also because it includes the exact character indices of the annotated argument components, which make it possible to know which exact part of the entire paper has been annotated. Just using the textual content of the annotation would not be enough as there might be duplicate text spans in a paper.

(RQ) How can we design a crowd-powered system to annotate scientific publications for argument mining effectively?

We answer our initial research question by summarising and combining the main answers to the subquestions.

The findings of (RQa) show that letting the crowd annotate a scientific paragraph without any answer restrictions allows us to use the crowd's knowledge to its fullest. However, allowing the crowd to freely annotate natural language text, instead of just asking single-choice questions, also results in answers of varying quality. We strive to increase quality by providing clear instructions and a well-designed annotation tool; however, we do not want to use the task design to enforce crowdworkers to behave in some way — for example, spending a minimum amount of time on a paragraph or doing a minimum amount of annotations before continuing to the next paragraph. We believe that those workers who tend to maximise profit by spending as little time as possible per question would always find a way to trick these design-imposed restrictions. Instead, we focused on other approaches to assure data quality, as discussed for (RQb): profile requirements, a filter mechanism and answer aggregation. We found that just combining worker profile requirements with good aggregation methods does not lead to accurately annotated scientific papers. There is a need to filter out workers who would not go on to produce high-quality annotations. The *ability filter* we implemented has the potential to detect high-performing workers. In addition to that, our final system does not only detect those workers but also detains workers who are expected to perform poorly from participating. Moreover, it allows the requester to adjust the filter with a configurable threshold as a means to steer the accuracy of the obtained crowd annotations.

In this work, we introduced an end-to-end process to create gold standard datasets for AM. To answer question (RQc), we designed a workflow, which we continuously optimised and extended. The first idea was always to automate as much as possible, to keep the manual effort as low as possible. However, we quickly realised that, as a requester, having more configuration options means having more flexibility to decide how exactly a HIT is set up. Therefore, we extended specific configuration option on batch, as well as on a HIT level. A requester can, however, set up the configuration once, according to the situation's requirements, and then run as many HITs as needed. At the same time, we give workers the guidance they need and the freedom deserve to emphasise their skills to the fullest. For this reason, we believe that our workflow optimally interweaves machine and human computation.

Regarding (RQd), we found that a summary of Lauscher et al.'s (2018c) detailed annotation guidelines is enough to instruct the crowd as they include important definitions and examples with regard to the underlying annotation scheme. An extension or an adjustment of the vocabulary used is not necessary.

For (RQe) we showed that after aggregating the annotations provided by the crowd, the results can be brought to a machine-readable format which is reusable in the future. The usage of MACE (Hovy et al., 2013) enables us to aggregate crowd annotations to a final solution which is likely to be correct, as the estimator mechanisms in place consider the trustworthiness of the annotators and also of the different annotations.

To conclude and give an answer to the initial research question, there are several things a crowd-powered system should include so that scientific publications for argument mining can be annotated effectively. The system should follow a workflow that optimally interweaves machine and human computation, by letting the machine do as much of the work as possible while at the same time giving the human user the possibility to configure and monitor the system at any time. In addition to that, the designed crowd tasks must include a quality assurance mechanism which combines crowdworker profile selection, a filter mechanism to detect high-performing workers and a method to aggregate the crowd's answers by estimating the trustworthiness of workers and annotations.

7

Limitations and Future Work

In this chapter, we acknowledge the limitations of our work and discuss how they can be addressed in future work. In addition to that, we describe potentially interesting follow-up studies to further validate and improve our system.

Annotating argumentative components and relations is a complex task. Considering the limited funds available, we had to select just six short paragraphs (three of which were only included in the pilot studies) from one scientific paper upon which to run our experiments. As described in Section 3.3.4, we tried to select representative paragraphs from a paper of average difficulty. But still, we cannot exclude the possibility that the characteristics of the chosen paragraphs biased the result. In addition to that, the paper we selected is from the domain of computer graphics, which is why our findings are not generalisable for all scientific papers. Authors of scientific papers in other research areas might have a different style of presenting their work and, with that, also different argumentative formulations.

The fact that we ran experiments with only a limited number of crowdworkers, due to time and budget constraints, potentially limits the generalisability of our findings. As there are many active workers on AMT at any one time, it might, at least partially, be based on chance which workers are the first to accept a HIT and hence who ultimately gets to work on a task. Our system includes human computation, therefore, the effect of the number of attempts needed to complete the *ability filter* on annotation performance is not able to exactly predict the final annotation quality but is rather meant to be a suggestion for steering the preferred quality. When deciding on the *ability filter* threshold, the user faces a quality-costs-time trade-off, because it is not possible to achieve high-quality crowd annotations with low costs in a short amount of time.

We chose to split the annotation into two separate tasks, one for the detection of argument components and one for the detection of argumentative relations. To annotate argumentative relations that hold between argument components in a scientific paper, these components have to be known. In our experiments, we decided not to use the crowdsourced argument component annotations as an input for the annotation of argumentative relations because using the crowd annotated components would have introduced bias. Instead, we used the same argument components from the corpus of Lauscher et al. (2018b), which we also used as the ground truth we held our results against. Our system is built to allow for crowd annotated components as an input for

the annotation of argumentative relations, however, more extensive empirical evaluations are needed to find out whether the crowd performs better when these two tasks are solved separately or when every worker has to solve them subsequently.

The choice of a single specific argumentation scheme represents another limitation. Due to time constraints (creating and refining an argumentation scheme is a complex task) budget constraints (running experiments to validate the argumentation scheme is costly because the crowd has to be paid for participation), we did not evaluate other argumentation schemes.

Last but not least, we build it specifically for seamless interaction between the admin area and AMT and for the direct embedding of the annotation tool in AMT. Not only the choice of AMT as the crowdsourcing platform but also the functionality offered by the AMT API impose restrictions on the usage of our system in the future.

Based on the limitations described above, various interesting possibilities for future research emerge. Applying our system to entire scientific papers will be necessary to evaluate whether the crowd performs better for a specific part of a paper, such as the introduction. Further, it should also be applied to scientific papers from other domains to find out whether the task instructions we created are clear enough for scientific papers in general or if there is a need for domain-specific explanations and examples (or even for an adjustment of the argumentation scheme). Moreover, additional experiments with more workers annotating the same parts of a scientific paper are necessary to assess whether our approach of letting three workers annotate the same paragraph is optimal regarding the accuracy-cost trade-off — more annotators increase the accuracy (Stab et al., 2018), but also increase the costs.

In a final scenario, where the crowd should annotate argumentative components and relations from scratch, the identified components serve as an input for the argumentative relation annotation task. Thereby, it will be interesting to evaluate whether the crowd performs better when these two tasks are solved separately (first the component annotation task and than, based on the aggregated annotations, the relation annotation task) or when every worker has to solve a combination of both tasks. Our final system is implemented specifically for this case; however, the results we achieved need to be validated, as we used the ground truth argument components as an input for the argumentative relation.

The *ability filter* we implemented can easily be extended to serve a training functionality. For that to be the case, the examples present in the *ability filter* need to be extended and detailed explanations for the solutions must be provided. Additional studies can investigate whether training workers who would not pass the filter otherwise, could enlarge the number of high-ability crowdworkers and even increase the final annotation accuracy. In Chapter 6, we suggested *ability filter* thresholds for both the argument component as well as the argumentative relation annotation task. Low thresholds decrease the number of admissible crowdworkers for the annotation task at hand — as described in Chapter 6, a low threshold corresponds to only accepting workers who did not use too many attempts and is associated with higher quality, on average. The impact these low thresholds have on total task completion time is not clear and should

therefore be examined carefully.

Applying these thresholds allows the selection of crowdworkers who are likely to perform well at argument annotation tasks. This enables follow-up studies, for example, an investigation of the impact of a reliable workforce consisting of just a few workers, which was created by applying low thresholds, on dataset quality. In this regard, workers who have proven to produce high-quality argument annotations could be given the chance to annotate more than just three short paragraphs. Especially for larger annotation projects, selecting the best workers to annotate the most parts might be beneficial in terms of accuracy (if workers become better with experience) as well as efficiency (if workers become faster with experience). Thereby, in case of satisfactory results, AM researchers could benefit by being able to find a few good annotators in a large crowd of potential workers.

Even though we selected a single specific argumentation scheme, our system can easily be extended to evaluate other types of argumentative component and relations, based on a different argumentation scheme. In addition to that, our system can also be used to annotate other entities in scientific publications, such as hypotheses, or even entities in natural language text from a different source, for example, news articles.

8

Conclusions

Argument Mining (AM) aims at the automatic extraction of argumentative components and the relations that hold between them in natural language text. As an emerging research area, one of the major challenges in AM research is the lack of annotated datasets (Peldszus and Stede, 2013; Lawrence and Reed, 2019). State-of-the-art AM approaches make use of Machine Learning (ML) methods, and they depend on the availability of these datasets to be used as training data. In addition to that, argument-annotated corpora are crucial for benchmark experiments.

As the number of scientific publications keeps increasing (Bornmann and Mutz, 2015), it becomes more and more difficult to trawl through all of the relevant scientific articles for a specific topic. For this reason, the mining of arguments within scientific papers is gaining more and more attention in the field of AM (Stab et al., 2014; Green et al., 2014b; Green, 2015; Kirschner et al., 2015; Lauscher et al., 2018b,a; Accuosto and Saggion, 2019; Song et al., 2019). Even though an annotated dataset of scientific papers was released in a machine-readable format by Lauscher et al. (2018b), we still lack a system to generate further ground truth datasets.

In this work, we took a novel approach by implementing a system that could be used to create gold standard datasets for AM with the help of hundreds of thousands of ordinary workers (i.e. the crowd). Accordingly, the goal was to find out how a crowd-powered system to accurately annotate scientific publications for argument mining can be designed. We broke this question down into various subtopics: task design, data quality assurance method, workflow definition, structured annotations and aggregation method. We quickly realised that due to the relatively low average performances achieved by the crowd, the data quality assurance method would be a crucial part of the system. Therefore, we stated two hypotheses, which we empirically evaluate in the form of experiments on Amazon Mechanical Turk (AMT): The first hypothesis, which has not been supported by our results, aimed at checking, with one basic question per annotation type, whether the annotators understand the difference between the argument component types (argumentative relation types respectively) to filter out spammers. The second hypothesis, which has been supported by our results, aimed at checking, with a few specific annotation tasks of varying difficulty, whether the annotators are able to accurately annotate the different argument component types (argumentative relation types respectively), to filter out workers who do not have the ability to provide high-quality argument annotations in scientific papers. As the results from the experiments we ran supported

the second hypothesis, we then included this quality assurance mechanism in our final system.

To answer our initial question, we developed an end-to-end process that can be used to create gold standard datasets for AM, based on which we implemented a system in the form of a Python-based Flask[1] WebApp. To enable the crowd-powered annotation of argumentative components and relations in natural language text, our system traverses the following stages: With a plain text file as an input, the system automatically preprocesses the data and transforms it so that the text can be annotated by the crowd. Then, the user configures the Human Intelligence Tasks (HIT) task by defining which part should be annotated by how many workers and for which reward. In addition to that, the preferred worker profile can be defined. After the HIT has been created, it appears on AMT and workers can complete the HIT using the annotation tool, which is part of the Flask WebApp, as it is directly embedded in AMT in the form of an inline frame. Workers who choose to work on the HIT have to complete a few specific annotation tasks of varying difficulty, based on which the system evaluates whether they should be allowed to continue to annotating the paragraphs. This mechanism filters out bad-performing workers at an early stage, which makes the overall process more efficient and also assures high-quality results. Finally, the system automatically aggregates the crowd's annotations and saves the annotated dataset to a human- and machine-readable format.

We ran experiments on AMT with a total of 70 participants. Thereby, we evaluated our system and measured the performance of the crowd by comparing it against the dataset released by Lauscher et al. (2018b). In an iterative fashion, we continuously developed our tool further by immediately implementing any gained insights. Our results show that annotating argumentative components and relations in difficult-to-understand scientific papers is a complex task which requires a lot of time, especially for non-experts. Further, our experiments showed that a profile-based selection of crowd annotators in combination with a good aggregation mechanism is necessary but not sufficient to assure high annotation quality. We therefore designed and empirically evaluated the introduction of a quality assurance filter mechanism. We found that with the usage of this mechanism, it is possible to detect high-performing workers and also to detain workers who are expected to perform poorly from participating. In this way, it is possible, to some extent, to steer the quality of the crowd-annotated dataset, in exchange for money and time — money, because workers need to complete the task that will determine whether they will be filtered out or not, and time, because filtering out workers results in a smaller workforce, meaning it could take longer for all annotation tasks to be completed by the crowd. Accordingly, it is at the requester's discretion to weight the quality-costs-time trade-off in the given situation for the text at hand.

Even though crowdsourcing has been suggested for annotating datasets (Pustejovsky and Stubbs, 2012), we are the first to use it as a means to curate ground truth datasets of scientific papers for AM from scratch. By involving the crowd, we enable a multi-

[1] *https://flask.palletsprojects.com.*

tude of people to help with the collection and annotation of natural language text data critical to AM research. The system we implemented contributes to the AM community as it provides a novel approach to generating gold standard datasets. As argumentative structures in scientific papers are often ambiguous, the crowd, consisting of non-expert annotators, is not able to generate a perfectly annotated AM dataset. We believe that the help of the crowd will be an important factor for AM research in the future. However, whether or not our system can serve as an alternative to experts for argument annotations in scientific papers depends on the quality requirements.

References

Accuosto, P. and Saggion, H. (2019). Transferring knowledge from discourse to arguments: A case study with scientific abstracts. In *Stein B, Wachsmuth H, editors. Proceedings of the 6th Workshop on Argument Mining; 2019 Aug 1; Florence, Italy. Stroudsburg: Association for Computational Linguistics; 2019. p. 41-51.* ACL (Association for Computational Linguistics).

Alonso, O. (2019). *The practice of crowdsourcing.* Morgan & Claypool Publishers.

Amazon Mechanical Turk (2015, Dec 17). *Simplified Masters Qualifications.* Retrieved Jun 23, 2020 from https://blog.mturk.com/simplified-masters-qualifications-137d77647d1c.

Amazon Mechanical Turk (2017, Jan 26). *Tutorial: Best practices for managing Workers in follow-up surveys or longitudinal studies.* Retrieved Jul 23, 2020 from https://blog.mturk.com/tutorial-best-practices-for-managing-workers-in-follow-up-surveys-or-longitudinal-studies-4d0732a7319b.

Amazon Mechanical Turk (2018). *FAQs.* Retrieved Jun 23, 2020 from https://www.mturk.com/worker/help.

Amazon Mechanical Turk (2020, Mar 10). *Paying for Non-Submitted HITs.* Retrieved on Sep 15, 2020 from https://blog.mturk.com/paying-for-non-submitted-hits-245c6c3323bb.

Antoine, J.-Y., Villaneau, J., and Lefeuvre, A. (2014). Weighted Krippendorff's alpha is a more reliable metrics for multi-coders ordinal annotations: experimental studies on emotion, opinion and coreference annotation.

Artstein, R. (2017). Inter-annotator agreement. In *Handbook of linguistic annotation*, pages 297–313. Springer.

Baker, G. P. and Huntington, H. B. (1925). *The principles of argumentation.* Ginn, 2nd edition.

Bench-Capon, T. J. M. and Dunne, P. E. (2007). Argumentation in artificial intelligence. *Artificial intelligence*, 171(10-15):619–641.

Bentahar, J., Moulin, B., and Bélanger, M. (2010). A taxonomy of argumentation models used for knowledge representation. *Artificial Intelligence Review*, 33(3):211–259.

Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Blake, C. (2010). Beyond genes, proteins, and abstracts: Identifying scientific claims from full-text biomedical articles. *Journal of biomedical informatics*, 43(2):173–189.

Bornmann, L. and Mutz, R. (2015). Growth rates of modern science: A bibliometric analysis based on the number of publications and cited references. *Journal of the Association for Information Science and Technology*, 66(11):2215–2222.

Cabrio, E. and Villata, S. (2018). Five years of argument mining: A Data-driven Analysis. In *IJCAI International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Damer, T. E. (2012). *Attacking faulty reasoning.* Cengage Learning.

Demartini, G., Difallah, D. E., Gadiraju, U., Catasta, M., and Delft, B. (2017). An Introduction to Hybrid Human-Machine Information Systems. *Foundations and Trends in Web Science*, 7(1):1–87.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Difallah, D., Filatova, E., and Ipeirotis, P. (2018). Demographics and dynamics of Mechanical Turk workers. *WSDM 2018 - Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, 2018-Febua(August 2017):135–143.

Dumitrache, A., Inel, O., Timmermans, B., Ortiz, C., Sips, R.-J., Aroyo, L., and Welty, C. (2018). Empirical methodology for crowdsourcing ground truth. *arXiv preprint arXiv:1809.08888*.

Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence*, 77(2):321–357.

Dusmanu, M., Cabrio, E., and Villata, S. (2017). Argument mining on Twitter: Arguments, facts and sources. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2317–2322.

Duthie, R., Budzynska, K., and Reed, C. (2016a). Mining Ethos in Political Debate. In *COMMA*, pages 299–310.

Duthie, R., Lawrence, J., Budzynska, K., and Reed, C. (2016b). The CASS Technique for Evaluating the Performance of Argument Mining. In *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*, pages 40–49.

Fisas, B., Ronzano, F., and Saggion, H. (2016). A multi-layered annotated corpus of scientific papers. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3081–3088.

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

García-Villalba, M. P. and Saint-Dizier, P. (2012). A framework to extract arguments in opinion texts. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 6(3):62–87.

Ghosh, D., Muresan, S., Wacholder, N., Aakhus, M., and Mitsui, M. (2014). Analyzing argumentative discourse units in online interactions. In *Proceedings of the first workshop on argumentation mining*, pages 39–48.

Green, N. (2014). Towards creation of a corpus for argumentation mining the biomedical genetics research literature. In *Proceedings of the first workshop on argumentation mining*, pages 11–18.

Green, N. (2016). Implementing Argumentation Schemes as Logic Programs. In *CMNA@ IJCAI*, pages 1–7.

Green, N., Ashley, K. D., Litman, D., Reed, C., and Walker, V. (2014a). Proceedings of the First Workshop on Argumentation Mining. In *Proceedings of the First Workshop on Argumentation Mining*.

Green, N., Cabrio, E., Villata, S., and Wyner, A. (2014b). Argumentation for Scientific Claims in a Biomedical Research Article. In *ArgNLP*, pages 21–25.

Green, N. L. (2015). Annotating evidence-based argumentation in biomedical text. In *2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 922–929. IEEE.

Green, N. L., Branon, M., and Roosje, L. (2019). Argument schemes and visualization software for critical thinking about international politics. *Argument & Computation*, 10(1):41–53.

Habernal, I., Eckle-Kohler, J., and Gurevych, I. (2014). Argumentation Mining on the Web from Information Seeking Perspective. In *ArgNLP*.

Habernal, I. and Gurevych, I. (2017). Argumentation mining in user-generated web discourse. *Computational Linguistics*, 43(1):125–179.

Hovy, D., Berg-Kirkpatrick, T., Vaswani, A., and Hovy, E. (2013). Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130.

Howe, J. (2006). The rise of crowdsourcing. *Wired magazine*, 14(6):1–4.

Howe, J. (2009,). *Crowdsourcing: A Definition*. Retrieved Jul 5, 2020 from http://crowdsourcing.typepad.com.

Hung, N. Q. V., Tam, N. T., Tran, L. N., and Aberer, K. (2013). An evaluation of aggregation techniques in crowdsourcing. In *International Conference on Web Information Systems Engineering*, pages 1–15. Springer.

Janier, M., Reed, C., and Lawrence, J. (2014). OVA+: An argument analysis interface. In *Computational Models of Argument: Proceedings of COMMA*, volume 266, pages 463–464.

Ketcham, V. A. (1914). *The theory and practice of argumentation and debate*. Macmillan.

Kirschner, C., Eckle-Kohler, J., and Gurevych, I. (2015). Linking the thoughts: Analysis of argumentation structures in scientific publications. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 1–11.

Kriplean, T., Morgan, J., Freelon, D., Borning, A., and Bennett, L. (2012). Supporting Reflective Public Thought with Considerit. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 265–274, New York, NY, USA. Association for Computing Machinery.

Krippendorff, K. (2018). *Content analysis: An introduction to its methodology*. Sage publications.

Lauscher, A., Glavaš, G., and Eckert, K. (2018a). Arguminsci: A tool for analyzing argumentation and rhetorical aspects in scientific writing. In *Proceedings of the 5th Workshop on Argument Mining*, pages 22–28. Association for Computational Linguistics.

Lauscher, A., Glavaš, G., and Ponzetto, S. P. (2018b). An Argument-Annotated Corpus of Scientific Publications. In *Proceedings of the 5th Workshop on Argument Mining*, pages 40–46.

Lauscher, A., Glavas, G., Ponzetto, S. P., and Eckert, K. (2018c). Annotating Arguments in Scientific Publications. Technical report.

Lavee, T., Kotlerman, L., Orbach, M., Bilu, Y., Jacovi, M., Aharonov, R., and Slonim, N. (2019). Crowd-sourcing annotation of complex NLU tasks: A case study of argumentative content annotation. In *Proceedings of the First Workshop on Aggregating and Analysing Crowdsourced Annotations for NLP*, pages 29–38.

Law, E. (2011). Defining (Human) Computation. *Control*.

Lawrence, J. and Reed, C. (2019). Argument mining: A survey. *Computational Linguistics*, 45(4):765–818.

Li, G., Wang, J., Zheng, Y., and Franklin, M. J. (2016). Crowdsourced data management: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 28(9):2296–2319.

Lippi, M. and Torroni, P. (2016a). Argument mining from speech: Detecting claims in political debates. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Lippi, M. and Torroni, P. (2016b). Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology*, 16(2).

Lippi, M. and Torroni, P. (2016c). MARGOT: A web server for argumentation mining. *Expert Systems with Applications*, 65:292–303.

Loepp, E. and Kelly, J. T. (2020). Distinction without a difference? An assessment of MTurk Worker types. *Research & Politics*, 7(1):2053168019901185.

MacEwan, E. J. (1898). *The essentials of argumentation*. DC Heath & Company.

Marcus, A. and Parameswaran, A. (2013). Crowdsourced data management: Industry and academic perspectives. *Foundations and Trends in Databases*, 6(1-2):1–161.

Miller, T., Sukhareva, M., and Gurevych, I. (2019). A streamlined method for sourcing discourse-level argumentation annotations from the crowd. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1790–1796.

Mochales, R. and Ieven, A. (2009). Creating an argumentation corpus: do theories apply to real arguments? A case study on the legal argumentation of the ECHR. In *Proceedings of the 12th international conference on artificial intelligence and law*, pages 21–30.

Mochales, R. and Moens, M. F. (2011). Argumentation mining. *Artificial Intelligence and Law*, 19(1):1–22.

Moens, M. F. (2018). Argumentation mining: How can a machine acquire common sense and world knowledge? *Argument and Computation*, 9(1):1–4.

Nguyen, Q. V. H., Duong, C. T., Nguyen, T. T., Weidlich, M., Aberer, K., Yin, H., and Zhou, X. (2017). Argument discovery via crowdsourcing. *VLDB Journal*, 26(4):511–535.

Nowak, S. and Rüger, S. (2010). How reliable are annotations via crowdsourcing? A study about inter-annotator agreement for multi-label image annotation. In *MIR 2010 - Proceedings of the 2010 ACM SIGMM International Conference on Multimedia Information Retrieval*, pages 557–566.

O'Keefe, D. J. (2012). Conviction, persuasion, and argumentation: Untangling the ends and means of influence. *Argumentation*, 26(1):19–32.

Ong, N., Litman, D., and Brusilovsky, A. (2014). Ontology-Based Argument Mining and Automatic Essay Scoring. In *Proceedings of the First Workshop on Argumentation Mining*, pages 24–28, Baltimore, Maryland. Association for Computational Linguistics.

Park, J. and Cardie, C. (2018). A corpus of erulemaking user comments for measuring evaluability of arguments. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Peer, E., Vosgerau, J., and Acquisti, A. (2014). Reputation as a sufficient condition for data quality on Amazon Mechanical Turk. *Behavior research methods*, 46(4):1023–1031.

Peldszus, A. and Stede, M. (2013). From argument diagrams to argumentation mining in texts: A survey. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 7(1):1–31.

Petasis, G. (2014). Annotating Arguments: The NOMAD Collaborative Annotation Tool. In *LREC*, pages 1930–1937.

Posten, H. O. (1984). Robustness of the two-sample t-test. In *Robustness of statistical methods and nonparametric statistics*, pages 92–99. Springer.

Powers, D. M. W. (2015). What the F-measure doesn't measure: Features, Flaws, Fallacies and Fixes. *arXiv preprint arXiv:1503.06410*.

Pustejovsky, J. and Stubbs, A. (2012). *Natural Language Annotation for Machine Learning: A guide to corpus-building for applications.* " O'Reilly Media, Inc.".

Quinn, A. J. and Bederson, B. B. (2011). Human computation: A survey and taxonomy of a growing field. *Conference on Human Factors in Computing Systems - Proceedings*, pages 1403–1412.

Rocha, G. and Cardoso, H. L. (2017). Towards a relation-based argument extraction model for argumentation mining. In *International Conference on Statistical Language and Speech Processing*, pages 94–105. Springer.

Rosenfeld, A. and Kraus, S. (2016). Providing Arguments in Discussions on the Basis of the Prediction of Human Argumentative Behavior. *ACM Trans. Interact. Intell. Syst.*, 6(4).

Sá, D. (2019). Argument Diagramming: Annotation and Evaluation.

Schneider, J. (2014). Automated argumentation mining to the rescue? Envisioning argumentation and decision-making support for debates in open online collaboration

communities. In *Proceedings of the First Workshop on Argumentation Mining*, pages 59–63, Baltimore, Maryland. Association for Computational Linguistics.

Sheehan, K. B. and Pittman, M. (2016). *Amazon's Mechanical Turk for academics: The HIT handbook for social science research.* Melvin & Leigh, Publishers.

Skeppstedt, M., Peldszus, A., and Stede, M. (2018). More or less controlled elicitation of argumentative text: Enlarging a microtext corpus via crowdsourcing. In *Proceedings of the 5th Workshop on Argument Mining*, pages 155–163.

Song, N., Cheng, H., Zhou, H., and Wang, X. (2019). Argument Structure Mining in Scientific Articles: A Comparative Analysis. In *2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 339–340.

Stab, C. and Gurevych, I. (2014). Annotating argument components and relations in persuasive essays. In *Proceedings of COLING 2014, the 25th international conference on computational linguistics: Technical papers*, pages 1501–1510.

Stab, C. and Gurevych, I. (2017a). Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659.

Stab, C. and Gurevych, I. (2017b). Recognizing insufficiently supported arguments in argumentative essays. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 980–990.

Stab, C., Kirschner, C., Eckle-Kohler, J., and Gurevych, I. (2014). Argumentation Mining in Persuasive Essays and Scientific Articles from the Discourse Structure Perspective. In *ArgNLP*, pages 21–25.

Stab, C., Miller, T., and Gurevych, I. (2018). Cross-topic argument mining from heterogeneous sources using attention-based neural networks. *arXiv preprint arXiv:1802.05758*.

Stede, M. and Schneider, J. (2018). Argumentation mining. *Synthesis Lectures on Human Language Technologies*, 11(2):1–191.

Stenetorp, P., Pyysalo, S., Topíc, G., Ohta, T., Ananiadou, S., and Tsujii, J. (2012). BRAT: AWeb-based tool for NLP-Assisted text annotation. In *EACL 2012 - Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107. Association for Computational Linguistics (ACL).

Teruel, M., Cardellino, C., Cardellino, F., Alemany, L. A., and Villata, S. (2018a). Guidelines for Annotating Argumentation Structures in Legal Cases. Technical report, Universidad Nacional de Córdoba, Argentina.

Teruel, M., Cardellino, C., Cardellino, F., Alemany, L. A., and Villata, S. (2018b). Increasing argument annotation reproducibility by using inter-annotator agreement to improve guidelines. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Teufel, S., Carletta, J., and Moens, M. (1999). An annotation scheme for discourse-level argumentation in research articles. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*.

Teufel, S. and Moens, M. (1999). Discourse-level argumentation in scientific articles: human and automatic annotation. In *Towards Standards and Tools for Discourse Tagging*.

Teufel, S. and Moens, M. (2002). Summarizing scientific articles: experiments with relevance and rhetorical status. *Computational linguistics*, 28(4):409–445.

Teufel, S., Siddharthan, A., and Batchelor, C. (2009). Towards discipline-independent argumentative zoning: evidence from chemistry and computational linguistics. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, volume 3, pages 1493–1502.

Toulmin, S. (2003). *The uses of argument*. Cambridge university press.

van Eemeren, F. H., Garssen, B., Krabbe, E. C. W., Verheij, B., and Wagemans, J. H. M. (2014). Handbook of argumentation theory.

Van Rijsbergen, C. J. (1979). *Information Retrieval*. Butterworth-Heinemann, 2 edition.

Wacholder, N., Muresan, S., Ghosh, D., and Aakhus, M. (2014). Annotating multiparty discourse: Challenges for agreement metrics. In *Proceedings of LAW VIII-The 8th Linguistic Annotation Workshop*, pages 120–128.

Walker, V. R., Han, J. H., Ni, X., and Yoseda, K. (2017). Semantic types for computational legal reasoning: propositional connectives and sentence roles in the veterans' claims dataset. In *Proceedings of the 16th edition of the International Conference on Articial Intelligence and Law*, pages 217–226.

Walton, D. (1998). *The new dialectic: Conversational contexts of argument*. University of Toronto Press.

Walton, D. (2015). *Goal-based Reasoning for Argumentation*. Cambridge University Press.

Walton, D., Reed, C., and Macagno, F. (2008). *Argumentation schemes*. Cambridge University Press.

Welch, B. L. (1947). The generalization ofstudent's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35.

Wells, S. (2014). Argument Mining: Was Ist Das? *Computational Models of Natural Argument*.

Young, J. A. and Young, K. M. (2019). Don't Get Lost in the Crowd: Best Practices for Using Amazon's Mechanical Turk in Behavioral Research. *Journal of the Midwest Association for Information Systems*, 2019(2):7.

Zhao, X., Liu, J. S., and Deng, K. (2013). Assumptions behind intercoder reliability indices. *Annals of the International Communication Association*, 36(1):419–480.

A

Framework Usage

A.1 Running Flask App Locally

Run the following commands to get started running this app locally:

```
$ git clone https://gitlab.ifi.uzh.ch/ddis/Students/Theses/
  2020-joachim-baumann.git

$ cd 2020-joachim-baumann

$ python3 -m venv venv

$ source venv/bin/activate

$ pip install -r requirements.txt

$ FLASK_APP=app FLASK_ENV=development

$ flask run
```

Now that the Flask app is running locally, you can visit the following pages:

- Admin area: *http://localhost:5000/admin_area*
- Argument component annotation HIT template (various versions for all experiments):
  - Final version: *http://localhost:5000/textannotation/arguments_final/batch0_argumentComponents_final/paragraph_2;paragraph_3;paragraph_4?assignmentId=ComponentsFinal1&workerId=workercomponent0*
  - Final version (as a worker who has already passed the *ability filter* previously): *http://localhost:5000/textannotation/arguments_final/batch0_argumentComponents_final/paragraph_2;paragraph_3;paragraph_4?assignmentId=ComponentsFinal2&workerId=workercomponent1-passedfilter*
  - Version used for experiment H1b: *http://localhost:5000/textannotation/arguments_filterStep2/batch3_argumentComponents_filterStep2/paragraph_2;paragraph_3;paragraph_4?assignmentId=ComponentsH1b*

  - Version used for experiment H1a: *http://localhost:5000/textannotation/arguments_filterStep1/batch2_argumentComponents_filterStep1/paragraph_2;paragraph_3;paragraph_4?assignmentId=ComponentsH1a*
  - Version used for experiment pilot 2 (paragraphs 16-18): *http://localhost:5000/textannotation/arguments/batch1_argumentComponents/paragraph_16;paragraph_17;paragraph_18?assignmentId=ComponentsPilot2*
  - Version used for experiment pilot 1 (paragraphs 2-4): *http://localhost:5000/textannotation/arguments/batch1_argumentComponents/paragraph_2;paragraph_3;paragraph_4?assignmentId=ComponentsPilot1*

- Argumentative relation annotation HIT template (various versions for all experiments):
  - Final version: *http://localhost:5000/textannotation/relations_final/batch0_argumentRelations_final/paragraph_2;paragraph_3;paragraph_4?assignmentId=TestRelations_final&workerId=workerrelation1*
  - Final version (as a worker who has already passed the *ability filter* previously): *http://localhost:5000/textannotation/relations_final/batch0_argumentRelations_final/paragraph_2;paragraph_3;paragraph_4?assignmentId=TestRelations_final&workerrelation1-passedfilter*
  - Version used for experiment H1b: *http://localhost:5000/textannotation/relations_filterStep2/batch3_argumentRelations_filterStep2/paragraph_2;paragraph_3;paragraph_4?assignmentId=RelationsH1b*
  - Version used for experiment H1a: *http://localhost:5000/textannotation/relations_filterStep1/batch2_argumentRelations_filterStep1/paragraph_2;paragraph_3;paragraph_4?assignmentId=RelationsH1a*
  - Version used for experiment pilot 2 (paragraphs 16-18): *http://localhost:5000/textannotation/relations/batch1_argumentRelations_16_18/paragraph_16;paragraph_17;paragraph_18?assignmentId=RelationsPilot2P1618*
  - Version used for experiment pilot 2 (paragraphs 2-4): *http://localhost:5000/textannotation/relations/batch1_argumentRelations/paragraph_2;paragraph_3;paragraph_4?assignmentId=RelationsPilot2P24*

All HIT templates outlined above can also be accessed online by replacing `http://localhost:5000` with `https://masterthesisjb.herokuapp.com` or with the URL to your own Heroku app once the deployment, which is described next, is complete.

A.2 Deployment to Heroku

Make sure Heroku is installed[1].

Then, run the following commands to deploy the Flask app to Heroku:

```
$ git clone https://gitlab.ifi.uzh.ch/ddis/Students/Theses/
  2020-joachim-baumann.git

$ cd 2020-joachim-baumann

$ heroku create

$ git push heroku master
```

After the deployment is complete you can open the flask app using the `heroku open` command. This brings you to the welcome page. With the URLs outlined above, you can then access either the admin area or one of the various HIT templates. Further, you can see your newly created Heroku app in the Heroku dashboard[2] from where you can manage the app (for example change the app name).

Important additional information:

- Make sure that the name of the Heroku app corresponds with the `APP_NAME` variable in the config file[3].
- Make sure to always push the repository to Heroku (using `git push heroku master`) after a new batch configuration has been added to the config file).
- Ass the aggregation of worker answers is done with Multi-Annotator Competence Estimation (MACE)[4], make sure to download it and place in the root of the repository.
- Used Python version: 3.8.2

A.3 Content of the Code Repository

Here, we describe the most important files contained in the of the digital submission of the code repository. Files in green color are crucial for the final system and files in blue color were needed for the experiments but not for the final system.

- AnswerAggregationAndPerformanceEvaluation:
  - H1a_2-sample-t-test.R: R script to evaluate hypothesis H1a for both task types

[1] *https://devcenter.heroku.com/articles/heroku-cli*
[2] *https://dashboard.heroku.com/apps*
[3] *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann/-/blob/master/config.py*
[4] MACE is a Java-based implementation which is available for download at: *https://www.isi.edu/publications/licensed-sw/mace*.

  - H1a_scores_components.csv: Datapoints for hypothesis H1a (argument component annotation task)
  - H1a_scores_relations.csv: Datapoints for hypothesis H1a (argumentative relation annotation task)
  - H1b_regression_components.R: R script to run regression for hypothesis H1b (argument component annotation task)
  - H1b_regression_relations.R: R script to run regression for hypothesis H1b (argumentative relation annotation task)
  - H1b_scores_components.csv: Datapoints for hypothesis H1b (argument component annotation task)
  - H1b_scores_relations.csv: Datapoints for hypothesis H1b (argumentative relation annotation task)
  - annotated_scientific_paper_components: Final annotated scientific paper with argument component annotations (exemplary file based on just one worker annotation)
  - annotated_scientific_paper_relations: Final annotated scientific paper with argumentative relation annotations (exemplary file based on just one worker annotation)
  - final_aggregation.py: Script to aggregate crowd answers in final system
  - mace_runner.py: Helper file to run MACE
  - power_analysis.py: Helper file to perform power analysis
  - worker_answer_aggregation_components_H1b.py: Script to aggregate crowd answers and evaluate performance of component annotation task for experiment H1b
  - worker_answer_aggregation_components_h1a.py: Script to aggregate crowd answers and evaluate performance of component annotation task for experiment H1a
  - worker_answer_aggregation_components_pilots.py: Script to aggregate crowd answers and evaluate performance of component annotation task for the pilot studies
  - worker_answer_aggregation_relations_H1a.py: Script to aggregate crowd answers and evaluate performance of relation annotation task for experiment H1a
  - worker_answer_aggregation_relations_H1b.py: Script to aggregate crowd answers and evaluate performance of relation annotation task for experiment H1b
  - worker_answer_aggregation_relations_pilots.py: Script to aggregate crowd answers and evaluate performance of relation annotation task for pilot studies

- MTurk:
  - WorkerAnswers/Production/componentAnnotation: Directory containing anonymised crowd answers for all experiments regarding the argument component annotation task
    * H1a_worker_answers.txt:
    * H1b_worker_answers_test.txt:
    * Pilot_1_p2-4.txt:
    * Pilot_2_p16-18.txt:
  - WorkerAnswers/Production/relationAnnotation: Directory containing anonymised crowd answers for all experiments regarding the argumentative relation annotation task

        * H1a_worker_answers.txt:
        * H1b_worker_answers_test.txt:
        * Pilot_2_p2-4.txt:
        * Pilot_2_p16-18.txt:
    - qualitative_answer_analysis.ipynb: Jupyter notebook for the qualitative analysis of worker answers
    - save_worker_answers.ipynb: Jupyter notebook to download worker answers from AMT and save them in a text file

- Procfile: File that specifies the commands that are executed by the app on startup. Necessary to run Flask app on Heroku.

- app: Flask app
    - admin_area/admin_area.py:
    - data:
        * annotations_on_load_for_relation_HITs.py: Specifcation of different argument component lists which can be used as the input for a argumentative relation annotation task
        * annotations_on_load_helper_script.py:
        * batch1:
            · A11_ToBeAnnotated.json:
            · entire_paper/A11.txt:
        * filterStep1.json:
        * filterStep2.json:
        * preprocessing.ipynb:
    - static/styles: Directory containing a style file for each Flask app blueprint
    - templates: Flask front-end containing all Jinja2 templates
        * admin_area: Admin area back-end
        * base.html: Base template for Flask app
        * mturk_hit_templates/payment_for_non_submitted_HIT.html: HIT template for the payment of non-submitted HITs
        * relationannotation: This folder contains the different versions of the argumentative relation annotation task. The back-end renders an *instructions* file and the instruction file extends the corresponding *relationAnnotation* file.
            · relationAnnotation.html: Pilot 1 and 2
            · relationAnnotation_filterStep1.html: Experiment H1a
            · relationAnnotation_filterStep2.html: Experiment H1b
            · relationAnnotation_final.html: Final annotation task template
            · relationAnnotation_instructions.html: Pilot 1 and 2
            · relationAnnotation_instructions_filterStep1.html: Experiment H1a

            - relationAnnotation_instructions_filterStep2.html: Experiment H1b
            - relationAnnotation_instructions_final.html: Final annotation task instructions
        * textannotation: This folder contains the different versions of the argument component annotation task. The back-end renders an *instructions* file and the instruction file extends the corresponding *textAnnotation* file.
            - arguments_instructions.html: Pilot 1 and 2
            - arguments_instructions_filterStep1.html: Experiment H1a
            - arguments_instructions_filterStep2.html: Experiment H1b
            - arguments_instructions_final.html: Final annotation task instructions
            - textAnnotation.html: Pilot 1 and 2
            - textAnnotation_filterStep1.html: Experiment H1a
            - textAnnotation_filterStep2.html: Experiment H1b
            - textAnnotation_final.html: Final annotation task template
    - textannotation/textannotation.py: Back-end for annotation tasks
    - welcome_page/welcome_page.py: Back-end for welcome page

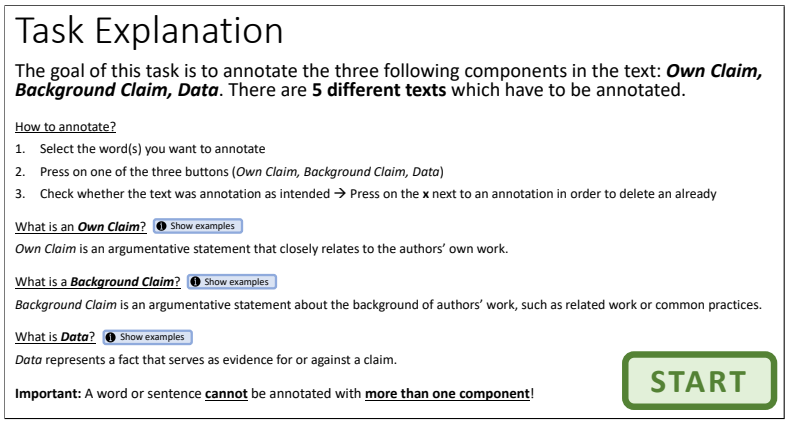
- config.py: File to configure important aspects of the Flask app, such as the app name, user logins or the configuration of batches of HITs.
- nltk.txt: Important so that Heroku downloads everything that is needed to enable the usage of nltk the tokeniser online.
- requirements.txt: Project requirements, for local installation as well as for automatic installation on Heroku.

B

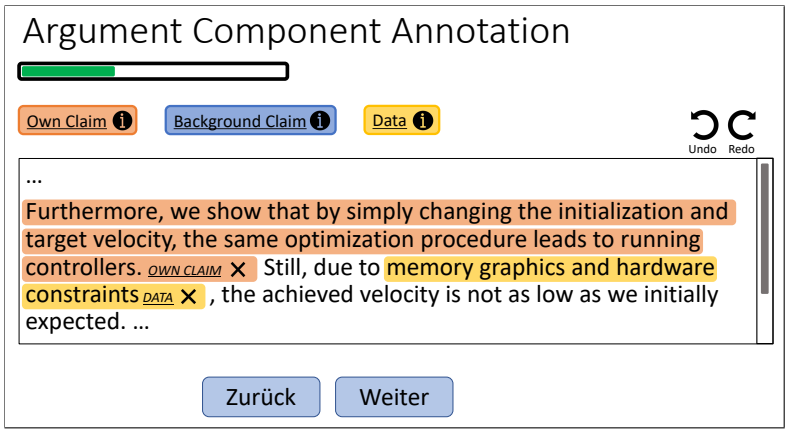
Annotation Tool Design

B.1 Mock-ups

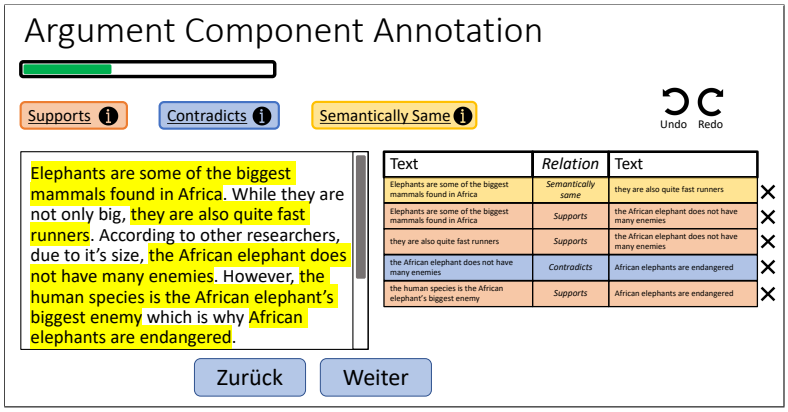
Figure B.1 includes the mock-ups we designed in the very beginning of this work. We implemented our annotation tool based on these mock-ups before we went on to evaluate and refine our implementation based on the insights we gained during the experiments on AMT.



(a) Argument component annotation instructions



(b) Argument component annotation task



(c) Argumentative relation annotation task

Figure B.1: Annotation tool mock-ups

B.2 Attention Check

Figure B.2 visualises the attention check that was included in the instructions of both HIT types during the pilots.

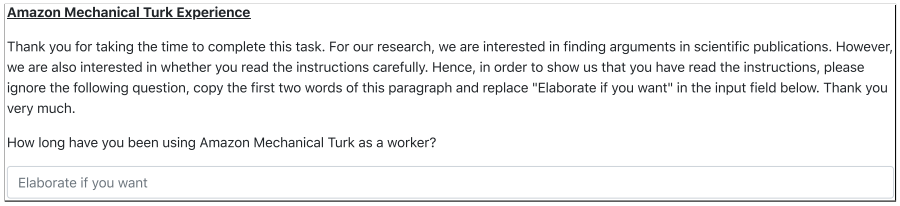


Figure B.2: Attention task

B.3 Argument Component Annotation Task

Instructions

Figure B.3 shows the instructions for the argument component annotation task. The blue info box regarding the pop quiz is not shown to workers who already passed the *ability filter* in a previous HIT of the same type.

Introduction and Instructions

Overview

We invite you to participate in an argument discovery study. We are exploring the effective discovery of arguments in scientific publications. After giving your consent, you will be presented three paragraphs of scientific publications which are to be annotated. The goal of this study is to simplify the creation of argument-annotated gold standard gold standard training data.

You will be presented three paragraphs of scientific publications. For each of these paragraphs you should annotate the three following components in the text: ***Own Claim***, ***Background Claim***, ***Data***.

A claim corresponds to the point an arguer is trying to make. A claim can be identified by looking at certain key phrases such as "*we think* ..." or "*we suggest* ..."
Interestingly, in scientific publications we can notice two types of claims:

- Own Claims: claims that are made about the work of the authors themselves.
- Background Claims: claims that describe the scientific community or the background of the work.

What is an Own Claim? ⓘ Example

Own Claim is an argumentative statement that closely relates to the authors' own work. In general, it provides really specific information.

What is a Background Claim? ⓘ Example

In contrast to the sub-category own claim presented before, *Background Claim* is an argumentative statement about the background of authors' work, such as related work or common practices. So these claims do not relate directly to the work presented in the subject paper but express a general believe or a general attitude regarding the domain.

What is Data? ⓘ Example

Data represents a fact that serves as evidence for or against a claim. It gives answer to the questions: What are the facts supporting the claim? What is the evidence? Why should someone believe this? In which particular case does the claim hold? It can for example be some kind of knowledge or an observation or even the result of an experiment. In scientific writing, citations often correspond to data, for example when the authors are referring to previous results to support their hypothesis. Sometimes, the authors use examples to support their argumentation. Therefore, in many cases, examples correspond to data. Especially when presenting results, the authors often refer to tables or figures in order to support their claims. In such a case, a figure corresponds to data. Always make sure that you only mark the minimal spans.

Steps

Carefully read all the instructions. **Always read the whole text from the beginning to the end to get an overview about the structure as well as about the overall content**. Then proceed to annotate the text by looking at one sentence at a time. Thank you! We very much appreciate the time that you put into helping us with our research.

How to annotate?

For each sentence you should do the following:

- **Decide if the sentence (or a part of the sentence) is of type *Data* or Claim and in case you identify a Claim, choose one of the two subcategories *Background Claim* and *Own Claim* (*If not, continue with the next sentence!*)**
- ***If yes*, select these words with your mouse and press on one the corresponding button above the text.**
  - **Choose how certain you are about this annotation on a scale from 1 to 3.**
    (1: I am not sure at all / 3: I am sure this annotation is correct)
  - **Select those words which made you decide to do this annotation.**
  - **Explain in your own words, what made you do this annotation.**
  - **Confirm your annotation by clicking on the button** Save Annotation**.**

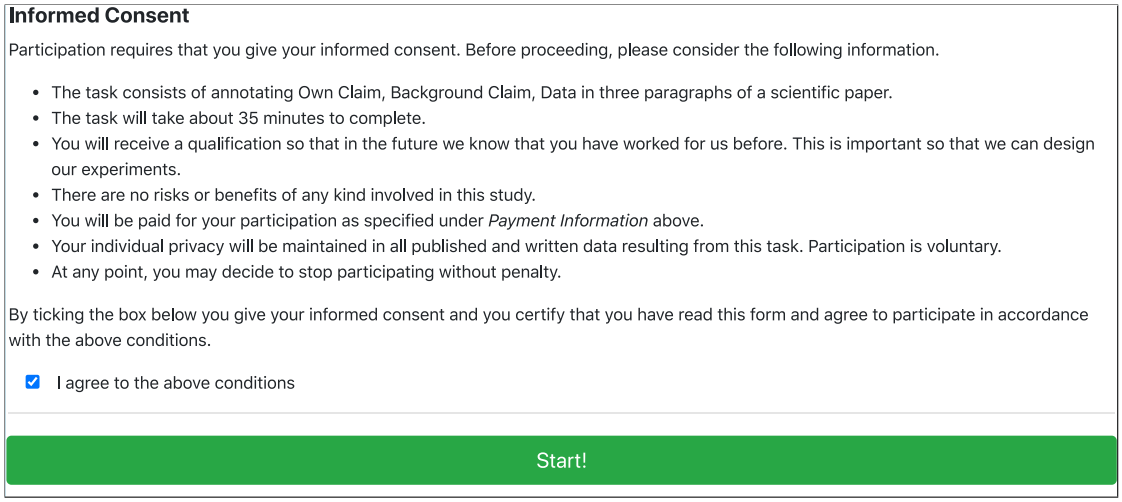
Important:

- The size of an annotation can be a single word, a few words, or a sentence!
- An annotation **can not** be longer than one sentence!
- The same word **can not be annotated twice**!
- Please always try to annotate the minimal text span and omit conjunctions, such as "because", when they are not part of an argument but rather connecting multiple.
- The annotated span must be understandable on its own.
- Try to **annotate as many argument components as possible**!

Payment Information

The final reward consists of two parts: base reward & bonus payment. The base reward is **$4,80** for every participant. In addition to the base reward, a bonus payment is paid to every participant. The more argument components you annotate correctly, the higher your bonus payment will be. We will pay a bonus of up to a maximum of **$3,20**.

(a) First part of the instructions

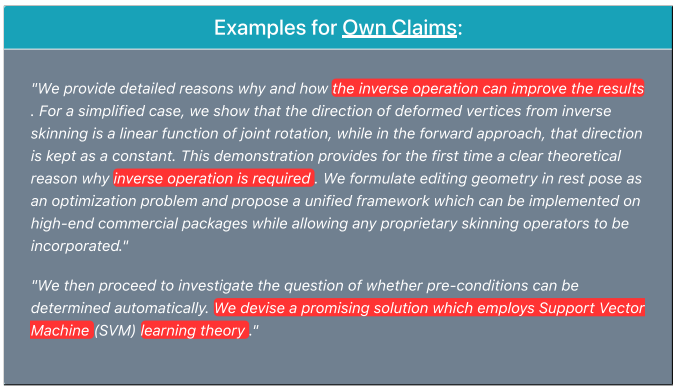


(b) Second part of the instructions

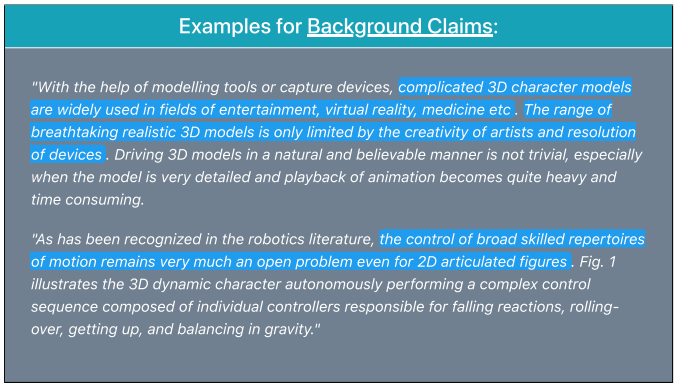
Figure B.3: Argument component annotation task instructions

Annotation Types Examples

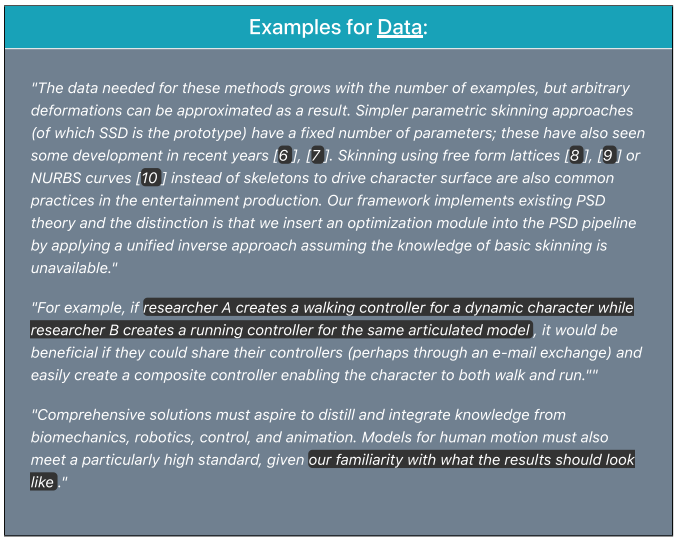
Figure B.4 shows examples for each argument component type. Crodworkers can access these examples in the HIT instructions as well as while annotating the paragraphs.



(a) Examples for *own claims*



(b) Examples for *background claims*



(c) Examples for *data*

Figure B.4: Examples for the three argument component types

Finish Survey in Final System

The following figure displays the finish page which is used in the final system. Instead of asking the workers detailed questions, as we did for the pilots, we just give the workers the opportunity to provide feedback if they wish.

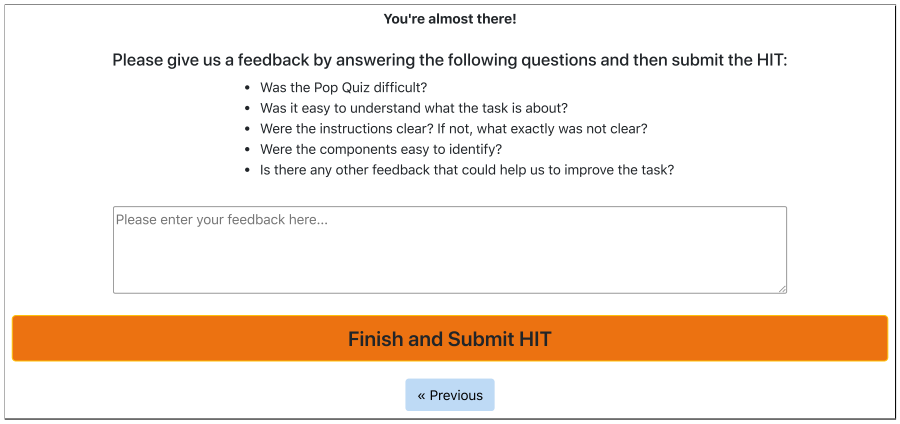


Figure B.5: Finish page to give feedback and submit HIT

Spammer Filter: Questions and Answers

Figure B.6 displays the question of the *spammer filter* with which the argument component annotation HIT was extended to evaluate the hypothesis H1a.

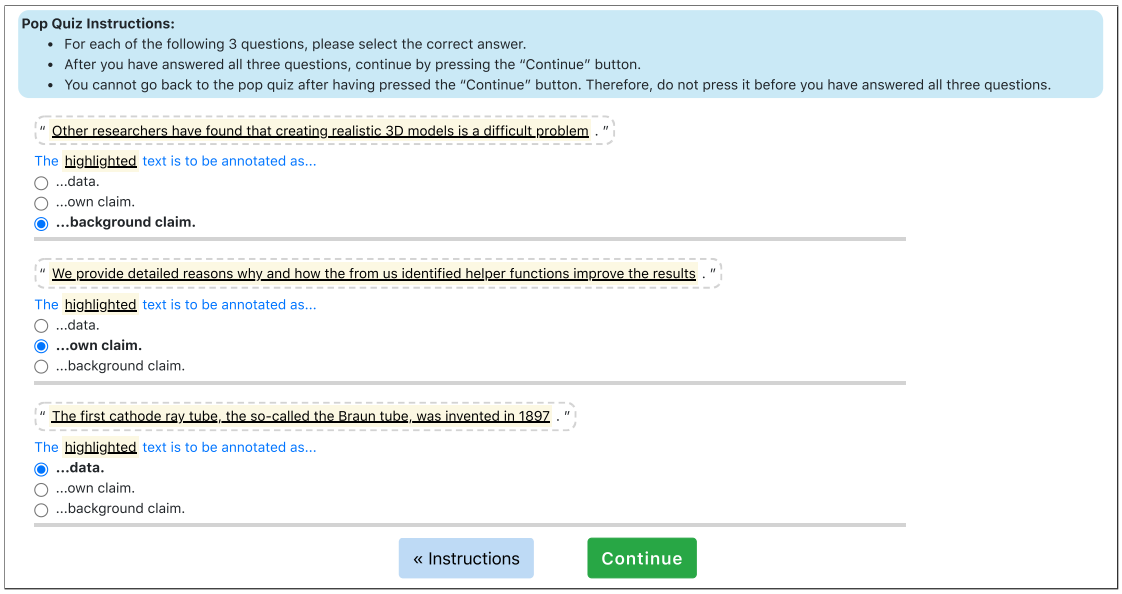
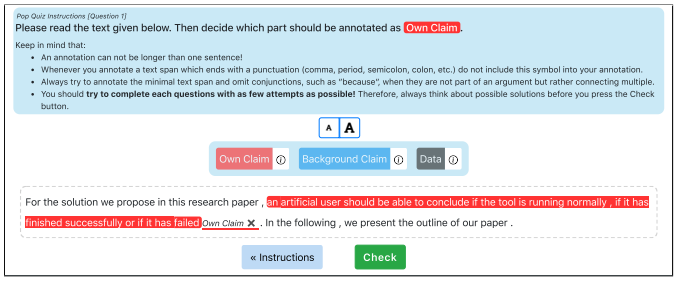


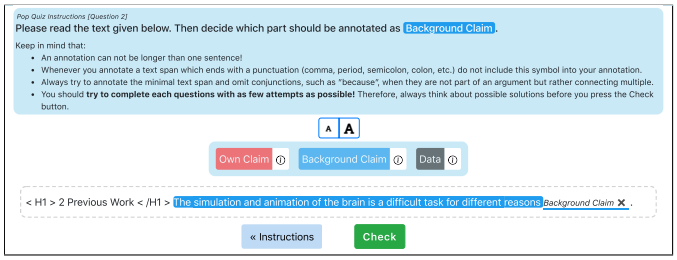
Figure B.6: Argument component HIT *spammer filter*: questions and answers

Ability Filter: Questions and Answers

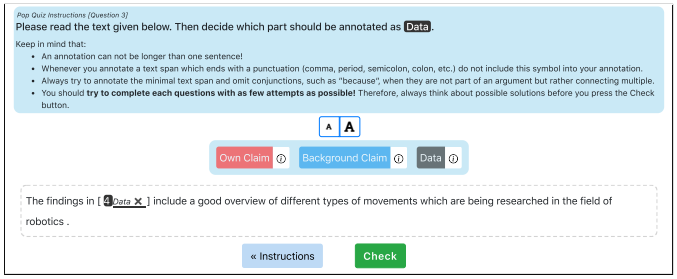
Figure B.7 shows the questions (including the correct answers) for the *ability filter* with which the argument component annotation HIT was extended to evaluate the hypothesis H1b.



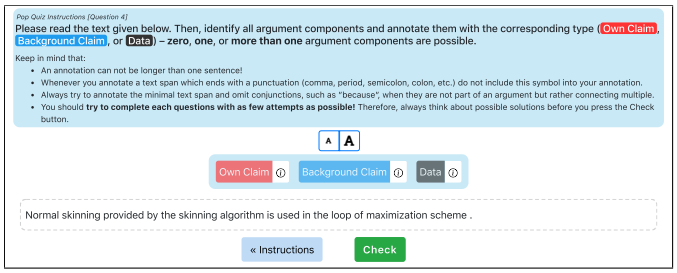
(a) Question 1 [level: easy]



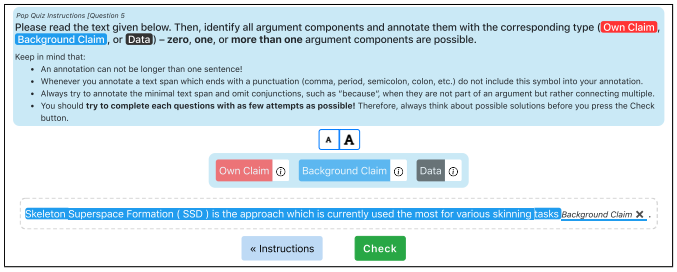
(b) Question 2 [level: easy]



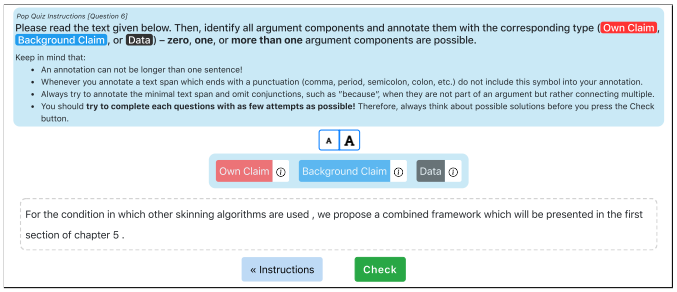
(c) Question 3 [level: easy]



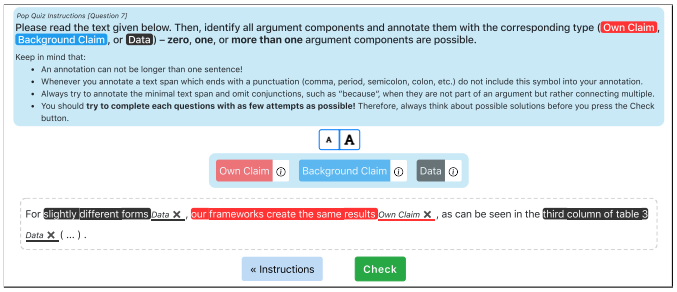
(d) Question 4 [level: medium]



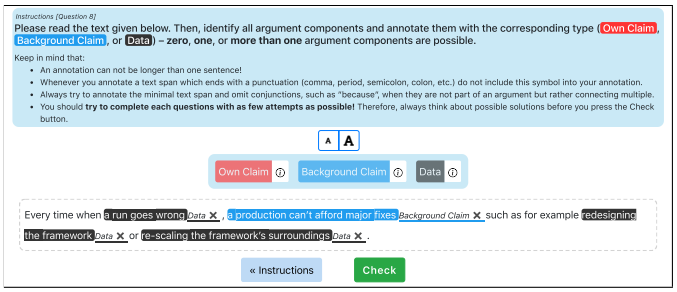
(e) Question 5 [level: medium]



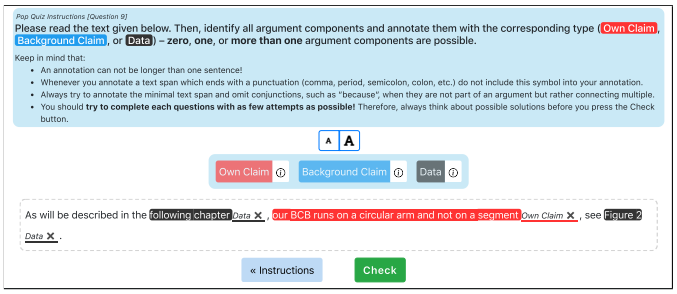
(f) Question 6 [level: medium]



(g) Question 7 [level: difficlut]



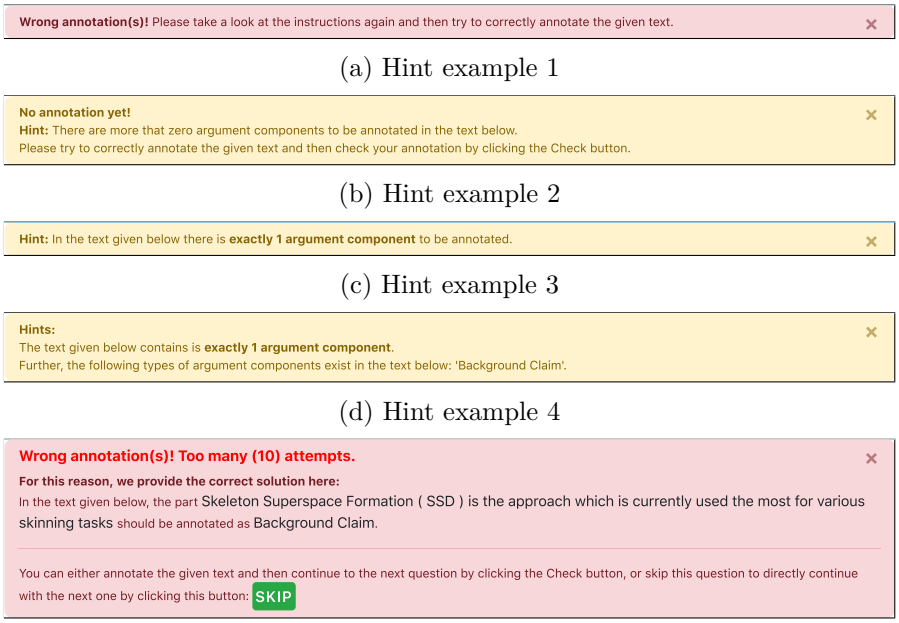
(h) Question 8 [level: difficlut]



(i) Question 9 [level: difficlut]

Figure B.7: Questions and answers of the argument component HIT *ability filter*

Figure B.8 shows some exemplary hints which we showed to the workers in case the answer they provided was wrong. The Subfigure B.8e visualises the situation where a worker needed ten wrong attempts and is therefore shown the correct solution to enable the continuation to the next question.



(a) Hint example 1

(b) Hint example 2

(c) Hint example 3

(d) Hint example 4

(e) Hint example 5

Figure B.8: Exemplary hints provided to workers to help solve the short annotation tasks

B.4 Argumentative Relation Annotation Task

Instructions

Figure B.9 shows the instructions for the argumentative relation annotation task. The blue info box regarding the pop quiz is not shown to workers who already passed the *ability filter* in a previous HIT of the same type.

Introduction and Instructions

Overview

We invite you to participate in an argumentative relation discovery study. We are exploring the effective discovery of argument relations in scientific publications. After giving your consent, you will be presented three paragraphs of scientific publications which are to be annotated. The goal of this study is to simplify the creation of argument-annotated gold standard gold standard training data.

You will be presented three paragraphs of scientific publications. Different argumentative component types (*Own Claim*, *Background Claim*, *Data*) are highlighted within each paragraph. For each possible combination of these highlighted sections you should decide whether they are related and if yes annotate one the three following relations: ***Supports***, ***Contradicts***, ***Parts of Same***.

Relation types:

What is a Supports relation? ⓘ Example

A *Supports* relation is a directed relationship from a component *a* to a component *b* if *a* backs *b*. Usually, this relationship exists from data to claim, but in many cases a claim might support another claim. Other combinations are still possible.

What is a Contradicts relation? ⓘ Example

A relationship of type *Contradicts* represents the counterpart of the Supports relationship, but in contrast, it is a bi-directional, i.e., symmetric relationship. An *Contradicts* relation exists between a component *a* and a component *b*, if *a* contradicts *b* and vice versa.

What is a Parts of Same relation? ⓘ Example

In addition to the two inter-argument relationships presented above, we have to specify a third "artificial" type of relation:
In real-world examples of arguments a single argumentative component, such as for example a claim, might be split up in several parts. Nevertheless, we would like to be able to recognize that those parts actually belong to the same argument. We call such a relation *Parts of Same*.

- For Parts of Same relations, it does not matter which of the two argumentative components is selected first and which is selected second.
- In case you encounter more than two instances of essentially the same component you do not have to draw the relationship to all instances, but to the last instance of that component that you have already connected.

Steps

Carefully read all the instructions. **Always read the whole text from the beginning to the end to get an overview about the structure as well as about the overall content**. Then, look at the highlighted argumentative components and then proceed to annotate the relations that hold between them. Thank you! We very much appreciate the time that you put into helping us with our research.

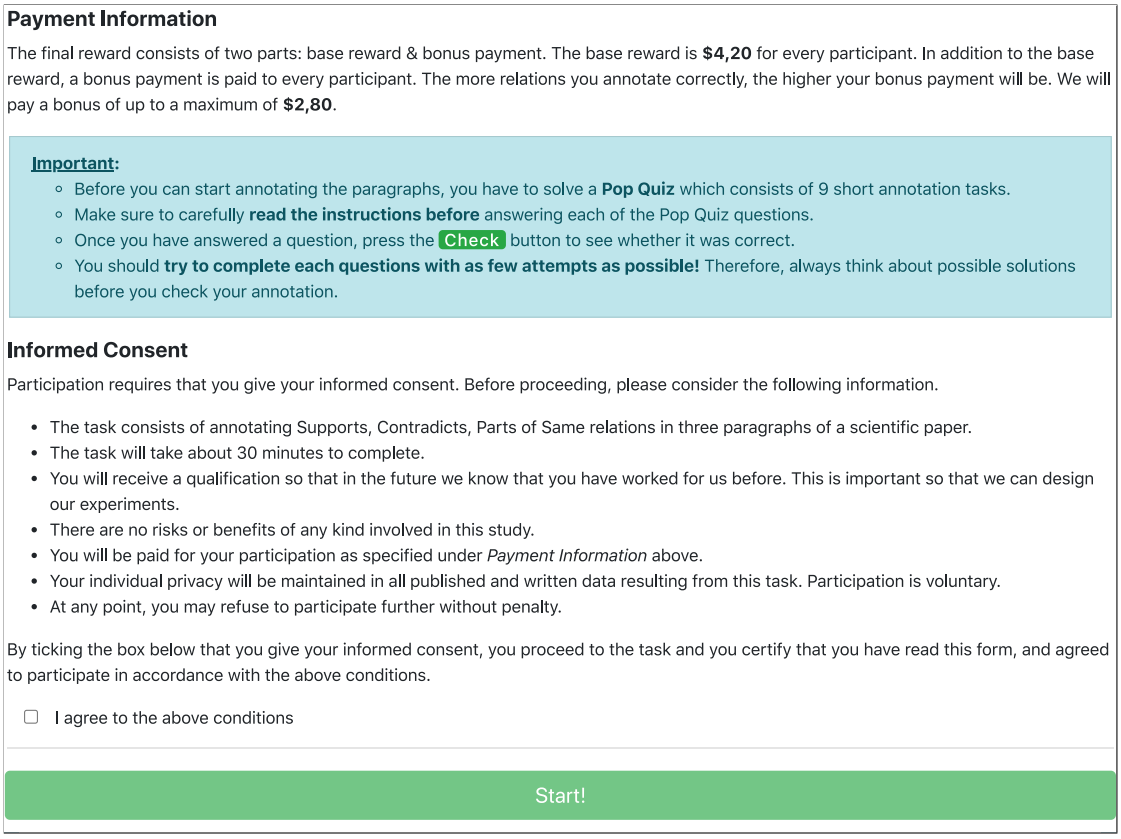
How to annotate the relations?

For each highlighted argumentative component you should do the following:

- **Are there other highlighted argumentative components shortly before or ofter which actually belong to the same argument and but are just split because of linguistic reasons?**
- ***If yes*, select both components by clicking on them with your mouse and press on one the button *Parts of Same*.**
  - Choose how certain you are about this relation on a scale from 1 to 3.
    (1: I am not sure at all / 3: I am sure this relation is correct)
  - For both selected components, select those words which made you decide to do this relation.
  - Finally, explain in your own words, what made you do this relation.
  - Confirm your annotation by clicking on the button Save Relation.
- **Think about possible relationships with other highlighted argumentative components.**
- **If you identify such a relationship: Decide of which type is it (*supports* or *contradicts*)? Click on both components and then click on one of the two buttons (*supports* or *contradicts*).**
  - Choose how certain you are about this relation on a scale from 1 to 3.
    (1: I am not sure at all / 3: I am sure this relation is correct)
  - For both selected components, select those words which made you decide to do this relation.
  - Finally, explain in your own words, what made you do this relation.
  - Confirm your relation by clicking on the button Save Relation.

Important:
- Try to identify as many relations as possible!
- An argumentative component can be part of many different relations!

(a) First part of the instructions

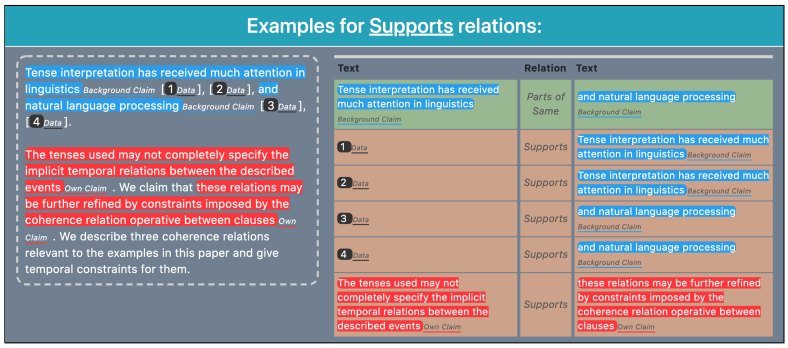


(b) Second part of the instructions

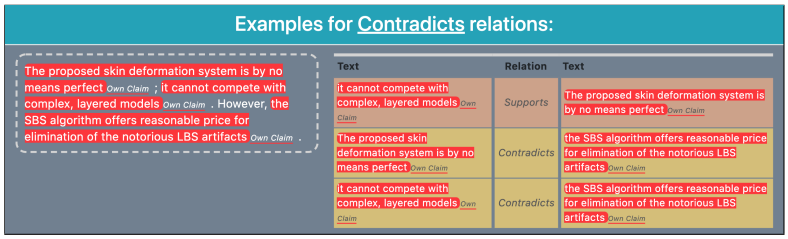
Figure B.9: Argumentative relation annotation task instructions

Annotation Types Examples

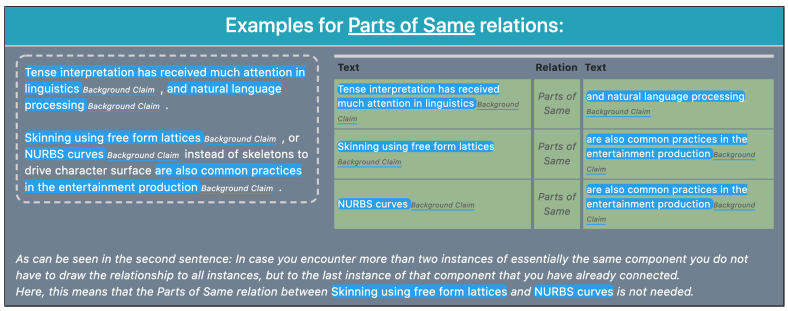
Figure B.10 shows examples for each argumentative relation type. Crodworkers can access these examples in the HIT instructions as well as while annotating the paragraphs.



(a) Examples for *supports* relations



(b) Examples for *contradicts* relations



(c) Examples for *parts of same* relations

Figure B.10: Examples for the three argument component types

Spammer Filter: Questions and Answers

Figure B.11 displays the question of the *spammer filter* with which the argumentative relation annotation HIT was extended to evaluate the hypothesis H1a.

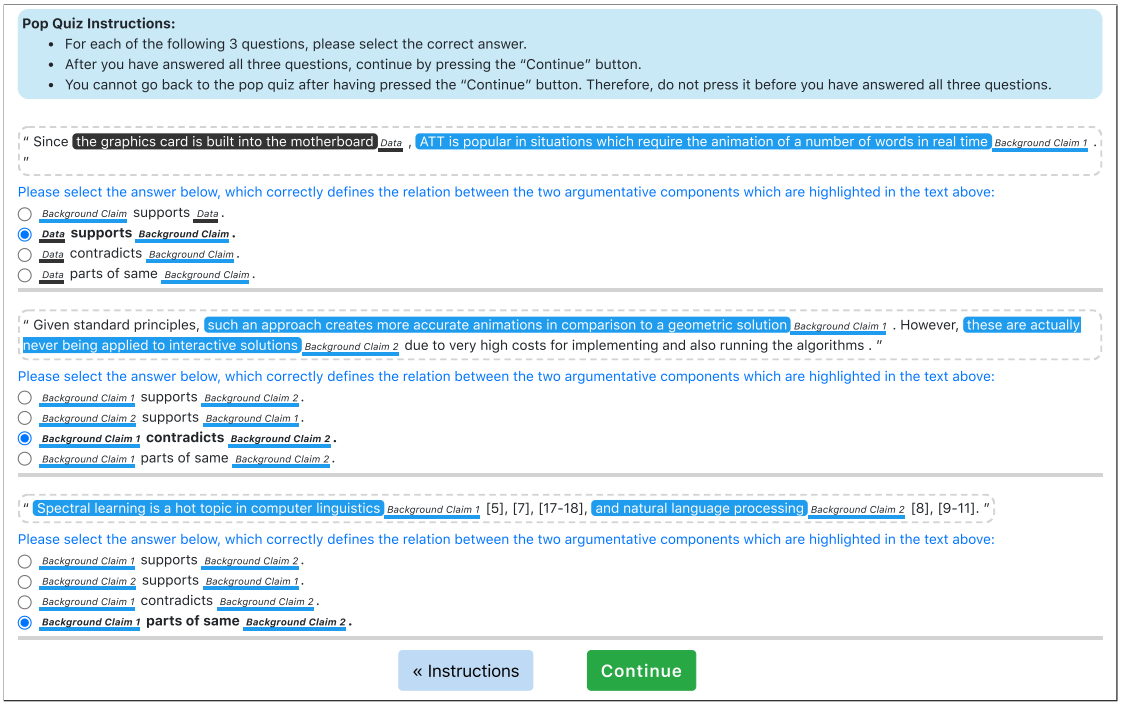
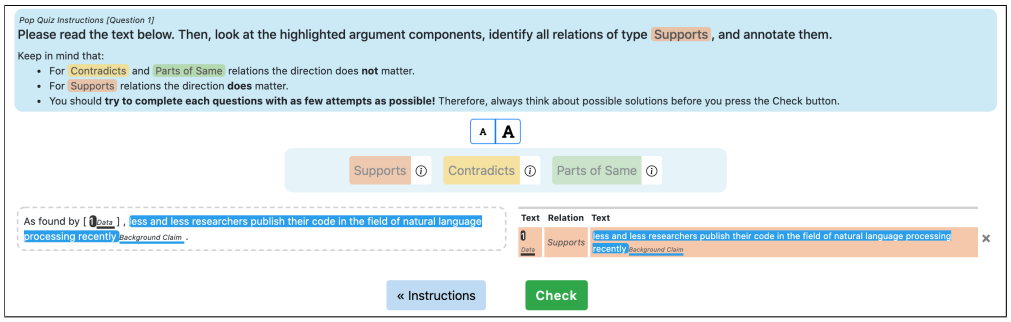


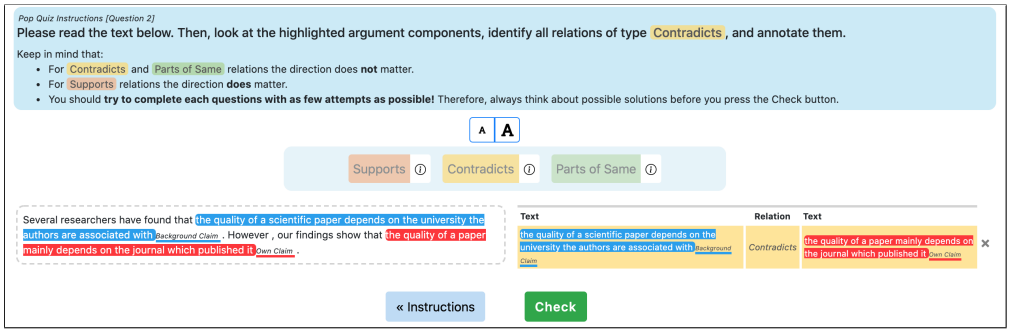
Figure B.11: Argumentative relation HIT spammer filter: questions and answers

Ability Filter: Questions and Answers

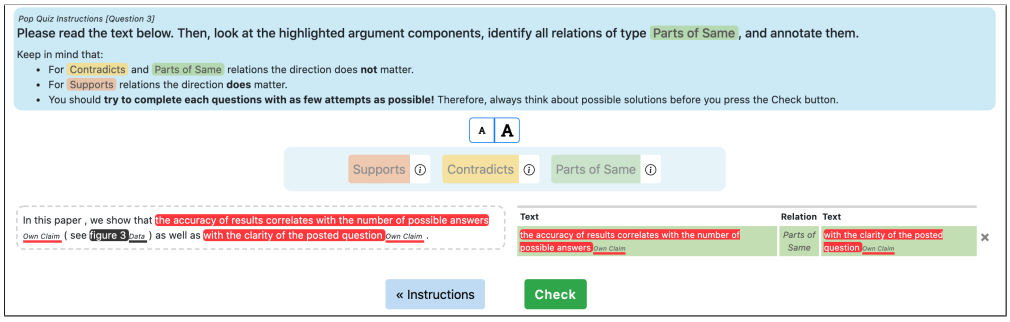
Figure B.12 shows the questions (including the correct answers) for the *ability filter* with which the argumentative relation annotation HIT was extended to evaluate the hypothesis H1b.



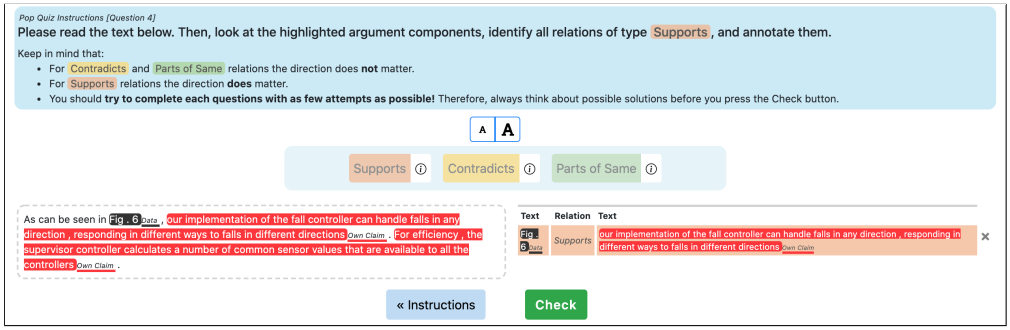
(a) Question 1 [level: easy]



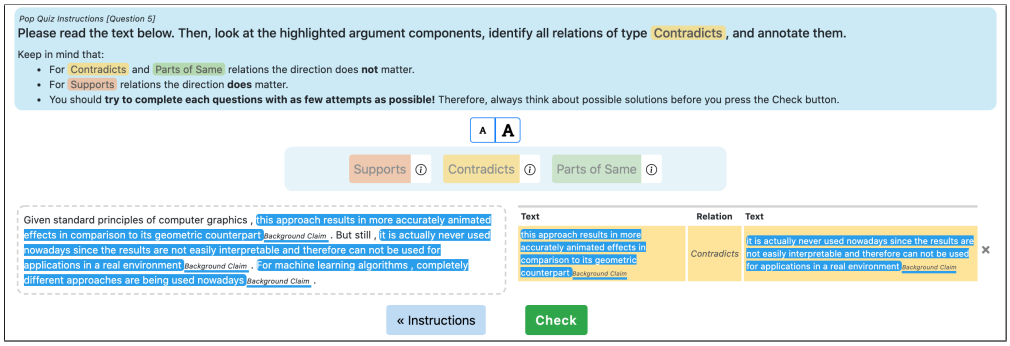
(b) Question 2 [level: easy]



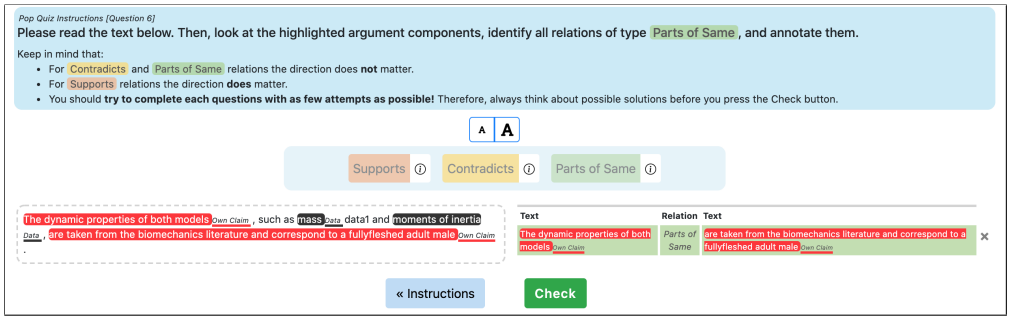
(c) Question 3 [level: easy]



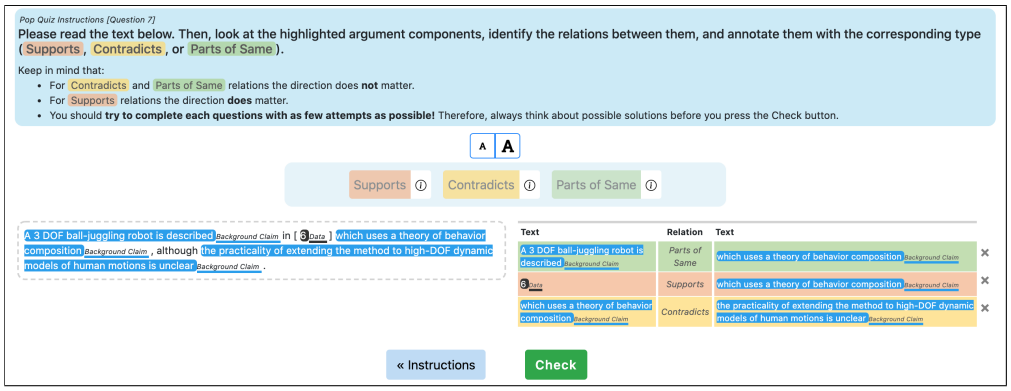
(d) Question 4 [level: medium]



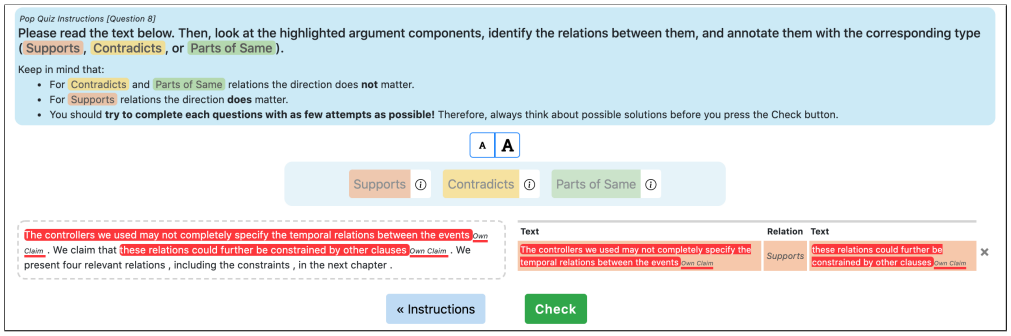
(e) Question 5 [level: medium]



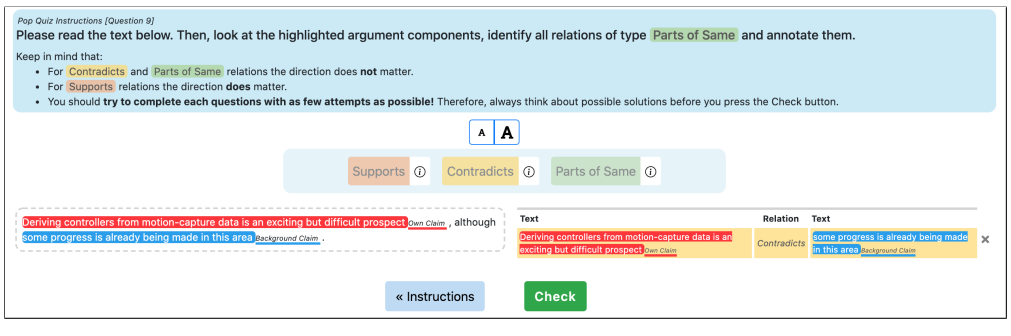
(f) Question 6 [level: medium]



(g) Question 7 [level: difficlut]



(h) Question 8 [level: difficlut]



(i) Question 9 [level: difficlut]

Figure B.12: Questions and answers of the argumentative relation HIT *ability filter*

C

Additional Results

C.1 Finish Survey Answers in Pilots

Figure C.1 visualises the survey workers had to fill out on the finish page of the argument component annotation HIT during the pilots.

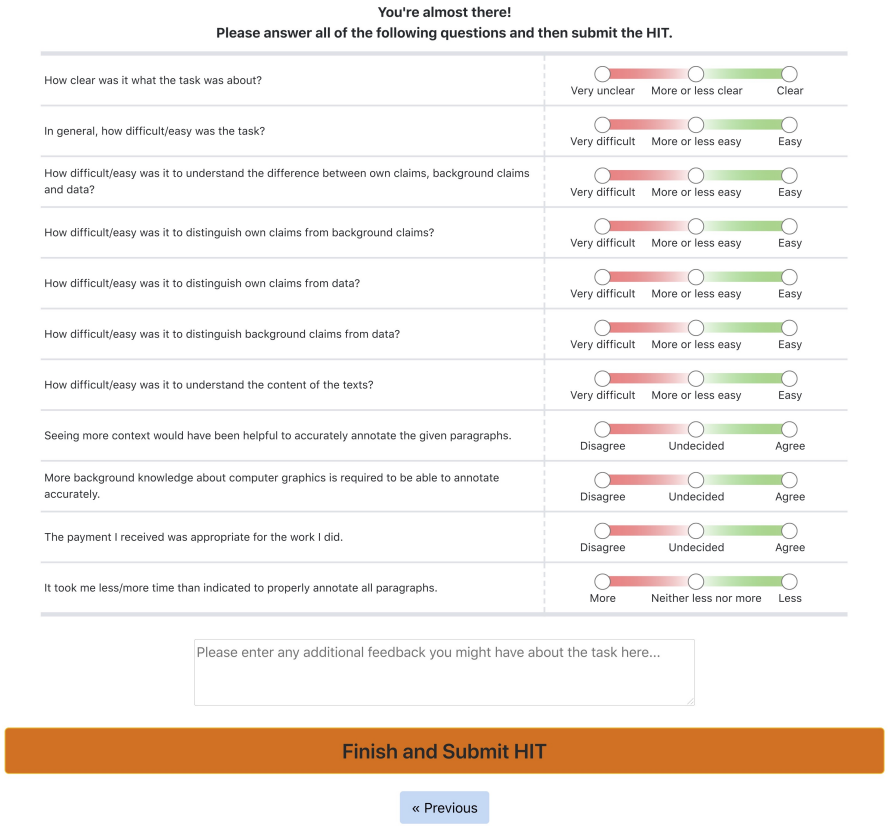


Figure C.1: Finish page survey of argument component annotation HIT during pilots

In the following, we list all answers for the survey on the finish page of the argument component annotation task in the two pilots. Note that we only consider answers from those workers who annotated at least one argument component. Further, note that not every worker answered to every question.

- How clear was it what the task was about?
  - 1 x very unclear, 5 x more or less unclear, 2 x clear
- In general, how difficult/easy was the task?
  - 3 x more or less difficult, 5 x very difficult
- How difficult/easy was it to understand the difference between own claims, background claims and data?
  - 5 x very difficult, 3 x more or less difficult
- How difficult/easy was it to distinguish own claims from background claims?
  - 1 x easy, 5 x more or less difficult, 1 x very difficult
- How difficult/easy was it to distinguish own claims from data?
  - 1 x easy, 5 x more or less difficult, 2 x very difficult
- How difficult/easy was it to distinguish background claims from data?
  - 4 x very difficult, 4 x more or less difficult
- How difficult/easy was it to understand the content of the texts?
  - 1 x easy, 4 x more or less difficult, 3 x very difficult
- Seeing more context would have been helpful to accurately annotate the given paragraphs.
  - 5 x agree, 1 x undecided, 2 x disagree
- More background knowledge about computer graphics is required to be able to annotate accurately.
  - 2 x agree, 4 x undecided, 2 x disagree
- The payment I received was appropriate for the work I did.
  - 5 x agree, 3 x undecided
- It took me less/more time than indicated to properly annotate all paragraphs.
  - 4 x neither less nor more, 4 x more

Figure C.2 visualises the survey workers had to fill out on the finish page of the argument component annotation HIT during the pilots.

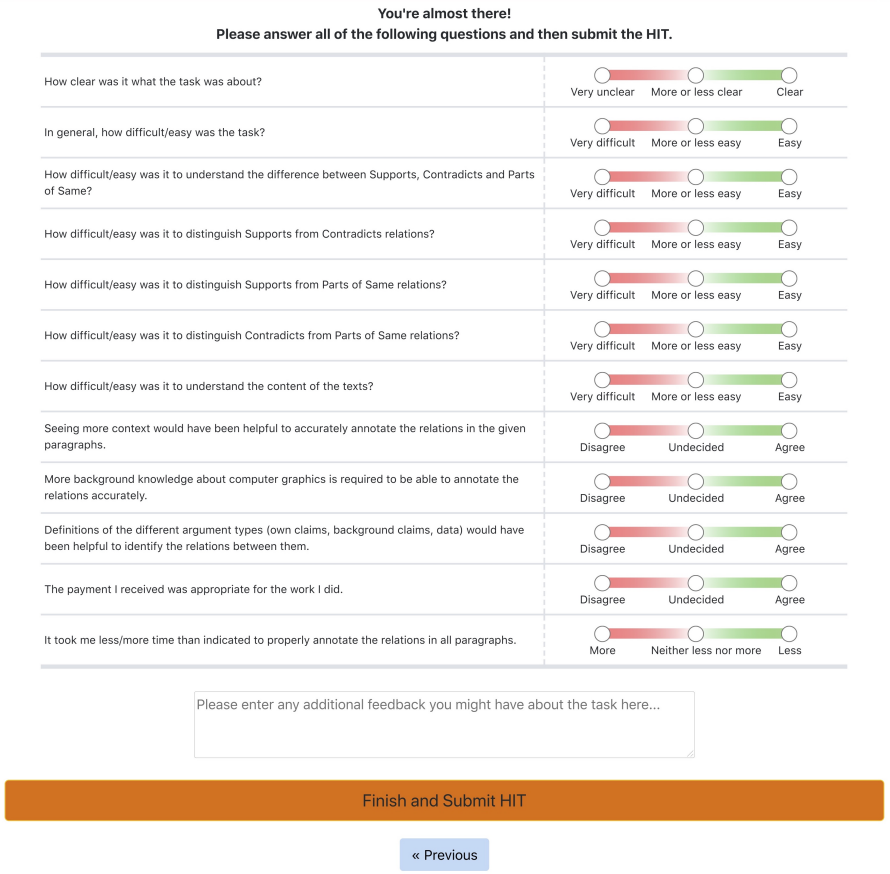


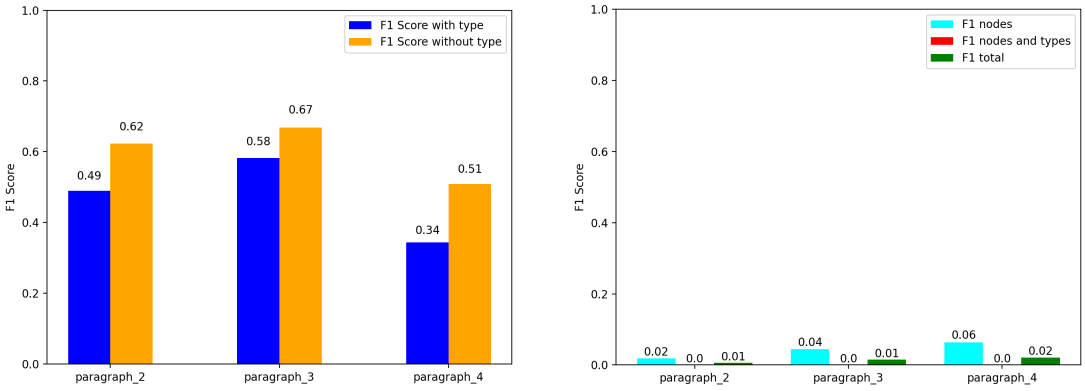
Figure C.2: Finish page survey of argumentative relation annotation HIT during pilots

In the following, we list all answers for the survey on the finish page of the argumentative relation annotation task in the two pilots. Note that we only consider answers from those workers who annotated at least one argumentative relation. Further, note that not every worker answered to every question.

- How clear was it what the task was about?
  - 4 x more or less unclear, 4 x clear
- In general, how difficult/easy was the task?
  - 3 x very difficult, 2 x more or less difficult, 3 x easy
- How difficult/easy was it to understand the difference between Supports, Contradicts and Parts of Same?
  - 1 x very difficult, 4 x more or less difficult, 3 x easy
- How difficult/easy was it to distinguish Supports from Contradicts relations?
  - 1 x very difficult, 4 x more or less difficult, 3 x easy
- How difficult/easy was it to distinguish Supports from Parts of Same relations?
  - 2 x very difficult, 3 x more or less difficult, 3 x easy
- How difficult/easy was it to distinguish Contradicts from Parts of Same relations?
  - 1 x very difficult, 1 x more or less difficult, 6 x easy

- How difficult/easy was it to understand the content of the texts?
  - 3 x very difficult, 3 x more or less difficult, 2 x easy
- Seeing more context would have been helpful to accurately annotate the relations in the given paragraphs.
  - 3 x undecided, 5 x agree
- More background knowledge about computer graphics is required to be able to annotate the relations accurately.
  - 2 x disagree, 2 x undecided, 4 x agree
- Definitions of the different argument types (own claims, background claims, data) would have been helpful to identify the relations between them.
  - 1 x disagree, 4 x undecided, 3 x agree
- The payment I received was appropriate for the work I did.
  - 3 x undecided, 5 x agree
- It took me less/more time than indicated to properly annotate the relations in all paragraphs.
  - 2 x less, 3 x neither less nor more, 3 x more

C.2 H1a: Average Crowd Performance per Paragraph

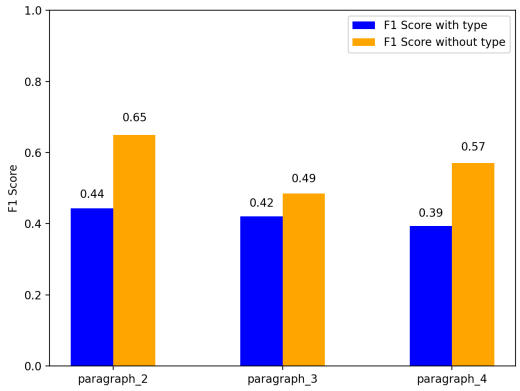


(a) Argument component annotation performance

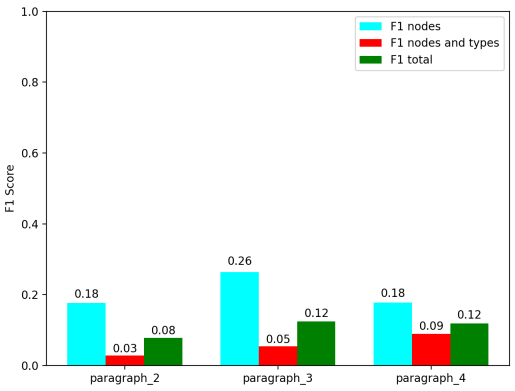
(b) Argumentative relation annotation performance

Figure C.3: H1a: average crowd performance per annotated paragraph

C.3 H1b: Average Crowd Performance per Paragraph



(a) Argument component annotation performance



(b) Argumentative relation annotation performance

Figure C.4: H1b: average crowd performance per annotated paragraph

C.4 Crowdworker Feedbacks

Table C.1 lists the feedback crowdworkers provided in all our experiments. In total, 15 workers did not provide any feedback.

Experiment	Type	Paragraphs	Feedback
Pilot 1	AC	2-4	NICE TO JOB
Pilot 1	AC	2-4	Very interesting Task
Pilot 1	AC	2-4	very very nice
Pilot 1	AC	2-4	please add some more examples
Pilot 1	AC	2-4	good
Pilot 1	AC	2-4	51 minutes rather than 34, although still plenty of time left over. I left a bunch of sentences which were basically a table of contents un-annotated; these are arguably facts, though I'd say \"meta-facts\" would be more accurate (no category for that!).
Pilot 1	AC	2-4	Thanks!
Pilot 2	AC	16-18	Happy to participate on annotation task.
Pilot 2	AR	2-4	Interesting task.
Pilot 2	AR	2-4	good survey
Pilot 2	AR	2-4	Very interesting survey, although maybe a few more examples in the instructions of annotating a paragraph with Support/Contradict/Parts of Same would have been helpful to give people a clearer idea of what to do.
Pilot 2	AR	16-18	Perhaps this was part of the study as it was addressed in the questions above but it was rather difficult to understand the nature of the text snipped without further context
Pilot 2	AR	16-18	good task about words relationship
H1a	AC	2-4	very good survey.
H1a	AC	2-4	The Pop Quiz was easy,The task is little difficult to understand.
H1a	AC	2-4	The pop quiz was not difficult. It was somewhat difficult to understand what the task is about. The instructions are somewhat clear. The components were challenging to identify.
H1a	AC	2-4	Amazing
H1a	AC	2-4	Pop quiz was very easy.\rTask is easy to understood, to annotate the own claim, background claim, data on given paragraph.\rAll instruction are very clear to understand.\rSomewhat difficult to identify the components\rIts very interested task, but i think it difficult for me to identify the components. Thank you.
H1a	AC	2-4	No the pop Quiz is very easy to me . Yes it is easy to understand the task. Yes the instructions are very clear. yes components are easy to identify .
H1a	AC	2-4	This is good task
H1a	AC	2-4	The pop quiz is easy.I understand the task.Instruction was clear.Easy to identify the components.It is interesting.

H1a	AC	2-4	The pop quiz was not difficult. The instructions were clear, but I think more examples could be useful. Some of the components were easy to identify and some were a little difficult.
H1a	AC	2-4	easy to understand
H1a	AC	2-4	Pop quiz was easy\rTask was a bit complicated and difficult\rDon't understand why I have to highlight individual words within the sentence I selected\rEven with the full instructions and examples I didn't feel confident in identifying the components\rMaybe try to state more clearly one single way to identify each component\rThe task is badly underpaid without the potential bonus
H1a	AC	2-4	the Pop quiz was easy, the task was a little confusing/daunting but I understood the instructions after parsing them properly. The components were a little hard to separate, but with a little practice I could probably do better. I would have more examples of each component for those who want it
H1a	AC	2-4	The pop quiz was easy and the task was easy to understand. Instructions were clear but more examples could have been provided. Annotation was somewhat difficult particularly related to data. Nice hit and I have done ample of data annotation tasks, so I can say that your tool is nice but it can be improved.
H1a	AR	2-4	This task was incredibly difficult and confusing. It was hard enough just to understand the actual scientific texts, much less the relationships between all the highlighted sentences. I found the instructions vague and confusing. I don't know why after selecting a text span, I am made to then select individual words and explain the reason for selecting it when the selection of text and their relation should already explain the reason they were selected... Everything about this was difficult and I don't think I did very well. I apologize if I misunderstood some parts of the task.
H1a	AR	2-4	no issuess...instruction is clear , identification of relation is little-bit difficult to co-relate to each other.
H1a	AR	2-4	Interesting
H1a	AR	2-4	This is good work. Thank you!!!!
H1a	AR	2-4	Good
H1a	AR	2-4	It was easy because the instruction was clear and simple to read. Some relations are difficult to understand but i did my best.
H1a	AR	2-4	No\rSome time to take understand\rYes there is clear instruction\rNo there is enough information for taking the survey.
H1a	AR	2-4	I was a little confused on the 3rd question on the Pop Quiz. I would like to try the Hit again in the future. It looked like it would have been a really interesting task.
H1a	AR	2-4	Good survey
H1a	AR	2-4	It was useful to improve my knowledge
H1a	AR	2-4	Nice
H1a	AR	2-4	It is difficult of task and instruction is clear.
H1b	AC	2-4	very hard
H1b	AC	2-4	this was pretty difficult. i did my best.
H1b	AC	2-4	its very interest experiment
H1b	AC	2-4	its very grate
H1b	AC	2-4	Wish claim/data definitions were given during quiz and task as well and were comprehensive.
H1b	AC	2-4	The instructions was clear and the hits were more usefull but timing should be for 2 hrs
H1b	AC	2-4	very difficult.
H1b	AC	2-4	this is very difficult
H1b	AR	2-4	1. The quiz was somewhat difficult.\r2. It was easy to understand ho to do the task.\r3. Everything was clear.\r4. It was hard to identify relations.\r5. No other feedback.
H1b	AR	2-4	It was so hard. Not easy at all. Even with a bachelor's and master's degree, this is very confusing.
H1b	AR	2-4	Sorry to inform you that Its some what very difficult to understand it. It would be easy if you would have provided few more example. Thank you!
H1b	AR	2-4	The time given is not enough to complete the hit. More time should be allocated. The limited time made me unable to finish the last task completely. Otherwise, everything is fine.
H1b	AR	2-4	yes i like hit
H1b	AR	2-4	Pop Quiz difficult
H1b	AR	2-4	yes i like hit
H1b	AR	2-4	this is really difficult
H1b	AR	2-4	Pop Quiz difficult

Table C.1: Feedback provided by crowdworkers. The acronyms stand for: Argument Component Annotation HIT (AC), Argumentative Relation Annotation HIT (AR).

D

Contents of the Digital Submission

This thesis was submitted digitally, containing the following files:

- German abstract (zusfsg.txt)
- English abstract (abstract.txt)
- Master's thesis (masterarbeit.pdf)
- Archive of the code repository (Uploaded to GitLab: *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-joachim-baumann*)
- AMT experiment data (Uploaded to seafile directory "MT Data JBaumann", no link provided for crowdworker privacy reasons)

List of Figures

List of Tables