BAM: Benchmarking Argument Mining on Scientific Documents

Florian Ruosch[1], Cristina Sarasua[1] and Abraham Bernstein[1]

[1]*University of Zurich, Department of Informatics, Binzmühlestrasse 14, 8050 Zürich, Switzerland*

Abstract

In this paper, we present BAM, a unified **B**enchmark for **A**rgument **M**ining (AM). We propose a method to homogenize both the evaluation process and the data to provide a common view in order to ultimately produce comparable results. Built as a four stage and end-to-end pipeline, the benchmark allows for the direct inclusion of additional argument miners to be evaluated. First, our system pre-processes a ground truth set used both for training and testing. Then, the benchmark calculates a total of four measures to assess different aspects of the mining process. To showcase an initial implementation of our approach, we apply our procedure and evaluate a set of systems on a corpus of scientific publications. With the obtained comparable results we can homogeneously assess the current state of AM in this domain.

1. Introduction

In the last 200 years, the number of published papers per year has consistently been increasing by around 5% [1]. With this rapidly growing landscape, it becomes harder to manually navigate the seemingly unending flood of new scientific information.

One of the emerging fields addressing the machine-assisted processing of scholarly documents is *Argument Mining* (AM), aimed at identifying and extracting argument components (and possibly relations) from natural language texts [2]. This information is not only useful for summarization but also for detecting connections between different entities such as individual papers or outlets [3]. This kind of network has been described as the Argument Web by Bex et al. [4] — a vision where all argument data is URI-addressable and linked. If we want to work toward the automatic implementation of such a knowledge graph containing arguments from scientific publications, we first need to be able to compare the performance of existing solutions. However, there is currently no widely established, standardized AM benchmarking approach.

Lippi and Torroni [5] point out several problem areas which stand in the way of a homogeneous evaluation: the granularity of the in- and output of AM systems, the variety of genres and domains they focus on, and the representation of arguments in the evaluation data, i.e. the argument model. Additionally, as previously noted by Duthie et al. [6], a wide spectrum of different *measures* are in use, and these are not accurately described or appropriately applied in all cases. To address the issues above, we propose BAM, a unified approach to **B**enchmarking **A**rgument **M**ining.

Following the AM pipeline described by Lippi and Torroni [5], we create a multi-level evaluation framework to enable the benchmarking of every task of their AM process: sentence classification, boundary detection, component classification, and relation prediction. We aim to provide a benchmarking framework that facilitates comparable results both within each stage and throughout the whole pipeline.

In this work, we present a two main contributions. First and foremost, we show the concept of a unified benchmark approach for Argument Mining: BAM. To the best of our knowledge, nothing of the sort exists yet. Furthermore, we showcase a preliminary implementation of BAM by applying our benchmark to a pre-existing argument annotated corpus of scientific publications [7]. This allows for not only showing the feasibility of our approach but also to present an initial comparison of the performance results of a range of AM systems in the domain of scholarly papers.

The remainder of this paper is structured as follows: Section 2 presents related works and Section 3 introduces our newly proposed methodology. In the ensuing Section 4, we showcase our benchmark and describe the results, before we draw conclusions in Section 5.

2. Related Work

We first explore the definition of AM and then point to an overview of efforts in the domain of scientific publications. In the second part, we address existing measures used to evaluate the performance of AM. Finally, we describe available benchmarks.

ruosch@ifi.uzh.ch (F. Ruosch); sarasua@ifi.uzh.ch (C. Sarasua); bernstein@ifi.uzh.ch (A. Bernstein)
0000-0002-0257-3318 (F. Ruosch); 0000-0002-2076-9584 (C. Sarasua); 0000-0002-0128-4602 (A. Bernstein)

2.1. Argument Mining

Despite the different interpretations of what AM entails [8], there is the well-established information extraction approach, as popularized by several experts in the field [9, 2, 5]. Stab et al. [10] explain AM as a multistage pipeline that extracts the arguments from text, usually by first separating non-argumentative from argumentative units, then classifying the argument components and, finally, identifying their structure with relations. We adopt this definition because it fits best with our ultimate goal of creating the Argument Web of Science [4] for which we need to extract information about argumentative units and their relations. Other AM papers [11], treat the mining process as a search task to retrieve arguments from a pre-computed set according to their relevance for a query or keyword.

For a detailed overview of literature of the last 20 years in the field of AM for scientific publications, we point the inclined reader to the survey of Al Khatib et al. [12]. They do not only present an overview about the efforts made but also indicate current applications and identified challenges.

2.2. Measures

There is a wide range of Information Retrieval (IR) measures used to evaluate AM systems. Typically, IR evaluations presume the existence of a gold standard or ground truth that a proposed solution is evaluated against [13]. The F1 (also F-score or F-measure) [14] can be used to assess the accuracy of the predictions made by a system by calculating the harmonic mean of the *precision* and the *recall*, both of which have also been applied on their own for performance evaluation. For multi-class tasks (such as AM, where we aim to identify various components or relations) different versions of F1 exist, based on how the score is averaged for the classes. The *macro-F1* variant weights all classes equally for the combination into a single F1, while *micro-F1* considers the number of occurrences for each label. Not only are we unable to directly compare results reported for different variants of the F-score, some literature also chooses not to include the specifics of which weighting method was employed.

Duthie et al. [6] raise the issue that traditional measures from the field of IR may over penalize when simply applying them for each of the pipeline stages successively. For example, wrongly or not at all identified components directly influence and reduce the calculated performance of the relation prediction task. To address this and other shortcomings, they introduce the Combined Argument Similarity Score (CASS) [6]. It splits the evaluation of AM into three individual scores which are then aggregated into a single number. First, the segmentation step evaluates the similarity of different partitionings of the same text, i.e. the boundaries of the identified components. For the relation scores, these segments are aligned between annotations. Considering the Levenshtein distance [15] and also the location in the text, the components are mapped. Then, the number of correctly predicted connections (also with respect to their types) is calculated for propositional (attack, support) and dialogical (considering the speaker's intent) relations.

Even though CASS is very flexible (i.e. scheme agnostic), it still has some drawbacks. Firstly, it assumes the existence of dialogical annotations, which is not common in current automated AM approaches. Also, there is no public implementation such that it could be put into practice. Finally, it wholly omits the component classification part of the AM pipeline by only focusing on the segmentation (i.e. boundaries of the components) of the text. By introducing our own evaluation method, we aim to remedy the points mentioned above.

2.3. Benchmarks

We found two previous works that designate themselves specifically as benchmarks.

NoDE [16] consists of a total of three data sets covering different domains: online discussions, a stage play, and the revision history of Wikipedia articles. The source for the first part were different online debate platforms which allow members to discuss controversial topics such as violent video games or abortion. Secondly, arguments were extracted from the play "Twelve Angry Men", where a jury discusses the culpability of a young man in a murder case. The third data resulted from comparing two different Wikipedia dumps based on the edits of the five most revised pages. All three sets were annotated by a team of two ($\kappa = 0.70 - 0.74$) and, in total, they contain 792 pairs, each connecting two arguments with information about entailment. Partly, they are also annotated with support- or attack-relationships. It is of note that it is not possible to use this benchmark to evaluate the whole AM pipeline since it does not contain any information about the boundaries of arguments in continuous text.

Aharoni et al. [17] present a data set based on Wikipedia pages for a range of controversial topics. In the labeling process, they first extracted claims from the articles, followed by supporting evidences. Given that each claim is identified context-dependently, they are inherently assigned to a topic. Every evidence is then connected to a claim and given a type (study, expert, or anecdotal). The labeling was conducted by 20 inhouse annotators with a Cohen's κ of 0.39 for the claims and 0.40 for the evidences. The corpus covers a total of 33 topics with 1392 claims and 1291 evidences. Notably, all evidences are supporting and no attack relation is annotated. It also does not contain explicit information about the location of the components in the text (and thus the boundaries).

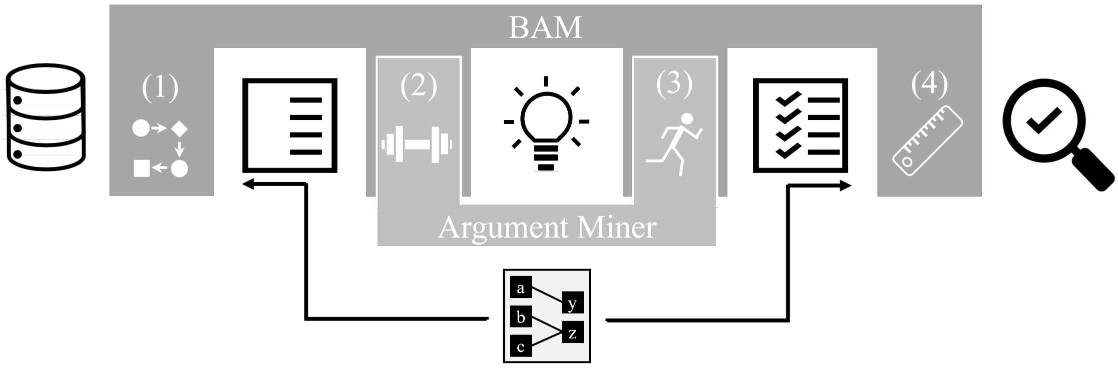

Figure 1: System vision of the different parts of the benchmark framework and their interactions.

These two works share one major drawback: Instead of providing a framework including one or more evaluation measures and a state-of-the-art benchmarking methodology, they solely present a new data set, that can be used as a ground truth. Thus, no uniform method to assess the performance of AM system is established since the choice of the measure has not been fixed. We address this issue in our work.

3. BAM: A Unified Approach to Benchmarking Argument Mining

We first describe the architecture of the end-to-end benchmarking pipeline. Then, we specify the measures employed to assess performance for the different stages. Finally, we describe our argument representation unification effort.

3.1. Overview

We designed BAM, the benchmark for Argument Mining, with the goal of not only providing an easy to access system but also considering all aspects of AM and to obtain performance results in a unified and homogeneous way. Figure 1 outlines the end-to-end pipeline and illustrates how BAM is built on four pillars, from left to right: (1) pre-processing, (2) training, (3) execution, and (4) evaluation. The implementation was done in Python and is available publicly.[1] We provide several examples of how to integrate AM systems via the implemented Python stubs.

With pre- and post-processing being taken care of by our framework, the system can then address the training, where applicable, and execution step, which are both integrated into the end-to-end pipeline. We explain each of these functionalities separately below.

(1) Pre-processing This step creates a data set suitable to be processed by a given system from a common ground truth corpus, according to specified configurations and the alignment of argument representations. It is tailored to the requirements of the system to be benchmarked such that it can be used as input at any stage, be it for training or evaluation. This ensures that every system tested in the benchmark will use the same data as basis, thus allowing for comparable results. The split of the data into development, training, and test set is specified not per system but rather per corpus ensuring comparability between systems.

(2) Training Given the prevalence of neural network approaches for AM, we included an optional training step. Here, the training API of the system to be integrated can be invoked using the specifically created data set.

(3) Execution The resulting trained model is then employed to annotate the test data set using the system's execution API. We enabled the functionality to either reuse the intermediate results as input for the subsequent steps or to test aspects independently and inject ground truth annotations into the pipeline (e.g., relation prediction with the components as annotated in the ground truth).

(4) Evaluation This stage aligns the computed results and the ground truth annotations to ensure the data conforms to the requirements set by the evaluation functions, which expect sequences of labeled tokens. This is achieved by applying NLTK's [18] tokenizer, where

[1] https://gitlab.ifi.uzh.ch/DDIS-Public/bam

necessary. Since the system's output may already be tokenized using an unknown technique, we have to expect differences in the labeled tokens. To address them, we match the two sequences with spaCy's [19] implementation of the token aligner[2] and, thus, all of the evaluation happens uniformly on token-level. Subsequently, several aspects are evaluated. Based on the AM pipeline described by Lippi and Torroni [5], our benchmarking framework assesses performance for four different tasks: argumentative sentence classification (**S**), boundary identification (**B**), argumentative component detection (**C**), as well as argumentative relation prediction (**R**). A sentence is classified as argumentative, if it contains any argument component [5]. Next, the similarity of the boundaries for the (non)argumentative segments is compared. Before the final stage, the detection and classification of the components themselves is assessed. Lastly, the predicted relations are compared to the ones annotated in the ground truth, i.e. which components are connected and how. It is important to note that we do not require every system to perform all the tasks, but rather the implementation specifies which are covered in the configuration and which are not. The details for each evaluated aspect are presented below.

By relying on a modular structure, we give enough room for customizations to account for any peculiarities that systems might exhibit.

Furthermore, each system needs to specify a mapping (represented by the graph icon on the bottom of Figure 1) to create a uniform view of the argument representation and to make the results comparable. It is employed for pre-processing, to create a specific data set, and for the evaluation, to map all systems to the same argumentation scheme. By specifying the mapping with Semantic Web technologies (OWL[3]), we not only ensure that it is machine-readable and interoperable, but we also facilitate its extension and reuse.

3.2. Evaluation Measures

Every task is treated as a (multinomial) classification. However, we use several evaluation methods because they differ slightly in the granularity and format of the data as well as their goal. We explain the measures and their reasoning for every step of the pipeline.

In the first task, the aim is to classify sentences as (non-)argumentative. If a sentence contains at least one argument component, it is defined as argumentative [5]. After extracting these annotations from the mined results as well as from the ground truth, we compare two lists of the same length with binary values using micro-F1, to ensure that a possible label imbalance does not affect the result and weigh both classes equally. In our benchmarking framework, we apply sklearn's implementation of the micro-F1 measure[4] to obtain a score between 0 and 1, where bigger signifies better.

For the comparison of the component boundaries, we follow the proposition of Duthie et al. [6] and use the implementation[5] of the segmentation evaluation [20]. The edit distance-based *boundary similarity* function assesses how well the results of segmentation tasks agree on a scale from 0 to 1. It compares pairs of boundaries, calculates the edit-distance, and normalizes based on the segmentation length. As input, we can simply pass two sequences of (multiclass) labels assigned to the tokens and the library will identify the boundaries automatically.

Given that the previous measure does not take the categories of the segments into account (i.e., the component types), we have to address the classification in a separate step. Based on the similarity of this task to Named Entity Recognition (NER) [12], we can employ the nervaluate-package[6] originally designed for the evaluation of NER. By treating the argumentative components as named entities, we apply the same functions and obtain the F1 through this well-established library.

The final evaluation step assesses the correctness of the predicted relations between the identified components. As pointed out by Duthie et al. [6], it is important to consider the possible double penalization since the previously detected argumentative units play a critical role. Not having identified certain components also takes away the opportunity to relate them and, thus, is not only penalized in the previous step but also has an impact on the relation prediction score. Consequently, we give the possibility to either use the argumentative units as identified by the system (i.e., the intermediate results) or to recourse to the ground truth as the input for this step. When using the computed intermediate results, we match the components to the ground truth to ensure fairness so that the boundaries do not need to coincide exactly. Instead, we assign each identified unit to one in the ground truth, if they overlap in at least one token. For components covering multiple ones in the ground truth, we select the one with the largest intersection. This does not only allow for differing boundaries, it also ensures that localization information of the units is factored in. By constructing triples out of the two components and the relation (subject, predicate, object), we obtain lists of predicted and gold data. This turns the problem into identifying retrieved/missed, relevant/irrelevant triples. Therefore, we can again employ the F1-score. One caveat is that we also need to consider the symmetric nature of some relation types. By converting the data into triples,

[2] https://github.com/explosion/spacy-alignments
[3] https://www.w3.org/OWL
[4] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score
[5] https://github.com/cfournie/segmentation.evaluation
[6] https://pypi.org/project/nervaluate

we risk not awarding a correct prediction if it is reversed (object and subject transposed) for a symmetric relation. To amend this issue, we always arrange them in such a way that the subject has the smaller identifier number than the object. Since no relation is reflexive, this results in unique triples.

3.3. Aligning Argumentation Schemes

To produce comparable results, a common view of how an argument is represented in data is necessary. This is achieved by aligning different argumentation schemes through mappings. Given the widespread adoption [5] of the *claim/premise* model [21] and its simplicity, we chose it for our benchmark and use the *attacks*- and *supports*-relations to connect components with the import notion that we do not restrict neither range (source) nor the domain (target) for both.

To align representations, we need two types of mappings: one for the components (*claim* and *premise*) and one for the relations (*supports* and *attacks*). There are two different scenarios: either one scheme is more complex than the other (i.e., it has more components and/or relations or has other levels of specificity) or they are the same but use a different naming convention (e.g., synonyms or similar but not identical terms such as *attacks* versus *attack*). There is also the special situation for the components that a model is as simple as to only segmenting text into non- and argumentative parts. In this case, we do not assess the system's ability to classify argumentative components due to the lack of information and, thus, no mapping is necessary.

More complex schemes can be reduced to a simpler model with the concepts of `equivalent`- and/or `subclass-of`-relations. Every component and relation from the original representation is assigned to exactly zero or one corresponding element of the benchmark model, depending on whether their complement exists and according to their definition in the original model descriptions. Elements without a counterpart are mapped to no type since they can not be considered in the evaluation. It is important to note that no annotations are discarded since the ground truth data is recomputed for every run and, if the mapping changes, the alterations are incorporated automatically.

In the case of using different naming, we only need to employ the `equivalent`-relation. The same concept may be called differently but still carry the identical semantics. *Claims* are labeled as *conclusions*, while *premises* have a plethora of names in literature such as *data*, *evidence*, or *reason* [5]. Similarly, the *attacks*-relation is also known as *contradicts*. Based on the definitions, we can create a one-to-one-mapping between model elements and, subsequently, a uniform view of the argument model.

4. Showcasing BAM

To illustrate the feasibility of our benchmarking framework, we showcase it with an example data set and a limited number of systems. This section first introduces the used corpus, before elaborating on the selection of argument miners. Ensuingly, we explain the alignment of the different argumentation schemes and, finally, present a set of initial results.

4.1. Setup

For our showcase, we use the corpus presented by Lauscher et al. [7], currently the only available collection of fully argument annotated scientific papers in English. The authors extend the Dr. Inventor data set [22] by annotating arguments for 40 publications in the field of computer graphics containing 10'780 sentences in total. According to the guidelines [23], several types of components have been annotated: *background claim* (i.e., a claim about someone else's work), *own claim* (i.e., proprietary contribution), and *data* (i.e., the evidence). Furthermore, they identify relations between the argumentative units: *contradicts*, *supports*, *semantically same*, and *part of*. The corpus is publicly available and can be downloaded from the project's homepage.[7]

According to our previously defined requirements, we select an initial list of systems to be included in the showcase. TARGER [24] identifies and tags argument units as claims or premises on token-level from free text input. It implements a BiLSTM-CNN-CRF [25] and uses pre-computed word embeddings, such as GloVe [26]. Mayer et al. [27] present an AM approach for the domain of healthcare employing bi-directional transformers and combining them with neural networks (LSTM, GRU, CRF), which we label as TRABAM (for TRansformer-Based AM) in this paper. Not only do they address the task of identifying argument components (claim, evidence, and major claim) with a sequence tagging solution but they also identify relations between these units phrased as a multichoice problem (attack, support, non). Trautmann et al. [28] also formulate AM as sequence tagging problem and define the task of Argument Unit Recognition and Classification (AURC). They argue for a more fine-grained identification of spans than on sentence-level. At the same time, the authors present a solution using the established sequence labeling model of Reimers et al. [29] which employs BILSTMs in combination with word embeddings.

We include two more systems that, despite being pretrained externally, have received attention in the state of the art due to their respective approaches. However, we do not strictly add them to the benchmarked results in

[7] http://data.dws.informatik.uni-mannheim.de/sci-arg/compiled_corpus.zip

System	Training Time	Run Time
AURC	3d 12h 37m	3h 05m
TARGER	1d 06h 05m	1h 53m
TRABAM	2d 22h 41m	5h 37m
ArguminSci	-	3m
MARGOT	-	37m

Table 1
Overview of the systems included in the showcase along with training and running times.

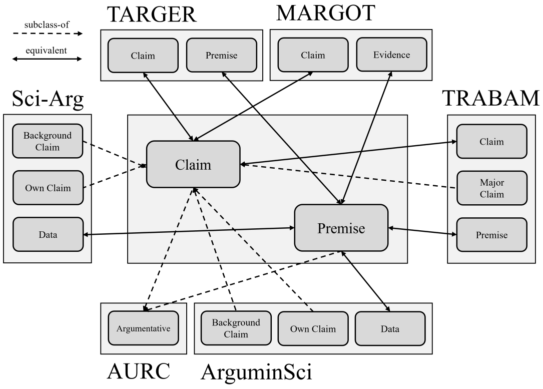

Figure 2: Mapping between different argumentation schemes for the *components*.

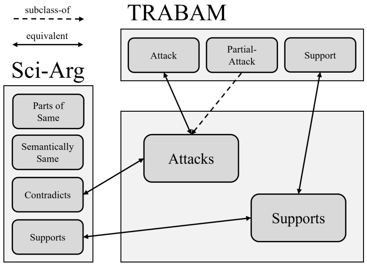

Figure 3: Mapping between different argumentation schemes for the *relations*.

order to ensure fair comparisons (i.e., of systems trained and executed uniformly and homogeneously within the framework). We consider these additions relevant to extend the range of initially available results and to demonstrate the inclusion of systems. While ArguminSci [30] is a suite of tools that enable the analysis of a range of rhetorical aspects, We solely employ the unit for argument component identification. Taking natural language text, it processes the vector representation of sentences with a pre-trained BiLSTM, feeds the results into a single-layer network, and, finally, applies a softmax-classifier to identify and tag tokens as argumentative components. The three labels coincide with the ones used in the ArgSci corpus: *own claim*, *background claim*, and *data*. MARGOT [31] makes use of the information contained in the structure of sentences, identifies *claims* and *evidences*, and detects their boundaries. Employing a subset tree kernel [32], the similarity of constituency parse trees is assessed and sentences classified accordingly as containing part of an argument.

As a baseline, we also evaluate the results of assigning the most frequent labels. Every token is outside of an argumentative component (*O*), and the relations are all non-existent (*noRel*).

As previously pointed out, we adopt the most general and widely-adopted model defining *claim* and *premise* for the conceptual representation of arguments. We connect components using the *attacks* and *supports* relations. The elements of the models (i.e., concepts and relations) of the individual systems are aligned to this unifying model via relations that denote `equivalence` or `subsumption`, implemented using RDF. Figures 2 and 3 visualize the mapping for the schemes of the components and relations, respectively.

Our experiments were executed on a Debian virtual machine with a single CPU with eight cores at 2.2 GHz and 209 GB of RAM.

4.2. Results

All systems required more than 30 hours to train and several hours to execute on the test data (see Table 1 for run times). TARGER takes the least amount of time for both training and execution, and its accuracy is similar to that of the other two systems, save for the classification of the sentences. It scores S = 0.653, which is several percentage points behind both AURC (S = 0.792) and TRABAM (S = 0.832) (see Table 2 for performance indicators). However, TARGER (B = 0.483) manages to beat AURC (B = 0.470) for the boundary identification. TRABAM still outperforms both of them (B = 0.506) in every aspect, while also exhibiting the additional functionality to predict the relations. TARGER (C = 0.656) is almost even with TRABAM (C = 0.662) for the component identification score. Still, TRABAM is the sole system performing relation prediction R = 0.318 and does so to score while relying on the components as annotated in the ground truth.

The two pre-trained systems achieve worse results. This comes partly as a surprise, given that at least ArguminSci was trained on the same data set. It clearly outperforms MARGOT on the sentence classification (S = 0.600 and S = 0.454, respectively), but has a similar score for boundary detection (B = 0.115 and B = 0.097) and is even beat for the component identification (C =

System		S	B	C	R
AURC	[28]	*0.792*	0.470	-	-
TARGER	[24]	0.653	*0.483*	*0.656*	-
TRABAM	[27]	**0.832**	**0.506**	**0.662**	**0.318**
ArguminSci	[30]	0.600	0.115	0.091	-
MARGOT	[31]	0.454	0.097	0.133	-
BASELINE	most frequent labels (*O*, *noRel*)	0.457	0.000	0.000	0.000

Table 2
Results of the benchmark showcase.

0.091 and C = 0.133).

A possible explanation for ArguminSci's poor performance is the fact that it does not always produce well-formed tags for all the chunks. These annotation errors are factored into the calculation of both the B and C score. Naturally, the non-existent training time very much accelerates the whole pipeline and in contrast to the other systems, pre-trained ones can annotate the whole test set in a matter of minutes instead of hours or even days.

When comparing the system results to the baseline, it can be observed that using the most frequent labels is only rewarded for the sentence classification score (S = 0.457), but yields zeroes across the board for the other individual measures. This is intended, since the benchmark is designed to only consider identified actual argumentative content (components or relations), which is useful for building a graph representation of content.

5. Conclusions

In this paper, we have presented BAM, a novel and unified approach to benchmarking Argument Mining. We described its modular architecture, consisting of four pillars (pre-processing, training, execution, and evaluation). To produce a first set of results and illustrate its application, we fully showcased our benchmarking framework which included several state-of-the-art AM systems (TARGER, TRABAM, and AURC) and, partially, (without training) two other systems (MARGOT and ArguminSci).

The main insight is that it was possible to create a unified benchmark to produce comparable results. Different systems could be integrated with some additional code and, subsequently, could execute our pipeline. Our experiments showed that longer execution time does not necessarily imply better performance. Also, more specialized systems do not guarantee higher scores in the tasks they cover compared to other approaches with more capabilities. From our results, we see not only the differences among the AM tools but also between the evaluated aspects with more complex tasks [12] resulting in lower scores. Furthermore, a gap between the best performing system and human annotators is also still evident in the domain of scholarly documents.

The biggest obstacles in both the implementation and the execution of the benchmark were the variety of applied approaches and differences in methodologies. Furthermore, the format of the in- and output varied, which necessitated a lot of custom code for every system. Ultimately, it was possible to develop an end-to-end benchmark for a handful of argument miners, which produces directly comparable results to gauge the state-of-the-art in the field. Although these results are based on the assumption that a ground truth data set labeled with high inter-rater agreement exists ex ante, the curation of annotated data remains a challenge in AM [33]. Here, the advent of deep learning techniques and their demand for data as well as the opportunity to incorporate the crowd in the annotation process [34] should produce relief in the long term.

As future work, we plan to evaluate our proposed approach and to provide a larger list of results obtained by our benchmark to analyse the state of AM in the domain of scholarly documents. We hope our work will serve as a step toward quantifying the quality of the Argument Web [4] of Science that the current state of the art could potentially achieve.

Acknowledgments

This research has been partly funded by the Swiss National Science Foundation (SNSF) under contract number 200020_184994 (CrowdAlytics Research Project). The authors would also like to thank the anonymous reviewers for their constructive feedback.

References

[1] M. Ware, M. Mabe, The stm report: An overview of scientific and scholarly journal publishing (2015).

[2] K. Budzynska, S. Villata, Argument Mining, IEEE Intelligent Informatics Bulletin 17 (2015) 1–6.

[3] N. Green, Argumentation mining in scientific discourse, CEUR Workshop Proceedings 2048 (2017) 7–13.

[4] F. Bex, J. Lawrence, M. Snaith, C. Reed, Implementing the argument web, Communications of the ACM 56 (2013) 66–73. doi:10.1145/2500891.

[5] M. Lippi, P. Torroni, Argumentation mining: State of the art and emerging trends, ACM Transactions on Internet Technology 16 (2016) 1–25. doi:10.1145/2850417.

[6] R. Duthie, J. Lawrence, K. Budzynska, C. Reed, The CASS Technique for Evaluating the Performance of Argument Mining, Proceedings of the Third Workshop on Argument Mining (ArgMining2016) (2016) 40–49. doi:10.18653/v1/w16-2805.

[7] A. Lauscher, G. Glavaš, S. P. Ponzetto, An Argument-Annotated Corpus of Scientific Publications, Proceedings of the 5th Workshop on Argument Mining (2018) 40–46. doi:10.18653/v1/w18-5206.

[8] S. Wells, Argument Mining: Was Ist Das?, Proceedings of the 14th International Workshop on Computational Models of Natural Argument (CMNA14), Krakow, Poland (2014).

[9] P. Saint Dizier, The lexicon of argumentation for argument mining: methodological considerations, Anglophonia. French Journal of English Linguistics (2020).

[10] C. Stab, C. Kirschner, J. Eckle-Kohler, I. Gurevych, Argumentation mining in persuasive essays and scientific articles from the discourse structure perspective, CEUR Workshop Proceedings 1341 (2014).

[11] C. Stab, J. Daxenberger, C. Stahlhut, T. Miller, B. Schiller, C. Tauchmann, S. Eger, I. Gurevych, ArgumenText: Searching for Arguments in Heterogeneous Sources, Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: demonstrations (2018) 21–25. doi:10.18653/v1/n18-5005.

[12] K. Al Khatib, T. Ghosal, Y. Hou, A. de Waard, D. Freitag, Argument Mining for Scholarly Document Processing: Taking Stock and Looking Ahead, in: Proceedings of the Second Workshop on Scholarly Document Processing, Association for Computational Linguistics, 2021, pp. 56–65. URL: https://2021.argmining.org/.

[13] H. Schütze, C. D. Manning, P. Raghavan, Introduction to information retrieval, volume 39, Cambridge University Press Cambridge, 2008.

[14] C. van Rijsbergen, Information retrieval, 2nd edbutterworths, 1979.

[15] V. I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals, in: Soviet Physics-Doklady, volume 10, 1966, pp. 707–710.

[16] E. Cabrio, S. Villata, NoDE: A Benchmark of Natural Language Arguments, Frontiers in Artificial Intelligence and Applications 266 (2014) 449–450. doi:10.3233/978-1-61499-436-7-449.

[17] E. Aharoni, A. Polnarov, T. Lavee, D. Hershcovich, R. Levy, R. Rinott, D. Gutfreund, N. Slonim, A Benchmark Dataset for Automatic Detection of Claims and Evidence in the Context of Controversial Topics, Proceedings of the first workshop on argumentation mining (2014) 64–68. doi:10.3115/v1/w14-2109.

[18] E. Loper, S. Bird, Ntlk: The natural language toolkit, arXiv preprint cs/0205028 (2002).

[19] M. Honnibal, I. Montani, S. Van Landeghem, A. Boyd, spaCy: Industrial-strength Natural Language Processing in Python, 2020. URL: https://doi.org/10.5281/zenodo.1212303. doi:10.5281/zenodo.1212303.

[20] C. Fournier, Evaluating text segmentation using boundary edit distance, in: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2013, pp. 1702–1712.

[21] D. Walton, Argumentation Theory: A Very Short Introduction, Springer US, Boston, MA, 2009, pp. 1–22. URL: https://doi.org/10.1007/978-0-387-98197-0_1. doi:10.1007/978-0-387-98197-0_1.

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

[23] A. Lauscher, G. Glavas, S. P. Ponzetto, K. Eckert, Annotating arguments in scientific publications, 2018.

[24] A. Chernodub, O. Oliynyk, P. Heidenreich, A. Bondarenko, M. Hagen, C. Biemann, A. Panchenko, TARGER: Neural Argument Mining at Your Fingertips, Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (2019) 195–200. doi:10.18653/v1/p19-3031.

[25] X. Ma, E. Hovy, End-to-end sequence labeling via bi-directional lstm-cnns-crf, arXiv preprint arXiv:1603.01354 (2016).

[26] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.

[27] T. Mayer, E. Cabrio, S. Villata, Transformer-based argument mining for healthcare applications, in: Frontiers in Artificial Intelligence and Applications, volume 325, 2020, pp. 2108–2115. doi:10.3233/FAIA200334.

[28] D. Trautmann, J. Daxenberger, C. Stab, H. Schütze, I. Gurevych, Fine-Grained Argument Unit Recognition and Classification, AAAI (2020) 9048–9056. URL: https://doi.org/10.1609/aaai.v34i05.6438.

[29] N. Reimers, B. Schiller, T. Beck, J. Daxenberger, C. Stab, I. Gurevych, Classification and Clustering of Arguments with Contextualized Word Embeddings, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Florence, Italy, 2019, pp. 567–578. URL: https://arxiv.org/abs/1906.09821.

[30] A. Lauscher, G. Glavaš, K. Eckert, ArguminSci: A Tool for Analyzing Argumentation and Rhetorical Aspects in Scientific Writing, Proceedings of the 5th Workshop on Argument Mining (2018) 22–28. doi:10.18653/v1/w18-5203.

[31] M. Lippi, P. Torroni, MARGOT: A web server for argumentation mining, Expert Systems with Applications 65 (2016) 292–303. URL: http://dx.doi.org/10.1016/j.eswa.2016.08.050. doi:10.1016/j.eswa.2016.08.050.

[32] M. Collins, N. Duffy, New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron, in: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, 2002, pp. 263–270.

[33] P. Accuosto, M. Neves, H. Saggion, Argumentation mining in scientific literature: From computational linguistics to biomedicine, CEUR Workshop Proceedings (2021) 20–36.

[34] Q. V. H. Nguyen, C. T. Duong, T. T. Nguyen, M. Weidlich, K. Aberer, H. Yin, X. Zhou, Argument discovery via crowdsourcing, VLDB Journal 26 (2017) 511–535. doi:10.1007/s00778-017-0462-9.