Documentation of BAM: Concept and Implementation

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

[1] lastname@ifi.uzh.ch, Dynamic and Distributed Information Systems, Department of Informatics, University of Zurich
[*]Corresponding author

March 24, 2023

1 Introduction

This document describes the concept and implementation of BAM [Ruosch et al., 2022], a **b**enchmark for **A**rgument **M**ining (AM). It is aimed at interested parties and prospective users. We hope to provide a concise overview of what is presented in the original paper[1] with a more practical use in mind.

In Section 2, we first give a definition of AM to foster a common understanding. Then, we follow along the AM pipeline to illustrate how it works with an example. Section 3 describes the concept of our benchmark and the theory behind it. The measures used in the evaluation of different argument miners is discussed in Section 4. Section 5 discusses the aligning of different argumentation models in order to produce comparable results. In Section 6, we detail the implementation of BAM and provide a guide on how to get started putting it to use. Finally, Section 7 provides the concluding remarks.

If you have any questions, please feel free to ask.

2 Argument Mining

Argument Mining (AM) can entail different tasks and purposes. For our intent, we adopt the well-established information extraction approach. It was popularized by several experts in the field [Saint Dizier, 2020, Budzynska and Villata, 2015, Lippi and Torroni, 2016]: a multistage pipeline that extracts the arguments present in a text by first separating non-argumentative from argumentative units, then classifying the argument components and, finally, identifying their structure with relations [Stab et al., 2014]. Figure 1 illustrates the pipeline.

We will now go through its stages with a working example to explain the individual steps. The raw text used as input is the following:

> Everybody should study abroad. I really enjoyed my time in Asia. It is an irreplaceable experience because you learn living without depending on anyone else. However, there were also certain struggles. You will experience loneliness, living away from family and friends.

For consistency and simplicity, we also adopt the *claim/premise* model [Walton, 2009] for the example annotation and employ two types of relations: *attacks* and *supports*. In this representation, arguments consist of two kinds of components. A *claim* is

[1] https://ceur-ws.org/Vol-3164/paper5.pdf

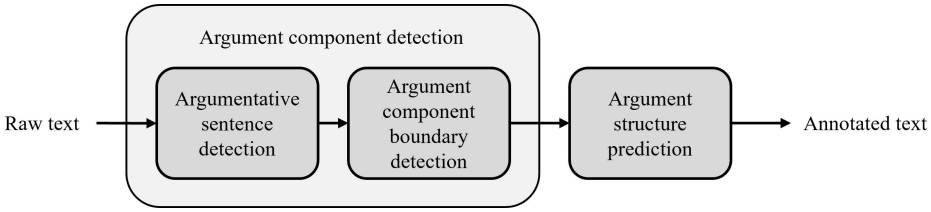

Figure 1: The AM pipeline adapted from [Lippi and Torroni, 2016].

the central statement of an argument. An example from the text above for a claim would be the proposition that "everybody should study abroad". *Premises* (also known as *data*, *evidences*, *grounds*, or *preconditions* [Lauscher et al., 2018]) are about the plausibility of the *claim*, e.g., the assertion that "[the studying abroad] is an irreplaceable experience". These components may be linked by a relation which can be an *attack* (component a undermines component b) or *support* (component a backs component b). Explicitly, we do not restrict neither the domain nor the range of the relations to certain types of components. This means that, in practice, a *claim* may *support* or *attack* another *claim*.

Argumentative Sentence Detection

The first stage of the pipeline aims to identify the argumentative sentences. A sentence is classified as argumentative, if it contains any argument component [Lippi and Torroni, 2016]. Thus, the subsequent steps only consider those sentences. In our example, we indicate non-argumentative sentences with strike through (e.g., "This is argumentative. ~~This is not.~~"), leaving the others to be argumentative.

> Everybody should study abroad. ~~I really enjoyed my time in Asia.~~ It is an irreplaceable experience because you learn living without depending on anyone else. ~~However, there were also certain struggles.~~ You will experience loneliness, living away from family and friends.

The argumentative nature of propositions may depend on their context. One of the simplest clues for the presence of an argument (component) is the presence of discourse indicators [Lawrence and Reed, 2015]. Key words such as *because* or *despite* may not only indicate components but also determine relations between them.

Argument Component Boundary Detection

After identifying the sentences containing argument components, we now need to identify their explicit boundaries since a component may not necessarily coincide exactly with the entire length of a sentence. The specifics of what belongs (and what does not belong) to an argument component depends on the annotation guidelines, such as the in- or exclusion of punctuation marks. Instead of indicating the boundaries, we highlight argument components (e.g., "this is a component, while this is not").

> Everybody should study abroad. ~~I really enjoyed my time in Asia.~~ It is an irreplaceable experience because you learn living without depending on anyone else. ~~However, there were also certain struggles.~~ You will experience loneliness, living away from family and friends.

For our example, we opt to exclude punctuation marks as well as discourse indicators. This changes the explicitly annotated component boundaries.

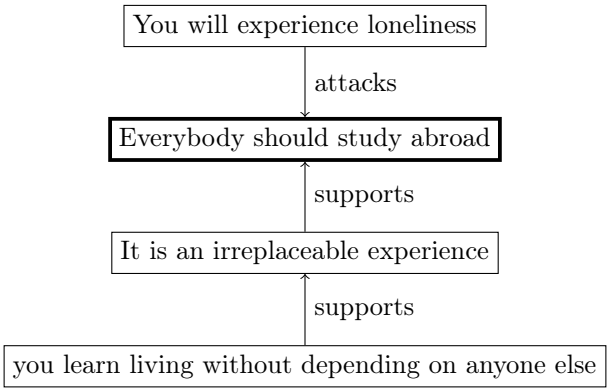

Figure 2: Output of the AM pipeline.

Argument Component Detection

The third step assigns the previously identified argumentative text spans a type from a defined set (i.e., *claim* or *premise* in our case). We visualize the classification with different types of highlighting: claims like this and premises like that.

> Everybody should study abroad. ~~I really enjoyed my time in Asia.~~ It is an irreplaceable experience because you learn living without depending on anyone else. ~~However, there were also certain struggles.~~ You will experience loneliness, living away from family and friends.

It is sometimes difficult to assign a proposition to either *claim* or *premise* [Lauscher et al., 2018]. This decision also heavily depends on the context as well as the specifics of the chosen argument representation.

Argument Structure Prediction

To form an argument, we now predict the structure (i.e., relations) of the components. We use two different relations to build triples, which in turn then form a graph: $\xrightarrow{attacks}$ and $\xrightarrow{supports}$. Based on the previously identified components, we can connect them with the following relations:

- "You will experience loneliness" $\xrightarrow{attacks}$ "Everybody should study abroad"
- "It is an irreplaceable experience" $\xrightarrow{supports}$ "Everybody should study abroad"
- "you learn living without depending on anyone else" $\xrightarrow{supports}$ "It is an irreplaceable experience"

Figure 2 shows the output of the AM pipeline explicitly modeled as a graph. The thick border signifies the sole *claim* in the graph with the other nodes representing *premises*. The edges are the relations. This demonstrates the process of mining arguments from raw text as input.

3 BAM: Benchmark for Argument Mining

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 3 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.[2]

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

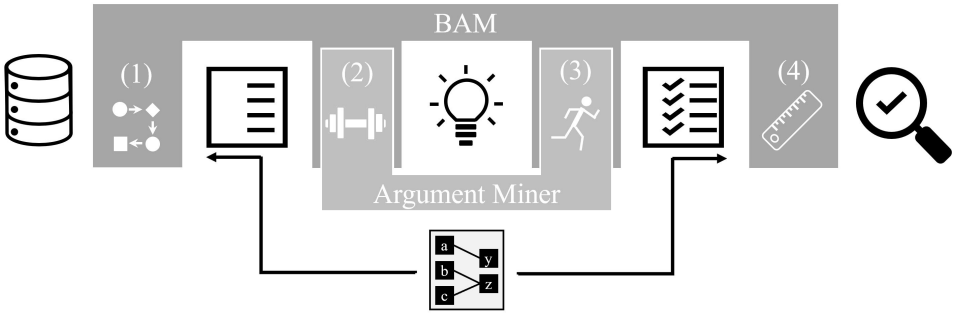

Figure 3: System vision of the different parts of the benchmark framework and their interactions.

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 [Loper and Bird, 2002] tokenizer, where 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 [Honnibal et al., 2020] implementation of the token aligner[3] and, thus, all of the evaluation happens uniformly on token-level. Subsequently, several aspects are evaluated. Based on the AM pipeline described in [Lippi and Torroni, 2016] (see Figure 1), 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 [Lippi

[3] https://github.com/explosion/spacy-alignments

and Torroni, 2016]. 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 3) 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[4]), we not only ensure that it is machine-readable and interoperable, but we also facilitate its extension and reuse.

4 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 [Lippi and Torroni, 2016]. 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[5] 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., 2016] and use the implementation[6] of the segmentation evaluation [Fournier, 2013]. 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) [Al Khatib et al., 2021], we can employ the nervaluate-package[7] originally designed for the evaluation of NER [Segura-Bedmar et al., 2013]. 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 in [Duthie et al., 2016], 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

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

Task	Measure	Reference
Sentence classification	micro-F1	[van Rijsbergen, 1979]
Boundary detection	boundary similarity	[Fournier, 2013]
Component identification	NER F1	[Segura-Bedmar et al., 2013]
Relation prediction	F1	[van Rijsbergen, 1979]

Table 1: Overview of the different AM tasks and the respective measures.

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, 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.

Table 1 provides an overview of the four AM tasks and their respective evaluation measures.

5 Aligning Argumentation Models

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 [Lippi and Torroni, 2016] of the *claim/premise* model [Walton, 2009] 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.

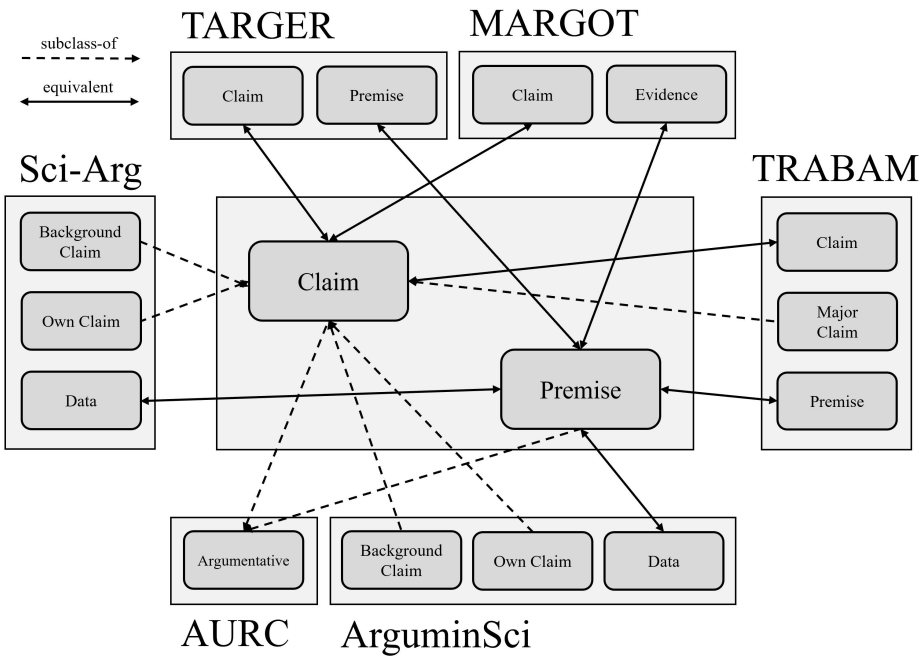

Figure 4: Mapping between different argumentation schemes for the *components*.

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* [Lippi and Torroni, 2016]. 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.

To illustrate the alignment with some examples, Figures 4 and 5 visualize the mapping for the schemes of the components and relations, respectively for a number of systems and the benchmark data set.

6 Implementation

The Python implementation of BAM is open-source and public.[8] It is structured in a modular fashion, to facilitate extensions and amendments. The data for the benchmark is contained in the `data` and the systems to be evaluated in the `AM` folder. Furthermore, the two central files in the implementation are `bam.py` and `evaluation.py`. The former is the main driver of BAM, invokes all necessary steps of the pipeline, and takes the system to be evaluated as the argument. The latter contains the evaluation routines which can be used by any system, provided the data passed conforms to the format requirements. Examples of how to convert the data are provided in the code.

To set up BAM, obtain the source code from the repository. Then, install the dependencies as specified in the file `requirements.txt`. This includes all packages necessary for the evaluation routine only, notably not for the individual systems. In order to benchmark a system, it first needs to be incorporated into the BAM pipeline (see Section 3 for details). Several examples are provided, which only need the code and models from the corresponding systems, as well as a "dummy system" (*noop*)

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

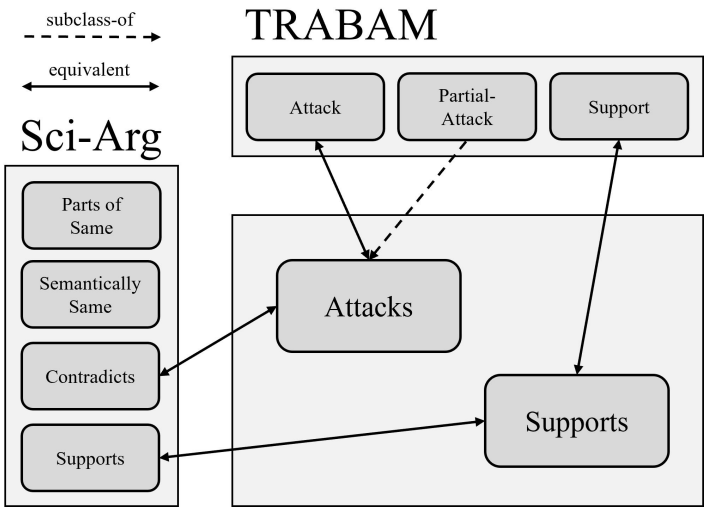

Figure 5: Mapping between different argumentation schemes for the *relations*.

which does not require any additional dependencies. Following the instructions and these examples, new systems can be added.

7 Conclusions

In this document, we summarized and explained several contributions of BAM, a benchmark for Argument Mining (AM) [Ruosch et al., 2022]. First, we explained our view of AM as an information extraction task. Then, we outlined the four stages of the AM pipeline (argument sentence detection, argument component boundary detection, argument component detection, and argument structure prediction) and illustrated them with a working example. This showed how to extract arguments from raw text and how they then form a graph using the identified structure. We model arguments as *claims* and *premises* [Walton, 2009] and connect them with *attacks* and *supports* relations. Then, we detailed the inner workings of BAM: built on four pillars (pre-processing, training, execution, and evaluation), it allows for the integration of any argument miner to be evaluated, and, more importantly, for the creation of comparable results. This is enabled by the newly proposed method of applying mappings to different argument models in order to unify to the *claim/premise*-model. Using a common view, we allow for comparable results of AM systems. We also explain the four measures that we apply, which correspond to the four AM pipeline stages. Subsequently, we lay out those evaluation measures in more detail and reason for our design choices using connections to other NLP tasks. Finally, we also point to our implementation of BAM and show how to install and use it.

References

[Al Khatib et al., 2021] Al Khatib, K., Ghosal, T., Hou, Y., de Waard, A., and Freitag, D. (2021). Argument Mining for Scholarly Document Processing: Taking Stock and Looking Ahead. In *Proceedings of the Second Workshop on Scholarly Document Processing*, pages 56–65. Association for Computational Linguistics.

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

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

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

[Honnibal et al., 2020] Honnibal, M., Montani, I., Van Landeghem, S., and Boyd, A. (2020). spaCy: Industrial-strength Natural Language Processing in Python.

[Lauscher et al., 2018] Lauscher, A., Glavas, G., Ponzetto, S. P., and Eckert, K. (2018). Annotating arguments in scientific publications.

[Lawrence and Reed, 2015] Lawrence, J. and Reed, C. (2015). Combining argument mining techniques. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 127–136.

[Lippi and Torroni, 2016] Lippi, M. and Torroni, P. (2016). Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology*, 16(2):1–25.

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

[Ruosch et al., 2022] Ruosch, F., Sarasua, C., and Bernstein, A. (2022). BAM: Benchmarking Argument Mining on Scientific Documents. In *Proceedings of the Workshop on Scientific Document Understanding co-located with 36th AAAI Conference on Artificial Intelligence (AAAI 2022)*.

[Saint Dizier, 2020] Saint Dizier, P. (2020). The lexicon of argumentation for argument mining: methodological considerations. *Anglophonia. French Journal of English Linguistics*, (29).

[Segura-Bedmar et al., 2013] Segura-Bedmar, I., Martínez Fernández, P., and Herrero Zazo, M. (2013). Semeval-2013 task 9: Extraction of drug-drug interactions from biomedical texts (ddiextraction 2013). Association for Computational Linguistics.

[Stab et al., 2014] Stab, C., Kirschner, C., Eckle-Kohler, J., and Gurevych, I. (2014). Argumentation mining in persuasive essays and scientific articles from the discourse structure perspective. *CEUR Workshop Proceedings*, 1341(1999).

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

[Walton, 2009] Walton, D. (2009). *Argumentation Theory: A Very Short Introduction*, pages 1–22. Springer US, Boston, MA.