# Is It Worth the Attention?
# A Comparative Evaluation of Attention Layers for
# Argument Unit Segmentation

**Maximilian Spliethöver**[*] and **Jonas Klaff**[*] and **Hendrik Heuer**
University of Bremen
Bibliothekstraße 1, 28359 Bremen, Germany
{mspl,joklaff,hheuer}@uni-bremen.de

## Abstract

Attention mechanisms have seen some success for natural language processing downstream tasks in recent years and generated new state-of-the-art results. A thorough evaluation of the attention mechanism for the task of Argumentation Mining is missing. With this paper, we report a comparative evaluation of attention layers in combination with a bidirectional long short-term memory network, which is the current state-of-the-art approach for the unit segmentation task. We also compare sentence-level contextualized word embeddings to pre-generated ones. Our findings suggest that for this task, the additional attention layer does not improve the performance. In most cases, contextualized embeddings do also not show an improvement on the score achieved by pre-defined embeddings.

## 1 Introduction

Argumentation Mining (AM) is increasingly applied in different fields of research like fake-news detection (Cabrio and Villata, 2018) and political argumentation and network analysis (Haunss et al.).

One crucial part of the AM pipeline is to segment written text into argumentative and non-argumentative units. Recent research in the area of unit segmentation (Eger et al., 2017; Ajjour et al., 2017) has lead to promising results with F1-scores of up to 0.90 for in-domain segmentation (Eger et al., 2017). Nevertheless, there is still a need for more robust approaches.

Given the recent progress of attention-based models in Neural Machine Translation (NMT) (Bahdanau et al., 2014; Vaswani et al., 2017), this paper evaluates the effectiveness of seperate attention layers for the task of argumentative unit segmentation. The idea of the attention layers added

to the recurrent networks is to preprocess the input data and enable the model to prioritize those parts of the input sequence that are important for the current prediction (Bahdanau et al., 2014). This can be achieved by learning additional parameters during the training of the model. With the additional information gained, the model learns a better internal representation which improves performance.

Additionally, we evaluate the impact of contextualized distributed term representations (also referred to as word embeddings hereinafter) on all our models. The goal of word embeddings is to represent a word as a high-dimensional vector that encodes its approximate meaning. This vector will be generated by a model trained on a language modeling task, like next-word prediction (Mikolov et al., 2013), for a given text corpus. The representation is based on the word's surrounding context in the corpus. Words with a similar semantic meaning should then also have similar vector representations, as measured by their distance in the vector space (Sahlgren, 2005, 2006; Firth, 1957; Heuer, 2015). Different methods to pre-compute the embeddings include word2vec (Mikolov et al., 2013), FastText (Bojanowski et al., 2017) and GloVe (Pennington et al., 2014). To make use of the capabilities of pre-trained Language Models (LMs), such as BERT (Devlin et al., 2018) or Flair (Akbik et al., 2018), we evaluate how well their semantic representations perform, by using contextualized word embeddings. Those are, in contrast to previously mentioned methods, specific to the context of the word in the input sequence. One major benefit is the fact that the time-consuming feature engineering could become obsolete since the features are implicitly encoded in the word embeddings. Furthermore, a better semantic representation of the input could lead to better generalization capabilities of the model and, therefore, to

---

[*] The first two authors contributed equally. Their listing order is random.

better cross-domain performance.

This paper answers the following research questions, which will help to assess the importance of the attention layers and contextualized word embeddings for the argument unit segmentation task:

- **RQ1:** To what extent can seperate attention layers help the model focus on the, for the task of unit segmentation relevant, sequence parts and how much do they influence the predictions?

- **RQ2:** What is the impact of contextualized distributed term representations like BERT (Devlin et al., 2018) and Flair (Akbik et al., 2018) on the task of unit segmentation and do they improve upon pre-defined representations like GloVe?

The contributions of this paper are as follows: first, we present and evaluate new attention-based architectures for the task of argumentative text segmentation. Second, we review the effectiveness of recently proposed contextualized word embedding approaches in regard to AM. We will continue by presenting the previous work on this specific task, followed by a description of the different architectures used, the data set and the generation of the word embeddings. Afterwards, we will report the results, followed by a discussion and the limitations. We will finish with a conclusion and an outlook on possible future work.

## 2 Related Work

Attention mechanisms have long been utilized in deep neural networks. Some of its roots are in the salient region detection for the processing of images (Itti et al., 1998), which takes inspiration from human perception. The main idea is to focus the attention of the underlying network on points-of-interest in the input that are often surrounded by irrelevant parts (Mnih et al., 2014). This allows the model to put more weight on the important chunks. While earlier salient detectors were task-specific, newer approaches (e.g. Mnih et al., 2014) can be adapted to different tasks, like image description generation (Xu et al., 2015), and allow for the parameters of the attention to be tuned during the training. These additional tasks include sequence processing and the application of such networks to different areas of Natural Language Processing (NLP). One of the first use-cases for attention mechanisms in the field of NLP was machine translation. Bahdanau et al. (2014) utilized the attention to improve their NMT model. Vaswani et al. (2017) achieved new State-of-the-Art (sota) results by presenting an encoder-decoder architecture that is based on the attention mechanism, only adding a position-wise feed-forward network and normalizations in between. Devlin et al. (2018) picked up on the encoder part of this architecture to pre-train a bidirectional LM. After fine-tuning, they achieved a new sota performance on different downstream NLP tasks like part-of-speech tagging and questions-answering.

A possible way of posing the unit segmentation as NLP task is a token-based sequence labeling (Stab, 2017). While Tobias et al. (2018) used non-recurrent classifiers to approach this problem, others mostly applied recurrent networks to the task of unit boundary prediction. For example, Eger et al. (2017) reported different long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) architectures. Further, Ajjour et al. (2017) proposed a setup with three bidirectional LSTMs (Bi-LSTMs) (Schuster and Paliwal, 1997) in total as their best solution. While the first two of them are fully connected and work on word embeddings and task-specific features respectively, the intention for the third is to take the output of the first two as input and learn to correct their errors. Even though the third Bi-LSTM did not improve on the F1-score metric, it did succeed in resolving some of the wrong consecutive token predictions, without worsening the final results.

To the best of the authors' knowledge, the attention mechanism has not been widely utilized for the task of argumentative unit segmentation. Stab et al. (2018) integrated the attention mechanism directly into their Bi-LSTM by calculating it at each time step $t$ to evaluate the importance of the current hidden state $h_t$. To do that, they employed additive attention. A similar approach has been applied by Morio and Fujita (2018) for a three-label classification task (claim, premise or non-argumentative).

While a direct integration of the attention mechanism is able to take the previous state of the Bi-LSTM into the calculation, it seems less trivial to implement with the current available programming frameworks. In contrast, the approach presented in this paper uses attention as a separate layer that encodes all sequences before they are fed into a Bi-LSTM. This might enable the recur-

rent parts of the network to learn from better representations that are specific to the task they were trained on. The aim is further to evaluate the possible applications of attention layers for the task of sequence segmentation and token classification. A recurrent architecture (Ajjour et al., 2017) is compared to multiple modified versions that utilize the aforementioned attention mechanism.

In order to derive a representation of the input text that better resembles the context of the input for a specific task, several approaches have been presented. Akbik et al. (2018), for example, pre-train a character-level Bi-LSTM to predict the next character for a given text corpus. The pre-trained model is able to derive contextualized word embeddings by additionally utilizing the input sequence for a specific task. This allows the system to encode the preceding and following words of the given input sequence into the word representation. In comparison to that, the pre-trained BERT-LM utilizes stacked attention layers (Vaswani et al., 2017). By feeding a sequence into it and extracting the output of the last layer for each token, the idea is to implicitly use the attention mechanism to derive a better representation for every token. As is the case for the character-wise LM from Akbik et al. (2018), the BERT embeddings are contextualized by the whole input sequence of the specific task.

This paper will compare the two contextualized approaches described above with the pre-defined GloVe (Pennington et al., 2014) embeddings in the light of their usefulness for AM. The goal is to encode the features necessary to detect arguments by utilizing the context of a sentence.

# 3 Methodology

This paper evaluates different machine learning architectures with attention layers for the task of AM, and more specifically unit segmentation. The problem is framed as a multi-class token labeling task, in which each token is assigned one of three labels. A (B) label denotes that the token is at the beginning of an argumentative unit, an (I) label that it lies inside a unit and an (O) label that the token is not part of a unit. This framework has been applied previously for the same task (Stab, 2017; Eger et al., 2017; Ajjour et al., 2017).

The architectures proposed in this section build on Ajjour et al. (2017), omitting the second Bi-LSTM, which was used to process features

other than word embeddings (see section 3.3). They are further being modified by adding attention layers at different positions. The goal is to reuse existing approaches and possibly enhance their ability to model long-range dependencies. Additionally, a simpler architecture, consisting of a single Bi-LSTM paired with an attention layer, is built and evaluated with the aim of reduced complexity.

In order to answer the second research question, this paper reports results in combination with improved input embeddings, in order to evaluate their effectiveness and impact on the AM downstream task.

All models are compared to the modified re-implementation of the architecture, which is defined as the baseline architecture.

## 3.1 Models

In order to evaluate the attention mechanisms, different architectures based on previous AM literature are implemented. The attention layer is added at different positions in the network.

All models were implemented using Python and the Keras framework with a TensorFlow backend. For the self-attention and multi-head attention layers, an existing implementation is used (HG, 2018a,b). The difference between the two is that the multi-head attention divides the input into multiple chunks and each head therefore works on a different vector subspace (Vaswani et al., 2017), while the self-attention works on the whole input sequence. This is supposed to allow the head to focus on specific features of the input. In this case, the self-attention layers use additive attention, while the multi-head attention layers use scaled dot-product attention, with the latter following the implementation of Vaswani et al. (2017).

**Baseline re-implementation** The baseline model from Ajjour et al. (2017) uses a total of three Bi-LSTMs (two of them fully connected) to assign labels to tokens (see Figure 1a). The re-implementation does not include the two fully connected Bi-LSTMs but instead uses only a single one that works on the word embeddings (see Figure 1b). Due to the fact that the second Bi-LSTM in the first layer is only used to encode the non-semantic features like part-of-speech tags and discourse marker labels, it is omitted in the re-implementation. Hereafter, we will refer
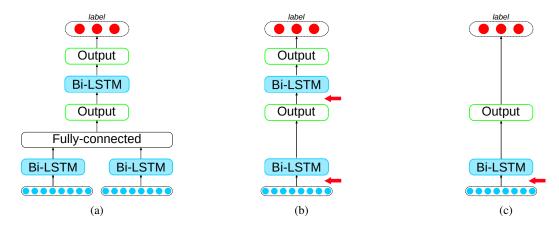
Figure 1: *(a)* The original baseline architecture as reported by Ajjour et al. (2017). *(b)* The modified baseline architecture without the second input Bi-LSTM. The bold arrows show the positions at which the additional attention layers are added to build the **Baseline**$_{+input}$ and **Baseline**$_{+error}$ architectures. *(c)* The **Bi-LSTM** architecture incorporates only one Bi-LSTM. The bold arrow shows the position at which the additional attention layer is added to build the **Bi-LSTM**$_{+input}$ architecture.

to this model as **Baseline**. Also, the batch size was increased from 8 to 64, compared to the original implementation, as a trade-off between convergence time and the model's generalization performance (Keskar et al., 2016). Nevertheless, this model achieves comparable scores to the ones presented in the original paper. The slightly lower performance can probably be attributed to implementation details.

**Baseline**$_{+input}$ **and Baseline**$_{+error}$    For both variations, the architecture shown in in Figure 1b was used as a basis. Multi-head-attention layers are added at different positions in the network. The number of attention heads depends on the dimension of the embedding vectors. For the GloVe (300 dimensions) and the BERT (3072 dimensions) embeddings, six heads are used, while the Flair (4196 dimensions) embeddings require four heads. Both numbers were the largest divisor for the respective input vector size that worked inside the computational boundaries available. In the first model, an attention layer was added before the first Bi-LSTM in an attempt to apply a relevance score directly to the tokens, in order to better capture dependencies of the input sequence. This model will be referred to as **Baseline**$_{+input}$. The second variation adds the attention layer after the first and before the second Bi-LSTM, which will be called **Baseline**$_{+error}$. According to Ajjour et al. (2017), the latter Bi-LSTM is used to correct the errors of the first one. The attention layer should be able to support the model in the error correction process. In contrast to the first approach, this does

not change the input data, but only works on the output of the first Bi-LSTM.

**Bi-LSTM and Bi-LSTM**$_{+input}$    To decrease the complexity of the architecture, two additional models with a single Bi-LSTM are trained. The first variant has no attention layer, while the second one utilized the same input attention described above (see Figure 1c). They will be refered to as **Bi-LSTM** and **Bi-LSTM**$_{+input}$ respectively. Both architectures use a self-attention mechanism instead of the above-mentioned multi-head-attention, due to better results in preliminary tests.

## 3.2   Data

The different architectures were trained and evaluated on the "Argument annotated Essays (version 2)" corpus (also referred to as Persuasive Essays corpus) (Stab and Gurevych, 2017). It was utilized for the same task in previous literature (Ajjour et al., 2017; Eger et al., 2017).

The corpus, compiled for parsing argumentative structures in written text, consists of a random sample of 402 student essays. The annotation scheme includes the argumentative units and the relations between them, as well as the major claim and stance of the author towards a specific topic. The texts were annotated by non-professionals, labeling the boundary of each argumentative unit alongside the unit type. A type can either be major-claim, claim or premise. For the unit segmentation task, the corpus is labeled by treating major claims, claims, and premises as argumen-

tative units[1]. For comparability reasons in the evaluation process, the models are trained and tested with the train-test-split defined by Stab and Gurevych (2017). The development set was composed of the last 20 percent of the training set and shuffled before use.

### 3.3 Features

For each token, a set of three different embeddings is generated and compared regarding their capability as standalone input features. The resulting weighted F1-score is then used as a proxy for measuring the usefulness of the generated text-representation in light of this specific downstream task.

In combination with the re-implemented architecture, the word embeddings approach GloVe (Pennington et al., 2014), trained on 6 billion tokens, serves as the baseline.

As a first approach to enhance the performance, the GloVe embeddings are stacked with the character-based Flair embeddings (Akbik et al., 2018), which are generated by a Bi-LSTM model. Akbik et al. (2018) argue that the resulting embeddings are contextualized, since the LM was trained to predict the most probable next character and therefore to encode the context of the whole sequence.

Similar to that, we also compare contextualized BERT-embeddings as standalone features (Devlin et al., 2018). An increased performance is expected because of the pre-training procedure of the LM. The BERT-LM was trained to predict a (randomly masked) word by utilizing the context of its appearance, as well as on next sentence prediction. Due to its sota performance for both, token-level and sentence-level tasks, the authors of this paper argue that the derived representations are well suited for the task of unit segmentation. Also, the representation fits the needs of the inter-token and sentence dependencies of the task. It is expected that this enables the model to better grasp the notion or pattern of an argument. Both contextualized embeddings are generated using the Flair library (Zalando Research, 2018).

For the of the BERT-embeddings the "bert-base-uncased" LM, consisting of 12-layers and pre-trained on lowercased data, is used. At the time

---

| Model | GloVe | BERT | Flair |
|---|---|---|---|
| **Baseline** | 0.86 | 0.83 | **0.87** |
| | | | |
| **Baseline**$_{+input}$ | **0.85** | 0.68 | 0.67 |
| **Baseline**$_{+error}$ | 0.67 | **0.68** | 0.67 |
| | | | |
| **Bi-LSTM** | **0.86** | 0.86 | 0.86 |
| **Bi-LSTM**$_{+input}$ | **0.84** | 0.83 | 0.81 |

Table 1: The weighted F1-scores for the **Baseline** and all four variations. Results are shown per variation and embedding. Each row shows the performance of one architecture with different word embeddings as input vector. The highest score for each architecture is marked in bold.

of writing, the Flair library extracts the representations for the first subword token from the last four layers of the pre-trained BERT model. The subtoken embeddings is then used as representation for the whole token. Features specifically engineered for this task are not included in the input, following the argumentation of Eger et al. (2017) that they will probably not be generalizable to different data sets.

## 4 Results

We evaluate the performance of all architectures on the Persuasive Essays data set detailed above. The models are re-initialized after every evaluation and do not share any weights. This allows us to answer the first research question of whether additional attention layers have a positive impact on the prediction quality.

To answer the second research question, we re-run each training, replacing the GloVe embeddings with BERT and Flair embeddings. Both contextualized embedding methods are tested separately. We contextualize the tokens on the sentence level since the BERT model (Devlin et al., 2018) only allows for a maximum input length of 512 characters. This makes document-level or paragraph-level embeddings impractical for the data set.

As a performance measure, we report the weighted F1-score instead of the macro F1-score, since it takes the imbalance of the samples per label into account.

For our re-implementation of the baseline, we are able to approximately reproduce the results reported by Ajjour et al. (2017). Additionally, we can verify that there is no major change in the performance when adding a second Bi-LSTM to

---

[1]All data pre-processing scripts are available in our code repository: https://gitlab.informatik.uni-bremen.de/covis1819/worth-the-attention.

the network (compare results for **Bi-LSTM** and **Baseline** in Table 1).

### 4.1 Attention Layers

The results of the token classification task are presented in Table 1. Generally speaking, the added attention encodings do not improve upon the original architecture's performance, no matter at which position they are added. Architectures with an input attention encoding, namely $\textbf{Baseline}_{+input}$ and $\textbf{Bi-LSTM}_{+input}$, do achieve similar performances compared to their respective baseline. But the F1-score performance is in strong contrast to the generalization error, which is in most cases lower for the **Baseline** model.

The $\textbf{Baseline}_{+error}$ architecture, on the other hand, which is supposed to help the second Bi-LSTM in the network to correct the errors made by the first one, performs worse across all tests. For the Flair embeddings, this results in a 0.20 points performance drop in the F1-score measure.

### 4.2 Contextualized Word Embeddings

The results for the enhanced word embedding evaluations are reported in Table 1. In some cases, the models utilizing the word embeddings generated by the BERT-LM achieve a lower performance score than the other embeddings. This drop is most noticeable for the $\textbf{Baseline}_{+input}$ model, while the performance for the $\textbf{Bi-LSTM}_{+input}$ decreases only slightly. The $\textbf{Baseline}_{+error}$ model is able to achieve results that outperform both, GloVe and Flair embeddings.

Compared to the GloVe vectors, the models trained on the Flair embeddings mostly lose in F1-score performance as well. For example, the $\textbf{Baseline}_{+input}$ model drops by 0.18. On the other hand, the **Baseline** model is able to slightly improve upon the GloVe score using the Flair embeddings, achieving a final score of 0.87, which also marks the best overall score in our testings.

An interesting observation is the fact that the enhanced embeddings seem to increase the generalization error (compare Figure 2). The **Baseline** model trained on the GloVe embeddings, for example, shows a difference in the final validation and training loss of around 0.17 and increases for the BERT and Flair embeddings to roughly 0.60 and 0.48, respectively.

## 5 Discussion

Given the experimental results, we discuss the resulting implications for our two research questions and conclude this section by presenting some limitations.

### 5.1 Attention Layers

Our results suggest that the attention encoding does not increase the performance of the model, as we hypothesized above. This is true for both, the input and the error encoding. A potential explanation is the fact that we use the attention mechanism as an additional layer to encode the input. Other approaches, like Morio and Fujita (2018) or Stab et al. (2018), incorporate it into the Bi-LSTM architecture and calculate the weight of the hidden states at every time step.

While the performance does not decrease meaningfully for the $\textbf{Baseline}_{+input}$ and $\textbf{Bi-LSTM}_{+input}$ models (using the GloVe embeddings as features), it does for the error encoding $\textbf{Baseline}_{+error}$ model. This drop might be explained by the vector space the attention mechanism is working on. Due to its small size of only four features, it is unlikely that the resulting vector has a meaningful encoding.

A deeper inspection of the output values from the different layers in the network and how they influence the overall classification task might give more insight into the cause of the problem.

### 5.2 Contextualized Word Embeddings

For most of the tests we conduct, the contextualized embedding approaches do not improve upon the GloVe embeddings. This is especially true for the architectures that include an attention layer, which does not seem to be able to handle the encoding of high dimensional vectors very well. The results further suggest that the amount of neurons in the Bi-LSTMs is not an issue in this case, since the **Baseline** model achieves comparable results across all three embeddings.

A potential way to improve the results of the enhanced embeddings is to contextualize them on the paragraph level. While we contextualize them on a sentence level, the dependencies between arguments might span over multiple sentences, sometimes even a paragraph, as described by Stab and Gurevych (2017) for the Persuasive Essays data set. Following this reasoning, one might think that a document level contextualization makes sense
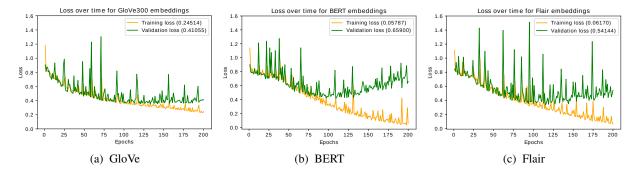
| (a) GloVe | (b) BERT | (c) Flair |

Figure 2: The loss curves of the **Baseline** architecture using different input embeddings. Figure *(a)* shows the training process of the model using the GloVe embeddings, while the model in Figure *(b)* used the BERT embeddings and Figure *(c)* the Flair embeddings. The orange line shows the training loss, the green line the validation loss.

and adds even more information to the embedding. For the task of AM, however, we argue against this for two reasons. First, argumentative units usually do not span over the whole document and it might include additional counter-arguments (Stab and Gurevych, 2017). The contextualization would most likely cause a lot of noise and make the vector less useful. Also, depending on the size of the document, the size of the vector might be too small to hold the contextual information of the full document. Second, the model trained on such embeddings would probably not generalize very well. An argumentative document can be written in different formats with different purposes, like an essay, a speech or a newspaper article. Contextualizing the embeddings on the document level might then also encode the structure of the text and decrease the cross-domain applicability of the model. However, further research is needed.

### 5.3 Limitations

The results we report and analyze above are the networks' performance as validated on the data splits provided by Stab and Gurevych (2017). Due to time and resource restrictions, we evaluate the results after a single training run and perform neither an averaging over multiple runs nor any cross-validation. Both could lead to more robust results. As another consequence of the above-mentioned restrictions, we are also not able to test the model's generalization capabilities on different data sets.

For the learning rate, we perform only a basic Bayesian hyperparameter optimization (Snoek et al., 2012) with four iterations per model. These limitations are especially important for the variations of the **Baseline** architecture, since the performed changes to the architecture, even though

rather small, entail the need for independently tuned hyperparameters.

Furthermore, an additional evaluation of the different contextualization levels for the embeddings could provide a clearer picture of how much the results actually improve, compared to non-contextualized methods.

## 6  Conclusion

Recent improvements in utilizing contextual information for sequence processing had a big impact on the area of NLP, namely advances of attention architectures and contextualized word embeddings. For example, the Transformer architecture (Vaswani et al., 2017) employs attention to achieve sota scores on different NLP tasks. Further, the Flair model (Akbik et al., 2018) incorporates character-wise context to generate enhanced word representations.

In this paper, we report on the usefulness of these two approaches for the task of AM. First, we are able to show that an attention layer as additional encoding of the input does not improve upon the current sota approach of a Bi-LSTM. Additionally, the attention mechanism seems to fail for a low-dimensional vector space. Second, we present the impact of contextualized word embeddings for AM. Although the Flair embeddings slightly improve upon the performance of the GloVe embeddings for the **Baseline** architecture, we can not confirm any advantage over non-contextualized embeddings.

### 6.1  Future Work

A first extension of this work could be a proper hyperparameter optimization for the attention-based models. Second, we plan to explore an attempt to

80

fine-tune solely attention based pre-trained models like BERT (Devlin et al., 2018) to domain-specific data. Recent research by Howard and Ruder (2018) in transfer-learning for NLP has shown great improvement for several NLP-downstream tasks, while reducing the needed amount of labeled training data.

Third, we contextualize the embeddings on the sentence level only. According to Stab and Gurevych (2017), arguments can sometimes span over multiple sentences. Therefore, the contextualization of the embeddings could be extended to a paragraph level, in order to make use of possible inter-dependencies within it. Additionally, a fine-tuning approach of the underlying LMs to the AM task could further enhance the embeddings.

## Acknowledgments

## References

Yamen Ajjour, Wei-Fan Chen, Johannes Kiesel, Henning Wachsmuth, and Benno Stein. 2017. Unit Segmentation of Argumentative Texts. In *Proceedings of the 4th Workshop on Argument Mining*, pages 118–128, Copenhagen, Denmark. Association for Computational Linguistics.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual String Embeddings for Sequence Labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. arXiv: 1409.0473.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Elena Cabrio and Serena Villata. 2018. Five Years of Argument Mining: a Data-driven Analysis. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 5427–5433, Stockholm, Sweden. International Joint Conferences on Artificial Intelligence Organization.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv: 1810.04805.

Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural End-to-End Learning for Computational Argumentation Mining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11–22, Vancouver, Canada. Association for Computational Linguistics.

J. R. Firth. 1957. A synopsis of linguistic theory 1930-55. pages 1–32.

Sebastian Haunss, Jonas Kuhn, Sebastian Padó, and Nico Blokker. MARDY: Modeling ARgumentation DYnamics in Political Discourse. https://www.socium.uni-bremen.de/projekte/?proj=570&print=1, last accessed: 2019-06-04, 16:48UTC+2.

Hendrik Heuer. 2015. Semantic and stylistic text analysis and text summary evaluation. Master thesis.

Zhao HG. 2018a. Attention mechanism for processing sequential data that considers the context for each timestamp. https://github.com/CyberZHG/keras-self-attention, last accessed: 2019-05-01, 21:39UTC+2.

Zhao HG. 2018b. A wrapper layer for stacking layers horizontally. https://github.com/CyberZHG/keras-multi-head, last accessed: 2019-05-01, 21:40UTC+2.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Laurent Itti, Christof Koch, and Ernst Niebur. 1998. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. arXiv: 1609.04836.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv: 1301.3781.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. 2014. Recurrent Models of Visual Attention. arXiv: 1406.6247.

Gaku Morio and Katsuhide Fujita. 2018. End-to-End Argument Mining for Discussion Threads Based on Parallel Constrained Pointer Architecture. In *Proceedings of the 5th Workshop on Argument Mining*, pages 11–21, Brussels, Belgium. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Magnus Sahlgren. 2005. An introduction to random indexing. In *Methods and applications of semantic indexing workshop at the 7th international conference on terminology and knowledge engineering*, volume 5. TKE.

Magnus Sahlgren. 2006. *The Word-Space Model: using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. Ph.D. thesis, Stockholm University.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45:2673–2681.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2951–2959. Curran Associates, Inc.

Christian Stab and Iryna Gurevych. 2017. Parsing Argumentation Structures in Persuasive Essays. *Computational Linguistics*, 43(3):619–659.

Christian Stab, Tristan Miller, Benjamin Schiller, Pranav Rai, and Iryna Gurevych. 2018. Cross-topic Argument Mining from Heterogeneous Sources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3664–3674. Association for Computational Linguistics. Event-place: Brussels, Belgium.

Christian Matthias Edwin Stab. 2017. *Argumentative Writing Support by means of Natural Language Processing*. Ph.D. thesis, Technische Universität Darmstadt, Darmstadt.

Mayer Tobias, Cabrio Elena, Lippi Marco, Torroni Paolo, and Villata Serena. 2018. Argument Mining on Clinical Trials. *Frontiers in Artificial Intelligence and Applications*, pages 137–148.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6000–6010, USA. Curran Associates Inc. Event-place: Long Beach, California, USA.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057, Lille, France. PMLR.

Zalando Research. 2018. A very simple framework for state-of-the-art Natural Language Processing (NLP). https://github.com/zalandoresearch/flair, last accessed: 2019-05-01, 21:39UTC+2.