Argument Discovery via Crowdsourcing

ABSTRACT

The amount of controversial issues being discussed on the Web has been growing dramatically. In articles, blogs, and wikis, people express their points of view in the form of arguments, i.e., claims that are supported by evidence. Discovery of arguments has a large potential for informing decision-making. However, argument discovery is hindered by the sheer amount of available Web data and its unstructured, free-text representation. The former calls for automatic text-mining approaches, whereas the latter implies a need for manual processing to extract the structure of arguments.

In this paper, we propose a crowdsourcing-based approach to build and maintain a corpus of arguments, an *argumentation base*, thereby mediating the trade-off of automatic text-mining and manual processing in argument discovery. We develop an end-to-end process that minimizes the crowd cost while maximizing the quality of crowd answers by: (1) ranking argumentative texts, (2) proactively eliciting user input to extract arguments from these texts, and (3) aggregating heterogeneous crowd answers. Our experiments with real-world datasets highlight that our method discovers virtually all arguments in documents when processing only 25% of the text with more than 80% precision, using only 50% of the budget consumed by a baseline algorithm.

1. INTRODUCTION

The Web became the central medium for the public discussion of controversial issues, spanning manifold domains and influencing the political, societal, and technical discourse. People constantly share knowledge, report scientific studies, upload comments, and write reviews via various Web sources, such as blogs, social media websites, commercial websites, and wikis. As a consequence, the Web emerged as the prime source for a wide range of controversial arguments and interrelated information. While the amount of Web data is expected to grow dramatically [39], already nowadays, it provides a unique opportunity to exploit the *wisdom of the crowd*.

Argument-Driven Decision-Making. One way of exploiting Web data is to use it as a source of arguments that inform decision making. An argument commonly consists of a claim that is supported by evidence, or, put differently, some premises and a conclusion. As such, an argument provides not only a viewpoint found on the Web, but includes also the provenance behind it. A corpus of such arguments, an *argumentation base*, has very wide applicability that is independent of the truth values of premises or conclusions. An argumentation base supports people to engage in discussions, to understand new problems, to perform scientific reasoning, to justify their opinions, and to foster agreement. For instance, understanding the arguments put forward in a law suit helps to achieve consensus. In scientific research, an argumentation base enables conclusions on the appropriateness of a chosen process. Turning to automated information processing, argumentation is the foundation of systems that formally capture the meaning of data, enabling systematic and meaningful processing. Applications that benefit from an argumentation base include systems for query answering [65], human-computer debating [66], decision-support [20], and recommendations [28].

Challenges of Argument Discovery. Construction of an argumentation base from text documents has been introduced as *argument discovery* or argument mining [44, 49]. This construction includes the detection of all the arguments in the documents along with their individual (local) structure, and their argumentative relations. However, it does not refer to an evaluation of their convincing power, i.e., an assessment of the truth values of premises and conclusions. In other words, argument discovery calls for correctness in the detection and extraction of arguments, without concluding on the correctness or appeal of the discovered arguments.

The broad availability of documents on the Web is an unprecedented opportunity for argument discovery. Yet, the construction of an argumentation base is hindered by the sheer amount of raw data and its unstructured, free-text representation prevalently encountered in the Web. A large number of documents, which may be of considerable length (e.g., legal documents) or are created in real-time (e.g., on social platforms) need to be processed. Hence, a fully manual detection and extraction of arguments is not feasible. On the other hand, Web documents are unstructured, do not follow formal guidelines, and may lack proper syntax or spelling. Consequently, it is non-trivial to identify argumentative roles of parts of a document and derive the structure of arguments.

Contributions. In this paper, we develop a semi-automatic approach to discover arguments from a corpus of documents. To cope with large-scale data and the inherent uncertainty of automatic text processing, we integrate automatic techniques and crowdsourcing, a paradigm that was shown to be effective for handling free-text documents [54]. Specifically, we proceed in three steps. First, we detect and rank documents that are likely to contain arguments. Second, candidate claims and evidence are automatically extracted and used as input for argument crowdsourcing. Third, crowd answers are aggregated to obtain a single trusted set of arguments.

We integrate the three steps in an *iterative learning process*. Answers obtained by crowdsourcing are used as training data for subsequent iterations of the process: crowd answers are used to (1) train a feature-based scoring model to rank texts by their likelihood to contain arguments, and (2) estimate the reliability of workers, which, in turn, guides the aggregation of answers.
Our contributions can be summarized as follows:

- *Iterative Learning Process for Argument Discovery.* Section 3 presents a model for argument discovery and, based thereon, develops an iterative learning process to construct an argumentation base. The process includes ranking of document paragraphs, argument crowdsourcing, and aggregation of arguments.
- *Paragraph Ranking.* Section 4 introduces a method to rank the paragraphs of text documents by their likelihood of containing arguments. The ranking helps us to focus on the most promising parts of a text, increasing the likelihood to detect arguments. The ranking is obtained by training a scoring model that exploits lexical and syntactical features.
- *Argument Crowdsourcing.* In Section 5, we show how to use crowdsourcing to extract arguments from the ranked document paragraphs. We elaborate on the design of questions to be posted to crowdsourcing platforms and present a technique to automatically identify candidate claims and evidence to instantiate these questions for a particular paragraph.
- *Argument Aggregation.* Section 6 presents a method to aggregate candidate arguments obtained from crowd workers to form a single trusted set of arguments. We model the relations between workers, answers, and arguments as a factor graph and leverage probabilistic techniques to obtain an aggregated result.

The remaining sections are structured as follows. Section 7 presents an evaluation of our method showing that it is effective, achieving high precision and recall, and efficient, consuming only 50% of the budget needed by a baseline algorithm for the same result. Section 8 reviews related work, before Section 9 concludes the paper.

2. BACKGROUND

Argumentation is an important element of human communication. It structures a discourse by enabling humans to convey their opinions in a justifiable way. Moreover, argumentation is a central phenomenon in the Web, where users can express their opinions without much restrictions in terms of traceability, plausibility, or appropriateness of their arguments. As such, the Web provides us with an unprecedented opportunity to discover arguments, i.e., to identify arguments in terms of their local structure as well as the relations between them.

Arguments. At the heart of argumentation is the notion of an *argument*, for which different philosophical interpretations have been proposed, see, for instance, the Toulmin Model [24]. In this work, we adopt an argument model that is widely established in Computational Linguistics. That is, an argument refers to a controversial statement (aka topic) and consists of two components: a claim and evidence. A claim is defined as *'a general concise statement that directly supports or contests the topic'* [2], while evidence is *'a text segment that directly supports a claim in the context of a given topic'* [2]. In other words, evidence provides the reason or justification supporting the claim, while the claim concludes the argument based on the assumptions provided by the evidence [7].

The identification of the abstract structure of an argument is a prerequisite for the evaluation of its *convincing power*, i.e., the strength of its justification [55]. Such an evaluation refers to the *verification* of the truth value of the evidence and the assessment of the *traceability* of the claim given the evidence. Evidence is assumed to be factual, so that it can be verified in an objective manner. A claim, in turn, cannot be verified—only the relation between the claim and the evidence of an argument can be assessed. The literature knows various dimensions for the evaluation of this relation, starting from the quantity of provided evidence, over the argumentation depths, to different types of appeal, see [55, 60]. However, this evaluation is inherently subjective and depends on prior beliefs, points of view, and background knowledge of an individual. As a result, in various debate systems [43, 47], arguments are discovered automatically, but their persuasiveness is assessed by users.

For instance, the following paragraph stems from a Web document that discusses processed food.

EXAMPLE 1. (S_1) *Processed meats are treated with a variety of additives that could be harmful to health if eaten too often and in large quantities.* (S_2) *That being said, an occasional hot dog or meat sandwich probably won't kill you.* (S_3) *But eating them all the time can result in health concerns: up to a 42-percent increase in the risk of heart disease and a 19-percent increase in the risk of diabetes, according to Reuters.* (S_4) *MayoClinic.com adds that most people in America eat too much salt and switching to fresh cuts of meat and fish can help you reduce the amount of salt in your diet.* (S_5) *This will also help you cut the amount of preservatives you are eating, making you healthier.*

Here, segment S_1 formulates a *claim* regarding the treatment of processed meat with additives and their negative impact on one's health. One may argue that S_1 defines two claims (the use of additives and the health implications). However, we notice that the subsequent segments provide evidence for the claim in S_1 as a whole. Segment S_3 presents empirical statements as justification, while segment S_4 adds abstract factual statements. Both segments refer to supposedly reliable sources (Reuters, MayoClinic), which is known as *appeal to authority* [55]. Given the above information, we can derive an argument $\{S_3, S_4\} \rightsquigarrow S_1$, built from the claim S_1 and the evidence given in S_3 and S_4. Whether this argument is considered to be convincing depends on the beliefs of the reader. Some will immediately trust the link established by the empirical statements, whereas others may require further details on the respective studies to be convinced or question the authority of the cited sources.

Argument Discovery. The detection of the abstract structure of argumentation, its elementary units and their relations is known as argument discovery or argument mining [44, 49]. Argument discovery establishes a relation between a text and concepts of discourse theory [62], such as claims and conclusions, regardless of their convincing power. As such, it provides a broad overview of *what* are the arguments brought forward in a discourse and thereby enables discussion, supports reasoning, justifies opinions, and fosters agreement. By focusing on the argumentative structure, argument discovery is independent of subjective factors, such as beliefs or background knowledge of particular people, which are relevant only when evaluating arguments in terms of their convincing power.

Argument discovery has to cope with the unstructured, free-text representation of arguments encountered in documents. Detection and extraction of arguments is hindered by a lack of a formal syntax of the argumentative structure as well as various linguistic phenomena. As a consequence, argument discovery has long been done by manual processing. For instance, IBM Watson's Debater [2] exploits an argumentation base that has been created by expert users.

A manual approach to argument discovery, however, does not scale to the number of documents found on the Web. Since most Web retrieval techniques are keyword-based, the input for argument discovery becomes a very large number of text documents of considerable length that may even be created in real-time.

Recently, techniques for fully-automatic argument discovery have been presented in order to cope with large-scale input data. Typically, these approaches follow a two-step approach. First, given a set of documents, parts of the document that may contain an argument are detected. To this end, common machine learning approaches, such as SVM [43] or LDA [47], are used to classify texts based on three classes of features: indicative linking words for argumentation ('consequently', 'in conclusion'), claim-related features (prototypical words such 'argue', 'believe') and evidence-related features (numbers, citations, cue phrases) [43, 47].

In a second step, the segments in the parts identified in the first step are classified based on their argumentative roles: claim or evidence. This is typically achieved by identifying the discourse relations between segments. Recently, two directions have been followed in this regard. Feng and Hirst [22] proposed a discourse parser that relies on conditional random fields to label sets of neighbouring segments and derive a tree of their relations. A different angle is taken by Cabrio and Villata [9], who adopt textual entailment [11] to identify and extract arguments.

Since automatic discovery of arguments is concerned with the intrinsic nature of the processed text and largely independent of extrinsic factors, such as readers' beliefs or background knowledge, it can be evaluated in a rather objective manner. To this end, a set of arguments is obtained from a group of 3-5 expert users and serves as ground truth, see [43, 47]. Then, the precision (ratio of correctly identified arguments and all identified arguments) and recall (ratio of correctly identified arguments and all correct arguments) of a discovery technique is assessed. Despite the above mentioned research efforts, however, fully automatic argument discovery is error-prone. State-of-the-art systems reach precision and recall levels of around 60% [9] and are considered to be incapable of extracting arguments with complicated structures.

The Quest for Semi-Automated Argument Discovery. Since fully manual processing does not scale in a Web setting and fully automated processing fails to comprehensively address the challenges stemming from unstructured textual representations of arguments, we advocate a semi-automated approach. It promises to combine the best of both worlds—high recall and precision in the discovery of arguments as achieved by manual processing and the low costs and the scalability of automatic processing.

To mediate the inherent trade-off between manual argument extraction and automatic text processing, semi-automated argument discovery has to address two core requirements:

(R1) *Efficient manual processing.* Human resources are scarce and induce relatively high costs. Hence, semi-automated argument discovery shall strive for maximizing the benefit of user input.

(R2) *Robustness against heterogeneous input.* Results of manual processing may be inconsistent or contradictory. Therefore, sensible argument discovery shall be robust in the presence of heterogeneous user input.

3. PROBLEM AND APPROACH

This section first presents a model and defines the problem of argument discovery involving user input. Then, we outline our semi-automated approach to solve the problem.

3.1 Model and Problem Formulation

The setting for argument discovery is defined by a set of *documents*. A document contains *paragraphs*, which are built of *segments*, i.e., sentences or clauses. The set of paragraphs in all documents is denoted by $\mathbb{P}$ and the set of all segments of all paragraphs is $\mathbb{S}$. The set of segments of a paragraph p is denoted by $seg(p) \subseteq \mathbb{S}$. An overview of the used notation is given in Table 1.

Table 1: Overview of notations

Notation	Description
$\mathbb{P}$	The set of all paragraphs of all documents
$\mathbb{S}$	The set of all segments of paragraphs in $\mathbb{P}$
$seg : \mathbb{P} \to \wp(\mathbb{S})$	The assignment of segments to a paragraph
$\Omega_{\mathbb{P}}$	The set of all arguments of paragraphs in $\mathbb{P}$
$\omega : \mathbb{P} \to \Omega_{\mathbb{P}}$	The assignment of arguments to paragraphs
$claim : \Omega_{\mathbb{P}} \to \mathbb{S}$	The assignment of a claim to an argument
$evid : \Omega_{\mathbb{P}} \to \wp(\mathbb{S})$	The assignment of evidence to an argument
$\Sigma_{\mathbb{P}}$	The set of all *discovered* arguments of paragraphs in $\mathbb{P}$
$\sigma : \mathbb{P} \to \Sigma_{\mathbb{P}}$	The assignment of *discovered* arguments to paragraphs

We ground our work in a model in which a paragraph p may contain a single *argument* $a = \langle c, V \rangle$, also denoted by $V \rightsquigarrow c$, built of a claim $c \in seg(p)$ and evidence segments $V \subset seg(p)$, which are also denoted by $claim(a) = c$ and $evid(a) = V$, respectively. One may consider a model in which a paragraph can contain multiple arguments. However, as exemplified above with Example 1, it may be impossible to disentangle the arguments since evidence may refer only to a set of claims as a whole. Therefore, our model incorporates the assumption that the notion of a paragraph defines the granularity of the argumentative structure of a document.

To deal with relations between arguments, we split an argument $V \rightsquigarrow c$ into *atomic arguments* $\{v_i\} \rightsquigarrow c$ with $V = \{v_1, \ldots, v_k\}$ and $1 \leq i \leq k$. Two arguments a_1 and a_2 are *equal*, $a_1 = a_2$, iff $claim(a_1) = claim(a_2)$ and $evid(a_1) = evid(a_2)$. They are *conflicting*, $a_1 \perp a_2$, iff $claim(a_1) \neq claim(a_2)$. These relations are defined on the level of segments, i.e., they are purely syntactic and do not relate to the segment semantics, which may be partially overlapping. Further, $\Omega_{\mathbb{P}}$ is the set of all arguments of all paragraphs $\mathbb{P}$ and the assignment of an argument to a paragraph is given as $\omega : \mathbb{P} \to \Omega_{\mathbb{P}}$.

In argument discovery, ω and $\Omega_{\mathbb{P}}$ are not known and shall be approximated by means of a discovery technique. Given such a technique, we denote the set of atomic arguments that are actually identified for all paragraphs $\mathbb{P}$ by $\Sigma_{\mathbb{P}}$ and the respective assignment of a discovered argument to a paragraph by $\sigma : \mathbb{P} \to \Sigma_{\mathbb{P}}$.

The performance of argument discovery is assessed based on the relation of ω and σ using an evaluation function f that maps the assignments of actual and discovered arguments into the unit interval $[0, 1]$. Common evaluation functions are precision and recall.

In semi-automated argument discovery, the amount of user input needs to be constrained by an effort budget b. It defines the number of paragraphs for which a single user is asked to provide feedback on the contained arguments.

Based on the above model, we define the problem of argument discovery under a given effort budget as follows.

PROBLEM 1 (BUDGET-LIMITED ARGUMENT DISCOVERY). *Given a set of paragraphs $\mathbb{P}$, an effort budget b, and an evaluation function f, budget-limited argument discovery refers to the identification of an assignment of arguments σ, such that $f(\omega, \sigma)$ is maximal while seeking a single user's input for at most b paragraphs.*

As mentioned above, ω is unknown. For evaluation purposes, thus, the set of actual arguments needs to be obtained from expert users. For practical reasons, this is done for a subset $\mathbb{P}' \subset \mathbb{P}$ and the evaluation is based on a partial assignment ω' with $\omega'(p) = \omega(p)$, $p \in \mathbb{P}'$.

3.2 Approach

The Argument Discovery Process. Given a set of documents, our approach to argument discovery, illustrated in Figure 1, enables efficient and robust integration of manual and automatic processing.

To realise efficient manual processing (R1), we employ guided crowdsourcing, which involves *(I) ranking of paragraphs* and *(II) argument crowdsourcing* for selected paragraphs. Documents are split into paragraphs and a scoring model is used to rank them based

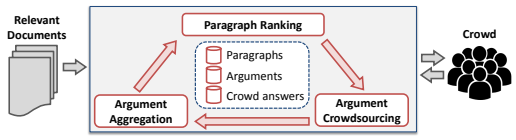

Figure 1: Overview of the approach

on their likelihood to contain an argument. Crowdsourcing can generally be seen as a means to scale manual processing to large-scale data. We further improve processing efficiency by exploiting crowdsourcing only for selected paragraphs and by implementing an adaptive strategy to post crowd tasks, which dynamically determines the amount of required user input per paragraph.

To obtain a single trusted set of arguments in the presence of uncertain, potentially inconsistent user input, our approach features a third step, *(III) argument aggregation*. Using a probabilistic model, we resolve conflicts in the answers and identify the arguments that best represent a paragraph, thereby addressing the need for robustness against heterogeneous user input (R2).

Our approach integrates the three steps (I)-(III) in an *iterative learning process*. The algorithm iteratively ranks paragraphs, crowdsources arguments, and aggregates the answers, until the effort budget has been spent or all paragraphs have been processed.

Crowdsourcing of arguments should involve several workers per paragraph to leverage the *wisdom of the crowd*. However, for some paragraphs, argument discovery is arguably harder than for others, so that more input shall be sought for them. To dynamically adapt the amount of user input per paragraph, one may argue that a single paragraph should be handled in each iteration of the process. This approach, however, would have limited potential for assessing the worker reliability and, thus, answer quality. Such an assessment becomes effective only when a worker handles multiple paragraphs.

To mediate this trade-off, our argument discovery process proceeds in rounds of batches. In each iteration, a set of selected paragraphs is handled by a single worker. The selection of paragraphs for a process iteration is governed by two aspects: a paragraph from the previous iteration is kept, if the respective arguments aggregation is considered to be uncertain; otherwise it is replaced by the top-ranked paragraph that has not yet been processed.

Algorithm. Our iterative learning process is defined in Algorithm 1. Given a set of paragraphs $\mathbb{P}$ and an effort budget b, it returns the assignment of discovered arguments to paragraphs. The algorithm first initialises the result function, the scoring model, and the certainty model (lines 1 to 3). It also determines the batch size—the number of paragraphs per crowd worker (line 4). Following studies on the effectiveness of crowdsourcing [29], the default batch size is set to 10 and we explore different values in our evaluation. The algorithm iteratively builds up the result, terminating when the effort budget has been spent or all paragraphs have been processed. In each iteration (lines 7 to 21), *open* holds the set of paragraphs that are processed in three steps:

(I) Paragraph ranking, detailed in Section 4, constructs a scoring model (*score*) to assess the likelihood of a paragraph to contain an argument (line 8). The model is trained with arguments discovered in previous iterations of the algorithm. Next, the set of current paragraphs (*open*) is extended with (not yet processed) top-ranked paragraphs to fill up the batch of size q (line 10). Initially, paragraphs are selected based on their usefulness to train the scoring model, whereas later, the actual scores determine the ranking.

(II) Argument crowdsourcing, discussed in Section 5, mines candidate claims and candidate evidence from the selected paragraphs (set *open*), creates crowd tasks, posts them to a crowdsourcing platform, and collects the crowd answers (line 11).

(III) Argument aggregation, described in Section 6, first updates

Algorithm 1: Iterative learning process for argument discovery

input : $\mathbb{P}$, a set of paragraphs; b, an effort budget.
output: σ, the assignment of discovered arguments to paragraphs.

1 $\sigma \leftarrow \emptyset$; // The assignment of discovered arguments to paragraphs
2 $score \leftarrow \emptyset$; // The scoring model for paragraphs
3 $cert \leftarrow \emptyset$; // The model for argument certainty
4 $q \leftarrow set\_batch\_size()$; // Determine batch size
5 $open \leftarrow \emptyset$; // The set of paragraphs handled in each iteration
6 $old \leftarrow \emptyset$; // The set of paragraphs processed already
7 **repeat**
  // (I) Paragraph ranking
8 $score \leftarrow construct\_score(score, \sigma)$; // Construct scoring model
9 **while** $|open| < q$ **do**
10 $open \leftarrow open \cup select\_next\_best(score, old)$; // Add paragraphs
  // (II) Argument crowdsourcing
11 $ans \leftarrow crowdsourcing(open)$; // Actual crowdsourcing
  // (III) Argument aggregation
12 $cert \leftarrow update\_cert(cert, ans)$; // Update argument certainty
13 $next \leftarrow \emptyset$;
14 **for** $p \in open$ **do**
    // If needed, re-post paragraph in next iteration
15 **if** $repost\_paragraph(p, cert)$ **then** $next \leftarrow next \cup \{p\}$;
    // Else, record processing, extract aggregated argument
16 **else**
17 $old \leftarrow old \cup \{p\}$;
      // If answers are certain, instantiate argument
18 **if** $certain\_answers(p, cert)$ **then** $\sigma \leftarrow \sigma \cup ins\_arg(cert, p)$;
19 $open \leftarrow next$;
20 $b \leftarrow b - |ans|$; // Reduce budget by effort spent
21 **until** $|open| = 0 \vee b = 0$;
22 return σ;

a probabilistic model (a factor graph, *cert*) that captures the relations between workers, arguments, and answers (line 12). For each of the currently processed paragraphs (set *open*), we then evaluate whether the paragraph is re-posted in the next iteration of the algorithm (line 15), using the factor graph (*cert*). If the paragraph is not re-posted, we record it as processed (line 17) and, if the answers are certain, include the respective assignment in the result (line 18).

4. PARAGRAPH RANKING

To rank paragraphs for argument crowdsourcing, Section 4.1 first presents a feature-based model for paragraphs. Then, Section 4.2 focuses on the question of how to determine whether a paragraph contains arguments and introduces the construction of a scoring model (function *construct_score* in Algorithm 1). Finally, Section 4.3 presents two strategies for the selection of paragraphs, i.e., function *select_next_best* in Algorithm 1.

4.1 Paragraph Modeling

Feature selection. To assess the likelihood that a paragraph contains an argument, we need to capture characteristics of paragraphs that hint at arguments. Since there is a very large set of feature candidates, see work on text classification [49], we conduced a preliminary study to identify features that hint at arguments. The details about this study along with a comprehensive overview of the derived features can be found in Appendix A. Below, we summarize the main results of this study.

Our study revealed *lexical* features related to the language vocabulary and *syntactical* features that refer to the text structure. Lexical features include the frequency of thematic words, the number of evidence-related words (e.g., numbers or citations), and the relative frequency of prototypical words (expressions that formulate arguments). As syntactical features, occurrence counts for thematic words appearing as (1) an subject or object (determined by part-of-speech tagging) or (2) a head word (root of the parse tree of a sentence) turned out to have large discriminative power.

Paragraph model. Each individual feature provides an estimation of the likelihood that the paragraph contains an argument. Hence, we model a paragraph $p \in \mathbb{P}$ as a normalized, s-dimensional feature vector $\vec{v}_p = \langle f_1, ..., f_s \rangle$ with $f_i \in [0,1]$ as the score for the i-th feature. All scores are normalized into the unit interval by dividing them by the maximum value observed among all paragraphs.

For each paragraph $p \in \mathbb{P}$, our model further includes a labeling function $\alpha : \mathbb{P} \rightarrow \{1, -1\}$, where $\alpha(p) = 1$ if p contains an argument and $\alpha(p) = -1$, otherwise. Initially, this labeling function is not defined for any paragraph. When executing the iterative learning process for argument discovery, however, it is updated based on the argument assignment (σ in Algorithm 1). It holds $\alpha(p) = 1$ if $p \in dom(\sigma)$, and $\alpha(p) = -1$, otherwise.

4.2 Paragraph Scoring

Scoring model. Based on the above features, we employ a scoring model to assess the likelihood that a paragraph contains an argument. This model is a function *score*, which takes the feature vector $\vec{v}_p$ of a paragraph $p \in \mathbb{P}$ as input and returns a value $score(\vec{v}_p) \in [0,1]$, or short $score(p)$. To summarize the individual features, we use weighted aggregation. Given a feature vector $\vec{v}_p = \langle f_1, \ldots, f_s \rangle$, the scoring model is defined as $score(p) = \vec{w}\vec{v}_p$, with $\vec{w} = \langle w_1, \ldots, w_s \rangle$ as a weight vector ($\sum_{1 \leq i \leq s} w_i = 1$) indicating the significance of the individual features. Further, $\beta(p) = score(p) - 0.5$ denotes the prediction made by the scoring model for p, so that $\beta(p) > 0$ means that p contains an argument.

Feature weights. The weight parameters of the scoring model are set by an active learning strategy. In each iteration of the argument discovery process, function *construct_score* adjusts the scores based on the arguments crowdsourced in the previous iteration. To this end, we use the Margin Infused Ranking Algorithm (MIRA) [10].

MIRA is based on the notion of a margin for paragraph p, defined as $m(p) = \alpha(p)\beta(p)$ based on the labeling function α and the prediction β made by the scoring model. The margin measures how good or bad the prediction has been for paragraph p. A positive margin indicates a correct prediction by the scoring model. In case of a negative margin, the model suffers a loss, defined as:

$$loss(m(p)) = \begin{cases} 0 & m(p) \geq 0 \\ 1 - m(p) & otherwise \end{cases} \quad (1)$$

The intuition behind the above equation is summarized as follows. When the model makes a correct prediction ($m(p) \geq 0$), the model is not changed ($loss(m(p)) = 0$). In case of an incorrect prediction, the update to the model is the smallest change needed to incorporate the new label ($loss(m(p)) = 1 - m(p)$).

The weights of the scoring model are updated iteratively as follows. Let $\vec{w}_t$ be the weight vector in the t-iteration of the process and let p be the paragraph, for which the argument assignment has changed. Then, the new weights are defined as

$$\vec{w}_{t+1} = \vec{w}_t + \frac{loss(m(p))}{||\vec{v}_p||^2} \alpha(p) \vec{v}_p. \quad (2)$$

This update of weights is grounded in the definition of the *loss* function. A correct prediction by the scoring model ($loss = 0$) does not change any weight. Incorrect predictions, in turn, lead to greedy modifications that are just large enough to cover the misclassified paragraph. Factor $\alpha(p)$ controls the direction of the model change (by taking the value -1 or 1), while $||\vec{v}_p||^2$ normalizes the *loss* value.

4.3 Selection of Paragraphs

Our approach includes two strategies for the selection of paragraphs, i.e., function *select_next_best* in Algorithm 1 has two implementations. An *uncertainy-based* strategy selects paragraphs based on their usefulness to train the scoring model. A *score-based* strategy exploits the scoring model to assess the likelihood that a paragraph contains an argument. Below, we first define the strategies before discussing when to use which strategy.

Uncertainty-based selection strategy. This strategy selects paragraphs, for which argument crowdsourcing is considered to be useful for improving the scoring model. We capture this usefulness of a paragraph with an information-theoretic model and apply uncertainty sampling [53] to chose the paragraph, for which it is least certain whether it contains an argument. Technically, the uncertainty related to a paragraph p, with $\vec{v}_p = \langle f_1, \ldots, f_s \rangle$ being its feature vector, is measured by the Shannon entropy:

$$H(p) = -\sum_{1 \leq i \leq s} f_i \log(f_i) \quad (3)$$

Then, the uncertainty-based strategy, denoted by $select\_next\_best_U$, selects the paragraph with the highest entropy that has not yet been processed (set *old* in Algorithm 1):

$$select\_next\_best_U(score, old) = \underset{p \in (\mathbb{P} \setminus old)}{\arg\max} \; H(p) \quad (4)$$

Scoring-based selection strategy. This strategy uses the scoring model to identify paragraphs that are likely to contain an argument. The selection function, denoted by $select\_next\_best_S$, is defined as:

$$select\_next\_best_S(score, old) = \underset{p \in (\mathbb{P} \setminus old)}{\arg\max} \; score(p) \quad (5)$$

Choosing a selection strategy. There is a trade-off between the application of the uncertainty-based strategy and the scoring-based strategy. Focusing solely on the former may lead to overconsumption of the budget for training the scoring model, without making effective use of the model. Excessive usage of the scoring-based approach, however, is undesirable without a proper training phase. Hence, we propose a mechanism to transition from the uncertainty-based strategy, used initially, to the scoring-based strategy.

Intuitively, we switch to the scoring-based strategy when the scoring model is stable. We define the model stability as the number of correct predictions made. Formally, in each iteration of the argument discovery process, we check whether all paragraphs p, for which we obtained a crowdsourcing result in the last iteration, have a positive margin, $m(p) > 0$. If so, all predictions in the last iteration have been correct. After a pre-defined number of iterations with correct predictions, the scoring-based strategy is applied.

Selection roboustness. Independent of the applied strategy, paragraph selection may be subject to a bias often encountered in the Web—replicated data [38]. Thus, we take the following measures:

- *Duplicate detection.* To avoid redundancies in the set of paragraphs used for argument discovery, pre-processing removes duplicate documents. To this end, we quantify document similarity using standard measures, e.g., the Jaccard coefficient over 3-grams [56], and filter documents by a similarity threshold.
- *Data source diversification.* The above selection strategies may consider only paragraphs from a small number of data sources. To not miss out argumentative information, we employ result diversification [18] to re-rank a selection of paragraphs.

5. ARGUMENT CROWDSOURCING

Having discussed the ranking of paragraphs, we turn to the actual argument crowdsourcing (function *crowdsourcing* in Algorithm 1). We first discuss how to design the questions that form the tasks to be posted to crowd workers (Section 5.1). Then, we show how to instantiate these questions with candidate claims and evidence that are mined from a paragraph (Section 5.2).

5.1 The Design of Questions

Good task design is crucial to obtain crowdsourcing results of high quality. In addition to infrastructure aspects such as a user-friendly interface [68], several requirements for the design of crowdsourcing tasks have been identified. Tasks shall have (I) a low complexity [21], (II) be self-reporting, and verifiable [33], and (III) include abundant context information [36]. Below, we show how to design questions in our context to satisfy these requirements.

Two types of questions can be used to identify arguments in terms of their claim and evidence: *open* and *closed* questions. In our setting, workers may directly be asked to identify arguments in a paragraph (open question). However, since crowd workers are assumed to strive for maximal profit and thus avoid relatively complex tasks, we opt for closed questions that limit complexity by pre-defined answer options. Also, the answers to closed questions are easy to process and rather robust to invalid answers. Posting closed questions, however, requires mining of candidate arguments and evidence from the paragraphs. Below, we first discuss the general structure of questions, before turning to the mining step.

Question for claims. A paragraph can contain many candidate arguments and, thus, many candidate claims. A question posted to crowd workers shall cover all of these candidate claims and workers shall be able to indicate that a paragraph does not contain any claim.

Question for evidence. We could follow a similar approach for questions for evidence and construct a multiple choice question with each answer being possible evidence. However, this approach is not feasible in practice, since workers tend to select only one answer, even if multiple answers are correct [3]. The reason is, again, the tendency to maximize profit by spending as little time as possible per question. Therefore, we employ several Boolean questions, verifying each candidate evidence separately. Our evaluation later shows that, indeed, this approach is significantly more effective than using a single multiple choice question.

EXAMPLE 2. *We take up Example 1. While we later show how to mine candidate claims and evidence, here, we assume that segments* $\{S1, S2, S5\}$ *have already been tagged as candidate claims, while segments* $\{S2, S3, S4\}$ *are potential evidence. Figures 2 and 3 illustrate the respective questions for the claim and evidence.*

Following this approach, for each paragraph, a set of questions is generated: one for the claim and one for each possible evidence. These questions are combined into a task to be posted on a crowdsourcing platform. The feedback obtained from the crowd workers for such a task is a set of candidate atomic arguments.

EXAMPLE 3. *For our running example, suppose a worker selects* S_1 *as the claim and confirms* S_2 *and* S_3 *as evidence. This results in a candidate argument* $\{S_2, S_3\} \rightsquigarrow S_1$*, which is split up into two atomic arguments,* $S_2 \rightsquigarrow S_1$ *and* $S_3 \rightsquigarrow S_1$.

Effectiveness of question design. Relating back to the requirements for the design of crowdsourcing tasks, our approach yields tasks that provide abundant context information, see [36], since they include paragraphs instead of single sentences. They are of low complexity, see [21], as they rely on closed questions, separately for claims and evidence. Finally, our questions support self-report and verification, as put forward in [33]. That is, segments of a paragraph are potentially included in the question for a claim and in the questions regarding the evidence. If so, the reliability of a worker can be assessed based on the absence of conflicting answers that would identify a segment as both, a claim and its supporting evidence. In our experimental evaluation, we show that this task design is, indeed, much more effective than alternative options.

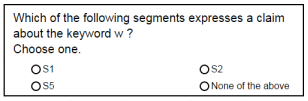

Figure 2: Claim question

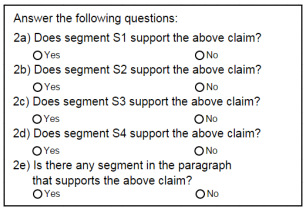

Figure 3: Evidence questions

5.2 Mining Candidate Claims & Evidence

Constructing questions for crowdsourcing requires identifying candidate claims and evidence in a paragraph. To this end, we first split paragraphs into smaller meaningful parts, i.e., segments, by means of sentence boundary disambiguation and discourse segmentation [22, 41]. For evidence that is supposedly objective and verifiable, we also exploit fact extraction [6].

Discourse relation extraction. Adjacent segments in a paragraph are not independent, but stand in a semantic relation to one another. By discovering these relations, we can identify which segment is the claim or evidence. Specifically, we exploit the Rhetorical Structure Theory (RST) [22] that defines 23 rhetorical relations for pairs of segments. Many of these relations signify the presence of an argument, i.e., *explanation, background, contrast, condition, evaluation* and *evidence*. The RST defines different roles for the segments in most of the relations. One segment (called *nucleus*) is the central one, whereas the other (*satellite*) has a supportive role. Depending on the relation, the nucleus is usually the claim, whereas the satellite is the evidence.

EXAMPLE 4. *Sentence boundary disambiguation and discourse segmentation split up the paragraph in Example 1 into five segments* $S_1, \ldots, S_5$*. An example for the discourse relations is the* contrast *relation between segment* S_2 *and* S_3*, which is identified based on the linking word 'but'.*

Augmenting the relations. To mine arguments, we further enrich the relations identified by the RST with the following features:

- *Position:* We include the position of a segment in the paragraph. Segments at the beginning of a paragraph could be part of a topic sentence, which is likely to be a claim.
- *Presence of keyword:* A claim often mentions thematic words. As a result, the presence of these thematic words in a segment may hint at a claim.
- *Presence of numbers:* The presence of numbers may hint at evidence that is used to support a claim.
- *Cue phrases:* Cue phrases, such as 'finding' and 'study', indicate that a segment is evidence. We extracted such cue phrases in the aforementioned preliminary study, see Appendix A.

Discourse relations along with the above features are captured by a set of predicates over single segments, as summarized in Table 2.

Rule-based mining. The predicates are employed to extract candidate claims and evidence by means of the following rules:

Relation(S) ∧ Satellite(S) ∧ Number(S)	→	Evidence(S)
Relation(S) ∧ Satellite(S) ∧ Phrase(S)	→	Evidence(S)
Relation(S) ∧ Satellite(S) ∧ Middle(S)	→	Evidence(S)
Relation(S) ∧ Nucleus(S) ∧ Top(S)	→	Claim(S)
Keyword(S)	→	Claim(S)

Using the above rules, a set of candidate claims and evidence is acquired to instantiate the questions of the crowdsourcing tasks.

EXAMPLE 5. *Using the above rules, examples for classifications of segments in Example 1 are segment* S_1*, which is likely to be a claim as it contains the thematic word 'processed', and segment* S_3*, which may be evidence because of the presence of numbers and its role as a satellite part of a contrast relation.*

Table 2: Predicates used to mine arguments

Predicate	Description
Relation(S)	S is in a relation such as contrast, explanation etc.
Nucleus(S)	S is a nucleus
Satellite(S)	S is a satellite
Top(S)	S is in the beginning of the paragraph
Middle(S)	S is in the middle of the paragraph
Number(S)	S contains numbers
Phrase(S)	S contains cue phrases
Evidence(S)	S is an evidence
Claim(S)	S is a claim
Keyword(S)	S contains keyword or part of it

6. ARGUMENT AGGREGATION

The aggregation of crowdsourced candidate arguments as part of our iterative learning process for argument discovery (Algorithm 1) includes: the creation of a certainty model (function *update_cert*); a mechanism to decide whether a paragraph shall be re-posted (predicate *repost_paragraph*); a check whether the answers are certain (predicate *certain_answers*); and a way to instantiate an argument for a paragraph from the certainty model (function *ins_arg*).

Below, we show how to create a factor graph as a model of argument certainty (Section 6.1) and how it is used to compute the certainty of an argument (Section 6.2). Then, we introduce methods to take a decision about re-posting a paragraph (Section 6.3) and to instantiate an argument from the certainty model (Section 6.4).

6.1 A Model for Argument Certainty

Crowd answer matrix. In each iteration of the argument discovery process, the answers obtained by crowdsourcing are modeled as a matrix over $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{P}$, the set of all paragraphs crowdsourced so far, and $W = \{w_1, \ldots, w_m\}$, the set of all workers that provided input. For this representation of the answers, we split up each argument $V \rightsquigarrow c$ provided by a worker into its atomic arguments $\{v_i\} \rightsquigarrow c$ with $V = \{v_1, \ldots, v_k\}$ and $1 \leq i \leq k$. Let $L_p = \{\{v\} \rightsquigarrow c \mid v \in seg(p) \wedge c \in seg(p)\}$ be the set of all atomic arguments that can be constructed from the segments of $p \in P$. Then, crowd answers are modeled as an $n \times m$ *answer matrix*:

$$M = \begin{pmatrix} l_{p_1 w_1} & \ldots & l_{p_1 w_m} \\ \ldots & \ldots & \ldots \\ l_{p_n w_1} & \ldots & l_{p_n w_m} \end{pmatrix} \tag{6}$$

where either $l_{pw} \subseteq L_p$ are the atomic arguments provided by worker $w \in W$ for paragraph $p \in P$ or $l_{pw} = \ominus$ denotes the absence of an answer for paragraph p by worker w. The latter distinguishes the absence of an answer from the empty response ($l_{pw} = \emptyset$, worker w replied that paragraph p does not contain any argument). We define $A_p = \bigcup_{w \in W} l_{pw}$ as the set of atomic arguments obtained for paragraph p and $A = \bigcup_{p \in P} A_p$ as the set of all candidate arguments. Finally, set Z contains an *atomic answer* $z_{pwa} \in \{1, 0, \emptyset\}$ for each worker $w \in W$ and argument $a \in A$ in paragraph $p \in P$, such that

- $z_{pwa} = 1$, if $a \in l_{pw}$, the worker found the argument,
- $z_{pwa} = 0$, if $a \notin l_{pw}$, the worker did not find the argument,
- $z_{pwa} = \emptyset$, if $l_{pw} = \ominus$, the worker did not process the paragraph.

Motivation for a probabilistic model. Aggregating arguments is more challenging than traditional aggregation of crowd answers, see [30]: (1) the answer matrix comprises partial functions, instead of discrete values; (2) there is a mutual reinforcing relation between workers and arguments (worker can provide multiple arguments, an argument can be provided by multiple workers); and (3) there are dependencies between candidate arguments of a paragraph, e.g., only one of two conflicting arguments shall be chosen.

Against this background, existing deterministic algorithms for aggregating crowd answers [12, 30] are inapplicable in our context. To cope with partial functions and capture the complex relations between arguments, we leverage a probabilistic graphical model, namely a *factor graph* [35]. It enables us to establish a relation between functions that are defined over potentially overlapping sets of random variables. Using probabilistic techniques, we can then compute the certainty of argument assignments for a paragraph, while taking into account the reliability of the workers and the correctness of their answers. Also, the model enables self-configuration when new information becomes available. Combined with active learning, the factor graph model allows us to handle the active nature of data generation in crowdsourcing, see also [26]. With the arrival of new crowd answers, the model is updated incrementally by adding variables and factors.

Creation of the factor graph. A factor graph is a bipartite graph $\langle V, F, E \rangle$ where V is a set of random variables, F is a set of functions (factors), and $E \subseteq \{\{v, f\} \mid v \in V, f \in F\}$ are undirected edges. A set of random variables V and a set of factors F fully characterises a factor graph. The definition of the edges relates each factor $f(v_1, \ldots, v_d) \in F$ to the random variables over which it is defined, i.e., $\{f, v_i\} \in E$ for $v_i \in V$, $1 \leq i \leq d$.

In our context, there are three types of random variables representing workers, arguments, and answers. We overload notation and use W, A, and Z to refer to the actual workers, arguments, and atomic answers, as well as the associated random variables, i.e., $V = W \cup A \cup Z$. Further, the model includes worker factors f_W, argument factors f_A, and answer factors f_Z to represent the relations between these variables, i.e., $F = f_W \cup f_A \cup f_Z$.

Worker variables. Each worker $w \in W$ is associated with a random variable, which, overloading notation, is denoted by $w \in [0, 1]$ indicating the reliability of the worker (higher is more reliable).

Argument variables. Each atomic argument $a \in A$ is associated with a variable $a \in \{0, 1\}$ indicating the correctness of the argument (1 denotes correctness, whereas 0 represents incorrectness).

Answer variables. Each atomic answer $z_{pwa} \in Z$ is also directly considered as an (observed) variable.

Worker factors. Each worker variable w is associated with a prior-distribution factor $f_w : \{w\} \rightarrow [0, 1]$ that is determined either in a training phase or stems from external sources such as the crowdsourcing service provider. If no information is available, we start with $f_w(w) = 0.5$ following the maximum entropy principle. The set of worker factors is defined as $f_W = \bigcup_{w \in W} f_w$.

Argument factors. Each set of atomic arguments $A_p = \{a_1, \ldots, a_k\}$ of a paragraph $p \in P$ is assigned an argument factor $f_{A_p} : A_p \rightarrow [0, 1]$ that captures the following relation between the arguments: (1) a paragraph cannot be assigned conflicting arguments; (2) if all arguments related to the paragraph are labeled as incorrect, there is no information that allows for conclusions on the argument for the paragraph (maximum entropy principle); and (3) there should be at least one argument (which may be the empty argument) assigned to each paragraph. We define the argument factor f_{A_p} as:

$$f_{A_p}(a_1, \ldots, a_k) \quad = \quad \begin{cases} 0 & \exists\, i, j \in \{1, \ldots, k\} : \\ & a_i = 1 \wedge a_j = 1 \wedge a_i \perp a_j \\ 0.5 & \forall\, i \in \{1, \ldots, k\} : a_i = 0 \\ 0.75 & \text{otherwise} \end{cases} \tag{7}$$

The intuition behind f_{A_p} is that (1) if there are correct arguments ($a_i = 1 \wedge a_j = 1$) that are conflicting ($a_i \perp a_j$), the factor is 0, which indicates impossibility; (2) in the absence of an argument labeled as correct, the factor of 0.5 indicates no argument preference; (3) to reinforce the existence of at least one argument assignment per paragraph, a factor of 0.75 to equally distribute the possibilities is used. The set of argument factors is defined as $f_A = \bigcup_{p \in P} f_{A_p}$.

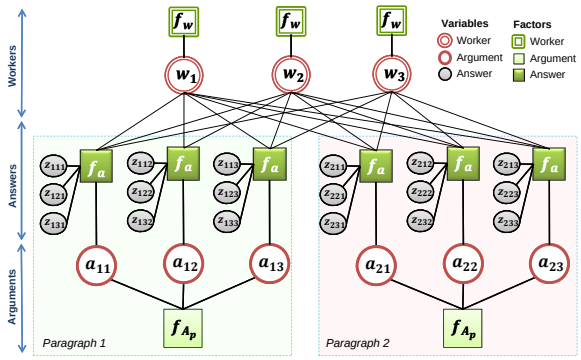

Figure 4: An example of factor graph

Answer factors. Each atomic argument $a \in A$ is assigned an answer factor $f_a : W \times Z \times \{a\} \rightarrow [0,1]$ that captures the relation between the argument, its related answers and all workers. The idea behind this factor is that there is a mutual reinforcing relation between workers and arguments via the answers. Answers from reliable workers have a higher weight compared to answer from those that are unreliable, whereas correct answers indicate high reliability. For an atomic argument $a \in A$, we model this relation as:

$$f_a(w_1,\ldots,w_m,z_{w_1pa},\ldots,z_{w_mpa},a) = \prod_{j\in\{1,\ldots,m\}|z_{w_jpa}\neq\emptyset} \left(|z_{w_jpa}-a| + (-1)^{|z_{w_jpa}-a|} w_j\right) \quad (8)$$

The model of answer factors, intuitively, incorporates the assumption that reliable workers give correct answers, whereas unreliable workers give incorrect answers. Consequently, the factor equals the worker reliability (w), when the worker answers correctly ($z_{wpa} = a$), whereas the probability of the worker giving incorrect answers ($1-w$) is used in case of an incorrect answer ($z_{wpa} \neq a$). Finally, a product aggregates the likelihoods over the answers of all workers.

EXAMPLE 6. *Figure 4 illustrates the factor graph for the case of two paragraphs and three workers. Each paragraph has three possible atomic arguments. Variables (circles) are linked to their respective factors (squares). There are two types of variables: latent variables (white circles) and observed variables (gray circles).*

6.2 Computation of Argument Certainty

A factor graph enables us to compute the certainty of an argument that is assigned to a paragraph. This computation exploits the (marginal) probabilities of the random variables representing the reliability of workers, the correctness of an argument, and the answers. Since the reliability of a worker $w \in W$ is defined over the unit interval $[0,1]$, the probability of it assuming a certain value is given as a distribution function $Pr(w)$ over $[0,1]$. Argument variables are binary, so that $Pr(a=1)$ (or short $Pr(a)$) is the probability that an argument $a \in A$ is correct. Answer variables, in turn, are observed, which means that the probability of their observed value is 1, whereas any other value has a probability of 0. Probability computation is based on the correlations defined by the factor functions that relate the random variables to each other.

Probability computation. To compute probabilities in a factor graph, *belief propagation* or *sampling* are commonly used. Belief propagation considers the (un)certainty as information that is propagated through the factor graph, e.g., by message-passing algorithms [35]. However, these techniques tend to converge slowly if the graph is large and contains circles [67]. When using crowdsourcing for argument discovery, the number of variables grows quickly, resulting in large and dense factor graphs. Hence, we resort to *sampling* to find the most probable values of random variables Specifically, Gibbs sampling proved to be a highly efficient and effective mechanism for factor graphs [67].

In brief, given an answer set N, probability computation based on the factor graph yields a value for the correctness $Pr(a)$ per atomic argument $a \in A$ and a probability distribution $Pr(w)$ for the reliability per worker $w \in W$.

Argument certainty of a paragraph. Using the computed probability values, the certainty with which a set of atomic arguments is assigned to a paragraph is quantified. Technically, we capture the uncertainty related to a paragraph, i.e., the lack of certainty in the assignment for this paragraph. For a paragraph $p \in P$, the argument uncertainty is defined as the Shannon entropy over the random variables of the atomic arguments A_p, as identified by crowd workers:

$$uncert(p) = -\sum_{a\in A_p} Pr(a)\log Pr(a). \quad (9)$$

It holds that $uncert(p) \geq 0$. A value of $uncert(p) = 0$ means that all argument probabilities are equal to one or zero. In other words, for each atomic argument there is a clear conclusion on its correctness.

6.3 The Reposting Decision

Next, we focus on the decision to repost a paragraph in the argument discovery process (predicate *repost_paragraph* in Algorithm 1). Taking this decision is hard due to the bi-objective nature of the discovery process—while we aim at discovering a large number of arguments, we also strive to minimize the invested effort.

To decide whether a paragraph is reposted, we apply a conjunction of the following three conditions:

(c1) Uncertainty condition: A paragraph with low argument uncertainty (Equation 9) indicates that the crowd workers reached a consensus on the atomic arguments. Since it would be cost-ineffective to repost the respective paragraph, we define a threshold τ and do not repost paragraphs for which the uncertainty drops below it. The actual value of τ mediates the trade-off between the quality (precision) and the number (recall) of the discovered arguments and can be set using a sampling approach: A few workers are asked to answer questions for a set of paragraphs. For different values of τ, crowdsourcing is simulated using the obtained answers. Observing the trade-offs for total effort, argument uncertainty, and the probability of derived arguments, a suitable value for τ is found.

(c2) Support condition: A paragraph p may satisfy the uncertainty condition ($uncert(p) < \tau$) with only a few answers since the smaller the number of workers, the easier it is to reach consensus. To avoid situations in which the 'wisdom of the crowd' is not exploited, we define a minimum support threshold m_{min} and only stop reposting a paragraph if the number of answers exceeds it. Following recent crowdsourcing studies [48], we set $m_{min} = 5$.

(c3) Effort condition: A paragraph may comprise ambiguous argumentative structures, so that workers cannot reach consensus. Such a paragraph would be reposted continuously until its uncertainty is lower than τ. To avoid such situations, we define a maximum effort threshold m_{max} that serves as an upper bound for the number of reposting iterations. Based on [48], we set $m_{max} = 10$.

6.4 Argument Instantiation

If a paragraph is not reposted, we have to decide whether an argument shall be instantiated (predicate *certain_answers* in Algorithm 1). To this end, we exploit the above effort condition ($c3$). If it holds true, the paragraph is likely to comprise ambiguous structures and, thus, is not suited for argument discovery. In all other cases, an argument is instantiated for the respective paragraph.

Given the candidate atomic arguments A_p for a paragraph $p \in P$ and their probabilities $Pr(a)$ for $a \in A_p$, the instantiation of an aggregated argument (function *ins_arg* in Algorithm 1) yields an argument $\hat{a} = \hat{V} \rightsquigarrow \hat{c}$ defined in two steps:

Claim selection. For each candidate claim, we sum up the probabilities of their atomic arguments. The idea behind this aggregation is that a large number of probable evidence segments hints at the claim for the argument that represents the paragraph. Then, we select the claim with the highest aggregated probability for the instantiated argument (assuming that $A_p \neq \emptyset$):

$$\hat{c} = \underset{c \in \bigcup_{a \in A_p} \{claim(a)\}}{\arg\max} \sum_{a \in \{a' \in A_p | claim(a') = c\}} Pr(a) \quad (10)$$

Evidence extraction. Given the selected claim, we complete the argument by extracting evidence that (1) supports the claim and (2) is part of supposedly correct arguments. The latter is assessed using the probabilities $Pr(a)$ assigned to atomic arguments $a \in A_p$.

$$\hat{V} = \bigcup_{a \in \{a' \in A_p \mid claim(a') = \hat{c} \wedge Pr(a') > 0.5\}} \{evid(a)\} \quad (11)$$

EXAMPLE 7. *Consider the following candidate arguments and probabilities:* $S_2 \rightsquigarrow S_1$*:* 0.52*;* $S_3 \rightsquigarrow S_1$*:* 0.59*;* $S_4 \rightsquigarrow S_1$*:* 0.21*;* $S_3 \rightsquigarrow S_2$*:* 0.31*;* $S_4 \rightsquigarrow S_2$*:* 0.27*. This example features two candidate claims,* S_1 *and* S_2*. Due to the higher aggregated probability,* S_1 *is selected as the claim for the argument. To extract the related evidence, arguments having* S_1 *as claim and a probability* > 0.5 *are considered. Then, then the aggregated argument is* $\{S_2, S_3\} \rightsquigarrow S_1$.

7. EXPERIMENTAL EVALUATION

This section reports on an experimental evaluation of our approach to argument discovery. We first elaborate on the used experimental setup (Section 7.1), before evaluating the efficiency and effectiveness of the following aspects of our approach:

- The scoring model for paragraph selection (Section 7.2).
- The crowdsourcing task design and difficulty (Section 7.3).
- The method to mine candidate claims and evidence (Section 7.4).
- The probabilistic aggregation of arguments (Section 7.5).
- The task posting strategy (Section 7.6).
- The end-to-end process of argument extraction (Section 7.7).
- The real-world deployment costs (Section 7.8).

7.1 Experimental Setup

Datasets. Our experiments use five real-world document collections, and a referential answer set created by expert users.

Document collections: We consider Web documents for five popular topics, namely vaccine *(vacc)*, processed food *(food)*, genetically modified food *(gmo)*, death penalty *(penalty)*, and globalization *(global)*, which span the domains of health, society, and economics. For each topic, we collected the top 10,000 documents, mostly news entries or articles, returned by a search engine (Bing). Each document has been split up into paragraphs, see also Table 3.

Table 3: Document collections

Dataset	Domain	#Paragraphs
vacc	health	151637
food	health	132953
gmo	health	126831
penalty	society	106496
global	economics	97425

Referential answer set: To obtain a controlled evaluation environment, we followed related work on automatic argument discovery [43, 47] and constructed a full referential answer set as follows. For a sample of 500 documents (topic *food*), five expert users identified all paragraphs that contain arguments and constructed the correct arguments for these paragraphs as ground truth. To conduct this task, the experts received minimal training by means of example paragraphs with and without arguments. Their results have been aggregated using majority voting and we measured the agreement level by the Krippendorff's α [34]. According to [34], the obtained value of $\alpha = 0.81$ indicates reliable agreement among the experts.

The experts identified a set of 40 paragraphs containing arguments. We then randomly selected an additional 60 paragraphs without arguments. The joint set of 100 paragraphs has been posted to Amazon Mechanical Turk (AMT), such that each paragraph is handled by the same 30 workers and each worker annotates all the paragraphs. Having this full answer set, we simulated the application of our iterative learning process for argument discovery.

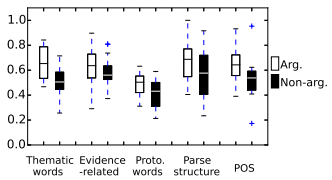

Figure 5: Discriminative capacity of model features

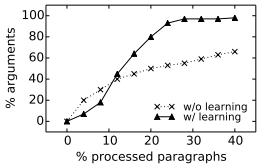

Figure 6: Effectiveness of paragraph selection method

Parameters. The *batch size q* determines the number of paragraphs that a worker needs to answer and is subject to a trade-off. A high value will allow for obtaining more high-quality arguments as argument instantiation and the construction of a stable paragraph scoring model benefit from additional answers. On the other hand, an overwhelming number of questions per worker has a negative influence on the answer quality. To mediate these aspects, we follow recent studies on crowdsourcing effectiveness [29], suggesting a task size of 10 questions, and also vary parameter q in the experiments.

To normalise the effort budget across datasets, we define the *budget ratio* $\beta = \frac{b}{|P|}$, where b is the budget and $\mathbb{P}$ is the set of all paragraphs. The rationale behind this measure is to enable comparison of experimental settings within the same relative budget constraint.

Metrics. We use the following evaluation measures:

Precision: is the ratio between the number of correct instantiated arguments and the number of all instantiated arguments.

Recall: is computed as the fraction of correct instantiated arguments over the total number of correct arguments in the dataset.

Both precision and recall are calculated on the basis of atomic arguments, e.g., instantiating argument $\{v_1\} \rightsquigarrow c$ when the correct argument is $\{v_1, v_2\} \rightsquigarrow c$ leads to a recall of 0.5.

Cost ratio: To have a normalized measure for the invested effort budget, the *cost ratio* γ captures the total number of worker answers relative to the number of processed paragraphs. It reflects the invested effort independent of the number of processed paragraphs.

Experimental environment. All results have been obtained on an Intel Core i7 system (3.4Ghz, 12GB RAM). Factor graph modeling and reasoning have been conducted using Elementary [67].

7.2 Effectiveness of Paragraph Selection

As discussed in Section 4.1, the paragraph model used for scoring paragraphs is based on lexical and syntactical features that have been extracted in a preliminary study. We validated this feature selection with the referential answer set obtained for the *processed food* topic, extracting for each paragraph whether it contains an argument (*Arg* or *Non-arg*, respectively). Figure 5 shows the obtained box plots for the feature scores for either group of paragraphs. For all features, the median scores are higher for paragraphs containing arguments, which indicates that the selected features are indeed suited to be used for paragraph scoring.

To evaluate the effectiveness of our paragraph selection method, we assessed how much of the effort budget has to be spent to extract the arguments of a collection of documents. Using the referential answer set, we compare our technique that learns from the feedback as part of the iterative argument extraction process (*w/ learning*) with a selection strategy that exploits a static version of the scoring model in which all features are equally weighted (*w/o learning*).

Figure 6 plots the obtained results in terms of the percentage of identified arguments relative to the percentage of paragraphs that have been processed. The strategy without learning performs better initially, when less than 10% of the paragraphs have been processed. The reason is that initially the *w/ learning* strategy will select paragraphs that are most beneficial for training the scoring model, which are not necessarily paragraphs that contain arguments. However, once the scoring model is stable, the learning-based approach performs far superior. With 40% of the paragraphs processed, the method without learning selected only around 60% of the paragraphs with arguments, whereas our approach retrieved most of the desired paragraphs (92%). This result demonstrates that the pro-active learning of our scoring model saves overall effort.

7.3 Crowdsourcing Task Design

Next, we evaluated the taken design choices for crowdsourcing tasks in terms of the *question design* and *task difficulty*.

Question design. To validate the decision to rely on a set of binary questions to identify candidate evidence, instead of asking a single open question, we explored the completeness of evidence when using both methods. Based on the referential answer set, we conducted crowdsourcing with an open question and assessed the frequency with which workers responded with a single evidence only. Figure 7 shows the results for paragraphs grouped by the actual number of evidence of the contained arguments. For instance, for the paragraphs containing an argument with three evidence parts, for 9 out of 11 paragraphs, workers identified only a single evidence when answering the question. These results highlight that workers strongly tend to answer open questions non-exhaustively, so that closed questions are preferred in this context.

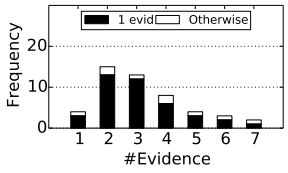

Figure 7: Issues of open questions

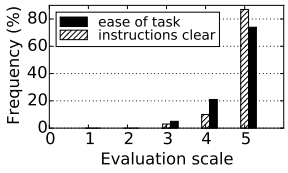

Figure 8: Perceived task difficulty

Task difficulty. A downside of closed questions in comparison to a single open one, is the overall increased number of questions. Therefore, we also checked whether the crowdsourcing tasks are still assessed to be easy and understandable by the workers. As part of the creation of the referential answer set, we asked crowd workers to judge the *ease of the task* and the *clearness of instructions* on a five-point Likert scale. Figure 8 shows that, indeed, the majority of workers agreed that the tasks have been easy to complete and have been equipped with clear instructions (no score value < 3).

7.4 Mining of Candidate Claims & Evidence

Mining claims. We first evaluated whether our claim extraction method covers the actual claims without including every possible text segment. The latter is important since consideration of all possible segments, each resulting in a possible answer, would be overwhelming for the workers. For this purely quantitative evaluation, knowledge about the actual arguments is not needed, so that we included all document collections mentioned above.

Figure 9 depicts the obtained results as the number of claims extracted by our technique (average, first and third quartile over all documents) relative to the number of segments in the paragraphs (large paragraphs with more than 20 segments are not visualized). We observe that the number of extracted claims is low and stable, meaning that it does not increase with the number of segments.

For the documents included in the referential answer set, we also evaluated the correctness of the extracted claims. For 87% of the paragraphs, the set of extracted claims included the correct one. Thus, our technique does not only return a small number of claims, but also extracts the correct ones in the vast majority of cases.

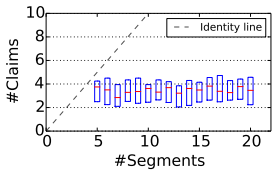

Figure 9: Claim mining

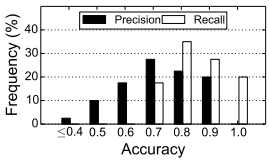

Figure 10: Evidence mining

Mining evidence. To evaluate the effectiveness of our method for evidence mining, we compare the evidence extracted for the referential answer set to the ground truth. As a result, for each paragraph containing an argument in the referential answer set, we obtained two values: one for precision and one for recall. The histogram in Figure 10 depicts the resulting precision and recall, where the bins are $(0,0.4],(0.4,0.5],(0.5,0.6],....$ We observe that our technique extracts evidence with high recall and moderate precision. While the moderate precision values highlight the need for crowdsourcing, high recall values indicate that the crowdsourcing tasks include the actual evidence in most cases.

7.5 Argument Aggregation

Effectiveness of probability computation. We studied the relation between probability computation and the correctness of the extracted arguments with an experiment that used the 10 highest-ranked paragraphs of the referential answer set. We executed the argument discovery process with a budget ratio of $\beta = 10$ and an uncertainty threshold of $\tau = 0.4$. After aggregating the collected answers, the derived probabilities were compared with the ground truth that labels atomic arguments as being correct or incorrect.

Figure 11 depicts the ratios of correct arguments and incorrect arguments for bins of probability values. For example, if the probability value lies within $[0.7,0.8)$, about 90% arguments are correct and 10% are incorrect. In contrast, for the $[0,0.1)$ bin, 92% of the arguments are incorrect. The obtained value distribution indicates that, even though there are exceptions, the derived probability values are generally well-correlated with the correctness of arguments.

Effects of batch size. The batch size parameter can be expected to influence the correctness of the obtained results. A large batch size means that multiple paragraphs are assessed by each worker and each paragraph is assessed by multiple workers, which creates mutual reinforcing dependencies between workers and the paragraphs, resulting in more accurate instantiation of arguments. We experimented with the referential answer set and executed the argument discovery process for varying batch sizes q, while keeping the budget ratio and uncertainty threshold constant, $\beta = 10$ and $\tau = 0.4$. The obtained results were compared to the ground truth.

Figure 12 shows that precision and recall increase with the applied batch size. This trend is due to the factor graph model being able to capture more relations between workers, paragraphs and answers for large batch size, yielding a higher result quality. We note, however, that large batch sizes also increase the budget needed to process a set of paragraphs—a trade-off that is evaluated below.

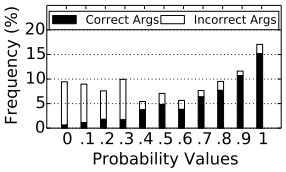

Figure 11: Effectiveness of probability computation

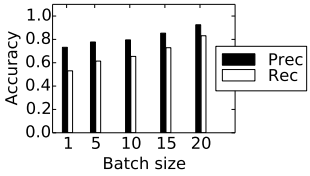

Figure 12: Effects of batch size

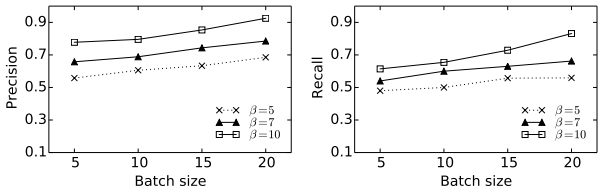

Figure 13: Effects of budget

7.6 Task Posting

Effects of budget. To shed light on the influence of the effort budget on the result quality, we conducted an experiment with the referential answer set and executed the argument discovery process for different budget ratios β and batch sizes q (fixing $\tau = 0.4$). Figure 13 shows the precision and recall of the instantiated arguments.

Both precision and recall increase with an increased budget ratio or an increased batch size. For instance, when the batch size is 15, precision of around 0.72 and recall of around 0.64 is achieved with a budget ratio of $\beta = 7$. Increasing the budget ratio means that more answers are obtained, while an increased batch size, again, means that more relations between workers, paragraphs and answers are captured and the collective assessment improves the result quality.

Effects of posting strategy. We further evaluated the presented task posting strategy that involves reposting decisions (*dynamic*) with the traditional crowdsourcing setting (*static*). The latter is the most common task posting strategy in which each paragraph in a batch receives the same, fixed number of answers. Again, we used the referential answer set and executed the argument discovery process. To achieve a fair comparison, we ensure that both approaches are evaluated under the same cost ratio. That is, for a specific budget q, all paragraphs are first posted with the dynamic strategy. Then, we calculate the cost ratio γ and repeat the experiment with the static strategy until the same (rounded) cost ratio is obtained. We varied the batch size q under a fixed budget ratio and uncertainty threshold, $\beta = 10$ and $\tau = 0.4$, and compared the results in terms of precision.

The results in Figure 15 highlight that the dynamic task posting strategy achieves consistently higher precision when the same amount of the effort budget is invested. The reason is that the static method posts more redundant questions, which incur costs, but do not improve the quality of results. In contrast, our dynamic strategy avoids seeking additional input for questions, for which there is a clear trend in the answers.

7.7 End-to-End Process

For a comparative evaluation of the end-to-end process, we assess our technique in the light of the state-of-the-art in automatic argument discovery as well as several baselines. As a state-of-the-art technique, we implemented the automatic argument discovery (*auto*) proposed in [49], which (1) detects argumentative sentences using a maximum entropy classifier, (2) classifies these sentences using support vector machines, and (3) detects the argument structure by context-free grammar parsing.

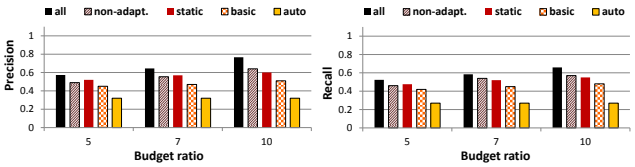

Figure 14: Comparative evaluation of the end-to-end process

We also compare against baselines that also involve user input for the batch size $q = 10$. Those are derived from our technique (*all*) by simplifying paragraph selection and task posting:

- *Basic*: the simplest approach uses a static scoring model (see Section 7.2) and static task posting (see Section 7.6)
- *Static*: extends *Basic* with adaptive learning of the scoring model.
- *Non-adaptive*: extends *Basic* with dynamic task posting.

Using the referential answer set for the experiment, the results in Figure 14 illustrate the limitations of automatic argument discovery (*auto*). Even the most naive baselines using crowdsourcing outperform the automatic approach, while our comprehensive method (*all*) achieves twice as high values for both precision and recall, even for a small budget ratio of $\beta = 5$. While the difference may be partly due to the fact that the approach of [49] has been tailored to legal documents, the benefits of integrating user input are striking.

The results obtained for the baselines reveal that both adaptive learning of the scoring model and dynamic task posting improve the overall performance. The generally higher precision and recall achieved by the approaches *static* and *non-adaptive* compared to *basic* and the fact that highest precision and recall is observed for the comprehensive technique *all* further illustrate that both aspects indeed have a positive impact. For $\beta = 10$, for instance, adaptive learning and dynamic task posting increase the precision from 0.51 (*basic*) to 0.77 (*all*) and the recall from 0.47 to 0.68.

7.8 Real-world Deployment Costs

Finally, we investigated the deployment costs induced by our approach in terms of the amount of effort budget that needs to be invested for real-world document collections. We prepared a budget of 2500 HITs (Human Intelligence Tasks) for each document collection and used AMT with the financial incentive of 0.05$/HIT. The argument extraction process is performed with batch size $q = 10$ and an uncertainty threshold of $\tau = 0.6$.

The constructed argumentation bases are publicly available [1] and summarised in Table 4. The cost ratios – denoting how many questions are asked per paragraph on average – vary slightly for the datasets. In particular, argument discovery appears to be more challenging for the documents in the topics *penalty* and *global*. Yet, even in these cases, the average cost per paragraph is less than 0.40$, which enables argument discovery for large document collections with arguably reasonable investments.

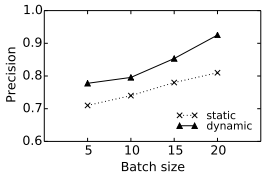

Figure 15: Posting strategy

Dataset	#Procd Paras	#Args	Cost Ratio
vacc	467	184	5.35
food	423	312	5.91
gmo	395	221	6.33
penalty	326	193	7.67
global	338	174	7.39

Table 4: Resulting argumentation bases and induced cost ratios

8. RELATED WORK

We elaborated on approaches for automatic argument discovery in Section 2. Below we thus focus on further related research areas.

Crowdsourcing. Crowdsourcing enables human computation for micro tasks [3], which are broadly classified by the function applied

by crowd workers to some objects: In *discrete* tasks, objects are assigned a label [31]; in *continuous* tasks objects are assigned real values [59]; *partial-function* tasks define objects as rules [4], which is the setting of our method for argument discovery. A major obstacle in crowdsourcing is quality control, i.e., how to ensure answer correctness. To this end, techniques for the assessment of worker quality [32, 37] are typically combined with crowdsourcing answer aggregation. The latter aims at finding the hidden ground truth from an answer set. Non-iterative aggregation [37] computes a single value for each object separately. Iterative techniques, see [31], perform a series of convergent aggregations that consider the answer set as a whole, which is the approach followed in this work.

Despite the above efforts, results of automatic quality control are inherently uncertain [30]. To address this dilemma, we leverage a factor graph model to capture the complex relations between workers, arguments, and answers. Factor graph models have been used in a crowdsourcing setting in [14, 61]. Closest to our work is the work by Demartini et al. [14] that targets the entity linking problem. Yet, since the addressed problems are different, the constructed factor graphs differ as well. In addition to this fundamental difference, our work is geared towards large-scale computation by relying on sampling for the probability estimation.

Truth finding. Aggregation of crowd answers is related to the field of truth finding, both having the goal to improve the quality of aggregated results by considering the reliability of data sources (workers). Given a set of data items claimed by multiple sources, the truth finding (a.k.a. data fusion) problem is to determine the true values of each claimed item, with various applications in information corroboration [17, 25]. Existing work on truth finding models the mutual reinforcing relations between sources and data items, e.g., by maximum likelihood estimation [58], and latent credibility analysis [50]. Closest to our work is the approach described in [51, 69], which is able to handle multiple correct values (in our setting, a paragraph may have many correct atomic arguments). However, these techniques are still not applicable in our setting, since they assume the values to be independent. In contrast, in our work, correctness of one atomic argument can affect the correctness of another one. Also, data generation in truth finding is passive [26], so that a newly available crowd answer cannot be incorporated directly. The use of the factor graph in our work, enables us to include new answers directly, by adding new factors and variables. Further details on the difference between truth finding and crowd answer aggregation can be found in [26].

Natural language processing (NLP). Argument discovery benefits from various techniques developed in NLP, in particular discourse parsing [22] and textual entailment [9]. Discourse parsing identifies discourse elements in a text and their relations, such as contrast, causality or explanation. Prominent theories in discourse parsing are the Penn Discourse Treebank (PDTB) [52] and the Rhetorical Structure Theory (RST) [22]. Both theories differ in how they model relations between elements, either with a shallow structure (PDTB) or with a tree (RST). The latter captures a hierarchy of discourse elements, which is why it is used in our approach to identify claims and evidence. Textual entailment, in turn, refers to the problem of discovering whether a textual fragment supports or contradicts an expression. It can be used to model the relations between arguments [9], but is not applicable if arguments are not known beforehand as it is the case in argumentation mining.

Argumentation systems. As discussed in Section 2, the detection of argumentative structures using argument discovery is the foundation for the evaluation of their convincing power. There are many aspects related to the convincing power of an argument, e.g., *stance classification* of arguments. Most of the works in stance classification leverage machine learning techniques, such as Bayes classifier [45] and feature-based decision trees [5]. Another field related to stance classification is *sentiment analysis*, which aims at identifying the polarity (positive, negative or neutral) of the provided text. Sentiment analysis can be used to identify the sentiment of the arguments. Approaches in sentiment analysis can be divided into machine learning approach, using support vector machines [27], naive Bayes classification [40], or mathematical modeling, such as latent semantic analysis [19]. Another approach to evaluate the convincing power of an argument is to match it with various *argumentation schemes*. Argumentation schemes are stereotypical patterns of human reasoning which are used in everyday conversation or legal, scientific argumentation [8]. These schemes are associated with a set of evaluation questions [57]. By identifying the argumentation scheme of an argument and answering corresponding schemes, one can assess the convincing power of the argument [57].

Text mining. Extracting important information from textual documents is addressed by methods for text summarization that summarize a long document by extracting the most important segments [42, 46]. These techniques are classified as extraction or abstraction. The former creates the summary by extracting important words, phrases and sentences from the original text. The latter leverages the semantics of the text to construct a summary. Our technique is similar to extraction-based text summarization since it also extracts segments of a document. Yet, the extraction goal is different as we want to classify segments as claims or evidence.

Knowledge base construction. Construction of a knowledge base includes extraction of instances, concepts and relations from text, and may be *domain-specific* or *global* [16]. Global knowledge bases such as Freebase [23], YAGO [64] and DBPedia [13] rely on Wikipedia to extract entities and their relations. Although these bases are broadly applicable, there is also a need to build domain-specific knowledge bases [16], such as DBLife [15] and DeepDive [67]. Extracting domain-specific arguments, our work provides a semi-automated approach that falls into the latter category.

9. CONCLUSION AND FUTURE WORK

This paper proposed an end-to-end process to build and maintain a corpus of arguments. It addresses the trade-off of automatic text-mining and manual processing with an iterative learning process for argument discovery that exploits crowdsourcing. In particular, this process involves ranking of argumentative texts, elicitation of user input to extract arguments from these texts, and aggregation of crowd answers using a probabilistic model.

Our experiments showed that our method is effective and efficient, discovering virtually all arguments after processing only 25% of the text with more than 80% precision. Also, it halves the budget spent compared to a baseline algorithm. As a result of our evaluation, we also contribute argumentation bases that have been constructed from large, real-world document collections [1]. These bases comprise between 174 and 312 arguments.

In future work, we strive for realizing argument discovery in a *pay-as-you-go* manner, bootstrapping a corpus with instantiated arguments and refining it when more data becomes available. Also, argument types beyond text (e.g., tables) can be considered to improve the usefulness of argumentation bases. Finally, argumentation bases can fuel a wide range of innovative applications, for instance, in decision making, similar to IBM Watson's Debater [2].

References

[1] URL: http://argdiscovery.github.io/.

[2] E. Aharoni, A Polnarov, T. Lavee, and D. Hershcovich. "A benchmark dataset for automatic detection of claims and evidence in the context of controversial topics". In: *ACL 2014* (2014).

[3] L. von Ahn. "Human computation". In: *DAC*. 2009.

[4] Y. Amsterdamer, Y. Grossman, T. Milo, and P. Senellart. "Crowd Mining". In: *SIGMOD*. 2013.

[5] P. Anand et al. "Cats rule and dogs drool!: Classifying stance in online debate". In: *Proceedings of the 2nd workshop on computational approaches to subjectivity and sentiment analysis*. Association for Computational Linguistics. 2011.

[6] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. "Open information extraction for the web". In: *IJCAI*. 2007.

[7] P. Besnard and A. Hunter. *Elements of argumentation*. MIT press, 2008.

[8] F. Bex, H. Prakken, C. Reed, and D. Walton. "Towards a formal account of reasoning about evidence: argumentation schemes and generalisations". In: *Artificial Intelligence and Law* (2003).

[9] E. Cabrio and S. Villata. "Natural Language Arguments: A Combined Approach." In: *ECAI*. 2012.

[10] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. "Online passive-aggressive algorithms". In: *JMLR* (2006).

[11] I. Dagan, B. Dolan, B. Magnini, and D. Roth. "Recognizing textual entailment: Rational, evaluation and approaches". In: *JNLE* (2009).

[12] A. P. Dawid and A. M. Skene. "Maximum likelihood estimation of observer error-rates using EM". In: *J. R. Stat. Soc.* (1979).

[13] *DBPedia*. `http://www.dbpedia.org`.

[14] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux. "ZenCrowd: Leveraging Probabilistic Reasoning and Crowdsourcing Techniques for Large-scale Entity Linking". In: *WWW*. 2012.

[15] P. DeRose, W. Shen, F. Chen, A. Doan, and R. Ramakrishnan. "Building structured web community portals: A top-down, compositional, and incremental approach". In: *VLDB*. 2007.

[16] O. Deshpande et al. "Building, maintaining, and using knowledge bases: A report from the trenches". In: *SIGMOD*. 2013.

[17] X. L. Dong and F. Naumann. "Data fusion: resolving data conflicts for integration". In: *VLDB*. 2009.

[18] M. Drosou and E. Pitoura. "Disc diversity: result diversification based on dissimilarity and coverage". In: *VLDB*. 2012.

[19] S. T. Dumais. "Latent semantic analysis". In: *Annual review of information science and technology* (2004).

[20] M. T. Dzindolet, S. A. Peterson, R. A. Pomranky, L. G. Pierce, and H. P. Beck. "The role of trust in automation reliance". In: *Int. J. Human-Computer Studies* (2003).

[21] C. Eickhoff and A. de Vries. "How crowdsourcable is your task". In: *CSDM*. 2011.

[22] V. W. Feng and G. Hirst. "A Linear-Time Bottom-Up Discourse Parser with Constraints and Post-Editing". In: *ACL*. 2014.

[23] *Freebase*. `http://www.freebase.com`.

[24] A. J. Freeley and D. L. Steinberg. *Argumentation and Debate*. Springer, 2013.

[25] A. Galland, S. Abiteboul, A. Marian, and P. Senellart. "Corroborating information from disagreeing views". In: *WSDM*. 2010.

[26] J. Gao, Q. Li, B. Zhao, W. Fan, and J. Han. "Truth discovery and crowdsourcing aggregation: a unified perspective". In: *Proceedings of the VLDB Endowment* (2015).

[27] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, and B. Scholkopf. "Support vector machines". In: *Intelligent Systems and their Applications, IEEE* (1998).

[28] J. L. Herlocker, J. A. Konstan, and J. Riedl. "Explaining collaborative filtering recommendations". In: *CSCW*. 2000.

[29] J. J. Horton and L. B. Chilton. "The labor economics of paid crowdsourcing". In: *EC*. 2010.

[30] N. Q. V. Hung, N. T. Tam, L. N. Tran, and K. Aberer. "An Evaluation of Aggregation Techniques in Crowdsourcing". In: *WISE*. 2013.

[31] P. G. Ipeirotis, F. Provost, and J. Wang. "Quality management on Amazon Mechanical Turk". In: HCOMP. 2010.

[32] G. Kazai, J. Kamps, and N. Milic-Frayling. "Worker types and personality traits in crowdsourcing relevance labels". In: *CIKM*. 2011.

[33] A. Kittur, E. H. Chi, and B. Suh. "Crowdsourcing user studies with Mechanical Turk". In: *CHI*. 2008.

[34] K. Krippendorff. *Content analysis: An introduction to its methodology*. Sage, 2012.

[35] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. "Factor graphs and the sum-product algorithm". In: *IEEE Trans. Inf. Theory* (2001).

[36] A. Kulkarni, M. Can, and B. Hartmann. "Collaboratively crowdsourcing workflows with turkomatic". In: *CSCW*. 2012.

[37] K. Lee, J. Caverlee, and S. Webb. "The social honeypot project: protecting online communities from spammers". In: *WWW*. 2010.

[38] X. Li, X. L. Dong, K. Lyons, W. Meng, and D. Srivastava. "Truth finding on the deep web: is the problem solved?" In: *VLDB*. 2013.

[39] A. McAfee and E. Brynjolfsson. "Big data: the management revolution." In: *Harvard business review* (2012).

[40] A. McCallum, K. Nigam, et al. "A comparison of event models for naive bayes text classification". In: *AAAI-98 workshop on learning for text categorization*. Citeseer. 1998.

[41] A. Mikheev. "Tagging sentence boundaries". In: *NAACL*. 2000.

[42] M. Mitray, A. Singhalz, and C. Buckleyyy. "Automatic text summarization by paragraph extraction". In: *Compare* (1997).

[43] R. Mochales and A. Ieven. "Creating an argumentation corpus: do theories apply to real arguments?: a case study on the legal argumentation of the ECHR". In: *ICAIL*. ACM. 2009.

[44] M.-F. Moens, E. Boiy, R. M. Palau, and C. Reed. "Automatic detection of arguments in legal texts". In: *ICAIL*. 2007.

[45] T. Mullen and R. Malouf. "A Preliminary Investigation into Sentiment Analysis of Informal Political Discourse." In: *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*. 2006.

[46] A. Nenkova and K. McKeown. "A survey of text summarization techniques". In: *Mining Text Data*. Springer, 2012.

[47] H. V. Nguyen and D. J. Litman. "Extracting argument and domain words for identifying argument components in texts". In: *ArgMining*. 2015.

[48] S. Oyama, Y. Baba, Y. Sakurai, and H. Kashima. "Accurate integration of crowdsourced labels using workers' confidence scores". In: *IJCAI*. 2013.

[49] R. Palau and M. Moens. "Argumentation mining: the detection, classification and structure of arguments in text". In: *ICAIL*. 2009.

[50] J. Pasternack and D. Roth. "Latent credibility analysis". In: *WWW*. 2013.

[51] R. Pochampally, A. Das Sarma, X. L. Dong, A. Meliou, and D. Srivastava. "Fusing data with correlations". In: *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM. 2014.

[52] R. Prasad et al. "The Penn Discourse TreeBank 2.0." In: *LREC*. 2008.

[53] B. Settles. "Active learning literature survey". In: *Computer Sciences Technical Report*. UW-Madison, 2010.

[54] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. "Cheap and fast: Evaluating non-expert annotations for natural language tasks". In: *EMNLP*. 2008.

[55] K. Sycara. "Persuasive argumentation in negotiation". In: *Theory and Decision* (1990).

[56] P.-N. Tan, M. Steinbach, V. Kumar, et al. *Introduction to data mining*. 2006.

[57] D. Walton and C. Reed. "Argumentation schemes and defeasible inferences". In: *Workshop on computational models of natural argument, 15th european conference on artificial intelligence*. 2002.

[58] D. Wang, L. Kaplan, H. Le, and T. Abdelzaher. "On truth discovery in social sensing: A maximum likelihood estimation approach". In: *IPSN*. 2012.

[59] P. Welinder and P. Perona. "Online crowdsourcing: rating annotators and obtaining cost-effective labels". In: *CVPRW*. 2010.

[60] A. Werder. "Argumentation Rationality of Management Decisions". In: *Organization Science* (1999).

[61] M. Wick, A. McCallum, and G. Miklau. "Scalable probabilistic databases with factor graphs and MCMC". In: *VLDB*. 2010.

[62] F. Wolf and E. Gibson. *Coherence in Natural Language: Data Structures and Applications*. Cambridge, MA, USA: MIT Press, 2006.

[63] L. Xu, P. Yan, and T. Chang. "Best first strategy for feature selection". In: *ICPR*. 1988.

[64] *YAGO*. `http://www.mpi-inf.mpg.de/yago`.

[65] Z. Yang, Y. Li, J. Cai, and E. Nyberg. "QUADS: Question Answering for Decision Support". In: *SIGIR*. 2014.

[66] T. Yuan, D. Moore, and A. Grierson. "A human-computer debating system and its dialogue strategies". In: *Int. J. Intell. Syst.* (2007).

[67] C. Zhang and C. RÃŕ. "Towards High-Throughput Gibbs Sampling: A Study across Storage Managers". In: *SIGMOD*. 2013.

[68] H. Zhang et al. "Human computation tasks with global constraints". In: *CHI*. 2012.

[69] B. Zhao, B. I. Rubinstein, J. Gemmell, and J. Han. "A bayesian approach to discovering truth from conflicting sources for data integration". In: *VLDB*. 2012.

APPENDIX

A. STUDY ON TEXTUAL FEATURES

Setup. To identify textual features that hint at arguments, we conducted a preliminary study. For this study, we collected 500 documents by querying common search engines such as Bing with 5 keywords from the domains of health, society and economics. The retrieved documents have been segmented into around 4000 paragraphs. Then, five experts assessed for each paragraph whether it contains an argument. For the 193 paragraphs that contain arguments, we then calculated the values for a set of features that are commonly used in argumentation mining and NLP [49]. For this dataset, we selected the features that turned out to be good predictors for paragraphs that contain arguments. Following a best-first

selection strategy [63], we ended up with the following features.

Lexical features. Lexical features deal with the words or vocabulary of a language. The following lexical information of a paragraph turned out to be useful to classify its argumentative nature.

Thematic words: the most frequent words (ignoring common stop-words, such as connectives and articles) in a document and thus paragraph are considered to be thematic words. Selecting a small number of thematic words that are particularly relevant to the keyword, this feature is defined as the frequency counts of thematic words. The intuition of this feature is that a paragraph containing the most important keywords should express the argumentative point of view of the writer.

EXAMPLE 8. *Consider the following article regarding vaccine* [1]*, there are various words that appear in high frequency in the documents such as vaccine, shots, disease, autism, health. This means these words discuss the theme of the documents. Among the paragraphs in this document, the following paragraph contains many thematic words, which shows that it may contains an argument:*

(S1) Yes. (S2) Vaccines are safe. (S3) In fact, experts including American Academy of Pediatrics, the Institute of Medicine, and the World Health Organization agree that vaccines are even safer than vitamins. (S4) Millions of children and adults are vaccinated every year safely. (S5) Thousands of people take part in clinical trials to test a vaccine before it is licensed by the Food and Drug Administration (FDA). (S6) After it's licensed, the Vaccine Adverse Events Reporting System (VAERS) helps track any health effect that happens hours, days, weeks, or even months later. (S7) Anyone can report a possible side-effect so that it can be studied. (S8) This monitoring helps ensure vaccines are safe. (S9) To learn more about vaccine safety from the Centers for Disease Control and Prevention, visit the CDC vaccine safety page.

Segment S_2 is likely to be a claim as the segment is the second sentence in the paragraph and the keyword vaccine is the subject. Segment S_3 may be an evidence as it contains evidence-related word (fact) and names (American Academy of Pediatrics, Institute of Medicine, World Health Organization). Segment S_4 and S_5 can also be evidence as they contains numbers (millions, thousands).

Evidence-related words: A paragraph containing many evidence-related words, such as numbers, citations, and cue phrases (e.g., 'because of') is likely to contain arguments. This feature, thus, is defined as the occurrence count of such evidence indicators.

EXAMPLE 9. *(S1) Processed foods destroy your mind. (S2) If you suffer from chronic bouts of brain "fog," or have difficulty concentrating and thinking normally, chances are your diet has something to do with it. (S3) And a recent study out of Oxford University lends credence to this possibility, having found that junk food consumption can cause people to become angry and irritable.(S4) Nutrient-dense whole foods, on the other hand, can help level out your mood, sustain your energy levels, and leave you feeling calmer and more collected.*

This paragraph may contain an argument as an evidence-related keyword (study) is available in segment S_3 of the paragraph. In addition, the keyword processed foods appears as a subject of the first sentence, which shows that this sentence is likely to be a claim.

Prototypical words: Prototypical words are lexical expressions used to formulate arguments, for instance, 'argue' or 'believe'. As part of our preliminary study, we learnt a list of 97 prototypical words. Each prototypical word is associated with a weight, indicating the likelihood of a paragraph containing the given word to include an argument. This weight is determined based on the relative frequency of the word in the training data and in the overall set of documents. Again, the respective feature is the occurrence count of the prototypical words, normalized by their respective weight.

EXAMPLE 10. *(S1) Believe it or not, almost all the food that you eat, even the foods 'made from scratch,' have actually been processed. (S2) According to an article published in the journal, Advances in Nutrition, any food that has been subject to washing, cleaning, milling, cutting, chopping, heating, pasteurizing, blanching, cooking, canning, freezing, mixing, and packaging that alter the food from its natural state is considered a 'processed food.'*

The above paragraph contains an argument as segment S_1 which contains a prototypical word (believe). This prototypical word signifies that it may be a claim. In addition, segment S_2 provides information coming from a reliable source which is the Advances in Nutrition *journal*

Syntactical features. Syntactical features refer to the text structure and capture local relations between words within a sentence. Our preliminary study identified the following syntactical features:

- *Part-of-speech*: Words in a sentence may be classified into different parts-of-speech, such as nouns, verbs and adjectives. Intuitively, this feature exploits the fact that, if the keyword appears to be a subject or object of a sentence, it is likely that the sentence is a relevant claim or evidence. Technically, this feature is defined as the occurrence count of the keyword in a paragraph either as a subject or as an object.
- *Parse structure*: A sentence can be parsed into a tree-like structure that captures syntactical relations between the phrases within that sentence. In the parse structure, the relation between the phrases is determined by the *head word*, which is an indicator of the mentioned topic. We use the appearance of the keyword as the head word of phrases in a paragraph as a classification feature indicating whether the paragraph contains arguments.

For illustration, we consider the following paragraph that is relevant for the topic of 'processed food' discussed above.

EXAMPLE 11. *Furthermore, the engineering behind processed food makes it virtually addictive. A 2009 study by the Scripps Research Institute indicates that overconsumption of processed food triggers addiction-like neuroaddictive responses in the brain, making it harder to trigger the release of dopamine. In other words the more processed food we eat, the more we need to give us pleasure; thus the report suggests that the same mechanisms underlie drug addiction and obesity.*

This paragraph illustrates various of the features that hint at arguments, see also Table 5. Examples include evidence-related words, such as 'study' and prototypical words, such as 'indicate'. In addition, thematic words such as 'addictive' render the paragraph important for argumentation mining. Further, the keyword 'processed foods' appears several times, e.g., in the first sentence, part-of-speech tagging identifies the keyword as the subject of a sentence.

Table 5: Features and example values

Feature types	Example values for keyword 'processed foods'
Thematic words	additive, addiction
Evidence-related words	because of, by the fact that
Prototypical words	argue, claim, believe
Keyword as subject or object	the engineering behind processed food makes [...]
Keyword as head word	overconsumption of processed foods

[1] http://www.whyichoose.org/vaccinesafety.html