LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
LEHRSTUHL FÜR DATENBANKSYSTEME
UND DATA MINING

Bachelorarbeit
in Informatik

Multi-Task Argument Mining

Johanna Reiml

Aufgabensteller:	Prof. Dr. Thomas Seidl
Betreuer:	Michael Fromm
Abgabedatum:	10.02.2022

Declaration of Authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references.
This paper was not previously presented to another examination board and has not been published.

Munich, 10.02.2022

.....
Johanna Reiml

Abstract

Argument mining (AM) is the automatic identification and extraction of the structure of inference and reasoning expressed as arguments presented in natural language. In this thesis, we explore the relationship between various argument related tasks, and investigate whether multi-task learning can improve the performance when fine-tuning argument mining tasks on the pre-trained language model BERT. In particular, we argue that the domain of topic information plays an important role in argument mining, since it provides semantic and contextual information for an argument. For this, we explore various settings such as cross-topic and in-topic scenarios. To the best of our best knowledge, we are the first to compare AM task similarities by examining how individual tasks can benefit from each other, while also exploring the effect of topic information. We release our code on GitHub[1] and our results can be reproduced with ease.

[1] https://github.com/jreiml/bachelor-thesis-argument-mining

Contents

Chapter 1

Introduction

1.1 Motivation

This thesis mainly discusses the topic of *Multi-Task Argument Mining*. However, we also extend it to leveraging the knowledge of *Transfer Learning* in general.

Argument mining (AM) is the automatic identification and extraction of the structure of inference and reasoning expressed as arguments presented in natural language. [15] AM gives rise to various practical applications of great importance, such as modern recommender systems. [22] In this context, arguments are effectively recommendations used in an attempt to provide useful suggestions to a potential customer. AM has often been mentioned in the same context as sentiment analysis [10] [2], and at first glance the tasks look fairly similar. For sentiment analysis we classify if a given input is positive, negative, or neutral, whereas for argument mining we can classify an argument as supportive, contesting, or neutral. However, argument mining is much more challenging. In certain argument mining tasks, we do not only detect if something is an argument, but also try to infer more information based on the claim and premise of the argument.

In prior work, the main approach for argument mining was often to train a task-specific model, without paying special attention to related tasks. Such models reached promising results in their respective tasks, but failed to generalize well in unseen tasks. However, even within the same task it is highly subjective and conceptual what is or what is not considered an argument. [20]

In the past few years, pre-trained language models (LMs) have established themselves as a popular way to solve natural language processing (NLP) tasks.

Pre-training in a self-supervised manner before later fine-tuning on supervised downstream tasks is nowadays ubiquitous. This approach has contributed to the state-of-the-art on a wide range of tasks, including argument mining. The idea behind this process is called transfer learning. There are roughly four types of transfer learning in NLP. They are domain-adaption, cross-lingual learning, multi-task learning and sequential learning. [19]

To explore these ideas, we propose zero-short learning [29] [31] [30] and multi-task learning strategies for argument mining, in which we leverage transfer learning to train objectives across different tasks and different corpora. For this, we choose BERT as our pre-trained model and select Argument Identification (AId), Evidence Detection (ED) and Argument Quality (AQ) as the tasks of interest. Moreover, we argue that topic information may provide important context for an otherwise out-of-context argument. For this, we explore various settings such as cross-topic and in-topic scenarios. To the best of our knowledge, we are the first to compare AM task similarities by examining how individual tasks can benefit from each other, while also exploring the effect of topic information.

1.2 Thesis Structure

In chapter 2 we talk about the background and some related work for this thesis. Then, in chapter 3 we give insights into the datasets we choose for our experiments. We illustrate our methods in chapter 4. After that, we demonstrate our results and evaluate them in chapter 5. In the end, we summarize our findings for this thesis and provide further research directions for future work in chapter 6.

Chapter 2

Related Work

2.1 BERT

BERT is an acronym for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. [6] Since its emergence in 2018, it has achieved great success in NLP and established itself as a new baseline, beating eleven previous baseline models. To better understand the background and model architecture of BERT, we should recall its precursor and inspiration, the Transformer. [27]

2.1.1 Transformer

The Transformer was originally proposed as a sequence-to-sequence model [24] for machine translation. Subsequently, with the help of the Transformer, various tasks reached new state-of-the-art performance. This inspired numerous spin-off models, which were based on various aspects of the Transformer architecture. BERT is one of those spin-off models, and is based on the encoder mechanism of the Transformer.

The original Transformer consists of an encoder and a decoder, each of which is a stack of identical blocks. Each encoder block is composed of a multi-head self-attention module and a position-wise feed-forward network (FFN). [27] Around each sublayer module, there are residual connections [11] followed by a layer normalization. [1] The decoder block has a similar architecture to the encoder. However, each decoder block inserts cross-attention modules between the multi-head self-attention modules and the position-wise FFNs. Moreover, to prevent positions from attending to subsequent positions, there is a modification in the self-attention layer of the decoder.

In the following, we highlight the main contribution of the Transformer: the

attention mechanism.

2.1.1.1 Self-Attention

Self-attention allows the model to attend to each input at once, instead of just left-to-right or right-to-left. The Transformer realizes the self-attention mechanism with a Query-Key-Value (QKV) model. For each token's input vector, we create a query, key, and value vector. These vectors are obtained by multiplying our input vectors with the weights contained in the W^Q, W^K and W^V matrices. These weight matrices are learned in the training process. The token-wise feed-forward layer then independently processes each of these modified input token embeddings without any interaction among them.

The input of the self-attention layer is a token embedding matrix of the original input sequence, formally defined as $X = x_1, x_2, ..., x_n$. The output after the embedding layer is then described as $E \in \mathbb{R}^{|X| \times d}$ where d corresponds to the embedding dimension. For an encoder block size l, each block $\mathcal{F}$ receives an input matrix containing token embeddings $H^{l-1} \in \mathbb{R}^{|X| \times d}$ and outputs a new matrix $H(l) \in \mathbb{R}^{|X| \times d}$. For each self-attention layer, the Transformer generates three matrix representations of queries $Q \in \mathbb{R}^{|X| \times D_k}$, keys $K \in \mathbb{R}^{|X| \times D_k}$ and values $V \in \mathbb{R}^{|X| \times D_v}$. The output matrix after each self-attention layer is formally denoted as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

The resulting matrix is called the attention matrix $\mathcal{A}$.

2.1.1.2 Multi-Head Self-Attention

Instead of applying only one attention function at once, the Transformer applies multi-head attention in parallel. This is beneficial when linearly projecting the queries, keys and values h times. The multi-head self-attention layer is defined as the concatenation of all heads. Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions [27].

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Where $W_i^Q \in \mathbb{R}^{d \times d_k}$, $W_i^K \in \mathbb{R}^{d \times d_k}$, $W_i^V \in \mathbb{R}^{d \times d_v}$, $W^O \in \mathbb{R}^{hd_v \times d}$, $h = 8$.

2.1.2 Input Representations in BERT

BERT uses WordPiece embeddings [28] to represent each word piece in the corpus. For the input, the first token of each sequence is a special classification token [CLS], which is used as the aggregate representation for classification tasks in the final hidden state. BERT can also handle text pairs, by adding a special separation token ([SEP]) between them. To mark the boundaries between text pairs, BERT adds a special segment embedding for each token. For any given token, its input representation is constructed by summing the corresponding token, segment and position embeddings.

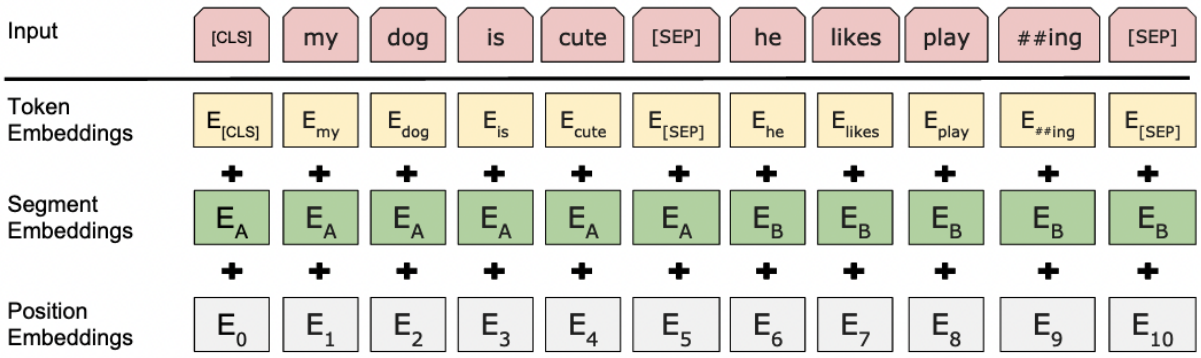


Figure 2.1: BERT input representation. Adopted from Devlin et al. [6], page 4.

2.1.3 Pre-training Objectives of BERT

BERT was proposed as a solution to the lack of deep bidirectionality in autoregressive language models. The huge advantage of this model is its ease of use, since one can simply add one output layer to the existing neural architecture. This way, one can obtain language models that surpass many previous approaches for NLP problems. As mentioned before, BERT is based on the encoder part of the Transformer. Unlike traditional pre-training strategies using left-to-right or right-to-left language models, BERT proposed two pre-training objectives: Masked Language Model and Next Sentence Prediction.

2.1.3.1 Masked Language Model

The idea behind Masked Language Model (LM) originated from the *Cloze* task in literature. During pre-training, BERT masks a random percentage of input

tokens and then predicts those masked tokens. This way, the model can obtain a bidirectional deep representation for the input tokens. In their experiments, the authors suggested randomly masking 15% of all WordPiece [21] tokens in each sequence. However, they do not simply replace all selected tokens with a special [MASK] token. That is because, if the specific [MASK] token is only part of the pre-training process, the model suffers from fine-tuning mismatch. In order to mitigate this problem, they replace masked tokens 80% of the time with [MASK], 10% of the time with another random token and 10% of the time they leave the token unchanged.

2.1.3.2 Next Sentence Prediction

To pre-train a model, which can understand the relationship between a sentence pair for NLP tasks such as Question Answering (QA), BERT applied Next Sentence Prediction as its second pre-training objective. Given a binarized next-sentence-prediction corpus, 50% of data samples are real sentence pairs. In the other 50% of samples, the first sentence is followed by a random sentence from the corpus. The model then has to correctly classify the real sentence pairs.

2.1.4 Fine-Tuning BERT

For fine-tuning BERT, a simple task-specific head is added to the pre-trained model, and all parameters are jointly fine-tuned on a downstream task. [6]

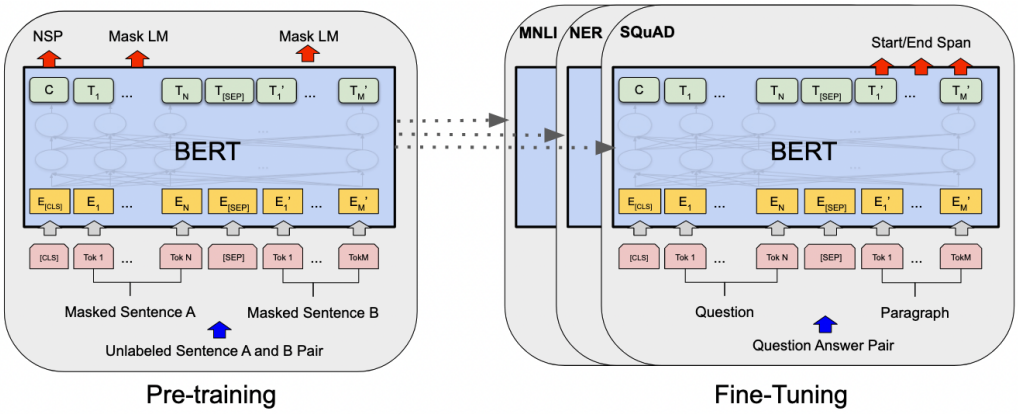


Figure 2.2: Overview pre-training and fine-tuning procedures for BERT. Adopted from Devlin et al. [6], page 3.

2.2 Multi-Task Learning

The most common way to approach NLP tasks, is to train a single model to perform one specific desired task. However, this way, we ignore potential other related tasks or training corpora, which could be helpful for the model's generalization performance. Multi-task learning (MTL) aims to improve learning efficiency and prediction accuracy by learning multiple objectives from a shared representation. To determine whether a model uses MTL methods, we can investigate the loss function of the model. If the model trains an objective by optimizing multiple loss functions in parallel, we are using multi-task learning. [18] This learning approach has been successfully applied to machine learning. This includes areas such as natural language processing [8], computer vision [3] and speech recognition [5].

There are two different MTL methods in Deep Learning: hard parameter sharing and soft parameter sharing. As for hard parameter sharing, all tasks share the same hidden layers during training, and only the output layers are task-specific. On the other hand, in soft parameter sharing, each task has its own model with its own parameters. Sharing is represented by calculating the distance between the parameters of the model. This distance is then regularized in order to encourage the parameters to be similar. [18]

In this thesis, we fine-tune BERT on multiple tasks at once by applying hard parameter sharing, i.e. that we do not explicitly train the model to have distinguished representations for different tasks. Instead, we let the model learn one objective on multiple tasks simultaneously.

2.3 Multi-Task Argument Mining

Compared to other NLP tasks, Argument mining (AM) is a relatively new field. Since the first ACL workshop on AM was held in 2014, it has gained more and more attention. At first, the focus for this research area was to create relevant corpora with proper annotations. Since then, multiple datasets have been released for a variety of argument mining related tasks. We touch upon some of these datasets in chapter 3. This spawned the interest of applying multi-task methods to the field of argument mining. To the best of our knowledge, we have not found a lot of work in this direction. The earliest work we found is from Schulz et al. [20]. In their research, they focused on argument component identification. They used MTL to greatly improve their results when compared to their single-task models. However, their research is relatively old, and does

not take advantage of modern pre-trained models like BERT. More recently, Tran et al. [26] explored MTL in the context of persuasive online discussions, in the hopes of improving argument mining by using MTL. They also use modern architectures like BERT, and report slightly improved results with the use of MTL. However, they argue that the tasks to be incorporated into MTL should be taken into consideration, as using all relevant tasks does not always lead to the best performance. Unlike us, both of these approaches do not explore the effect of topic information.

Chapter 3

Datasets

In this chapter, we look at the datasets we use to train all of our models. We look at the *IBM Debater – IBM-ArgQ-Rank-30kArgs* [9] (short: *IBM-Rank-30k*) dataset, the *IBM Debater – Evidences Sentences* [7] (short: *IBM Evidences Sentences*) dataset, and the *UKP Sentential Argument Mining Corpus* [23] (short: *UKP Sentential*) dataset. We introduce each dataset and explain how the data was collected and annotated. A quick overview can be seen in figure 3.1.

Name	Sentences	Topics	Task	Domain
IBM-Rank-30k [9]	30,497	71	Argument Quality	crowd collection
UKP-Sentential [23]	25,492	8	Argument Identification	web documents
IBM-Evidence [7]	29,429	221	Evidence Detection	Wikipedia

Table 3.1: Overview of the different Argument Mining datasets.

3.1 IBM-Rank-30k

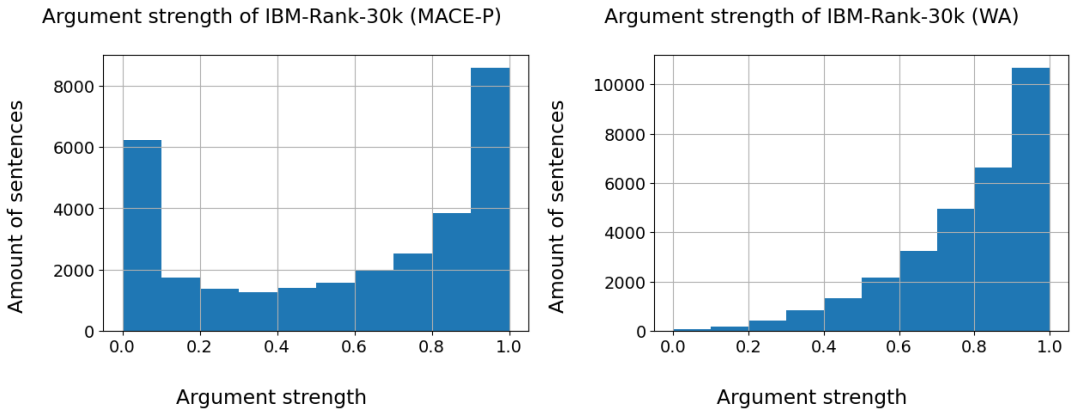
First, we consider IBM-Rank-30k, a fairly large point-wise annotated argument quality dataset. The published work is compromised of 30,497 arguments for 71 different topics. This is around 5 times as large as prior point-wise annotated datasets [25] and it seems to be the largest one published so far in this category.

The researchers collected their data using a combination of crowdsourcing and expert annotation. Most of the data was crowdsourced on the Figure Eight platforms, and a small portion of arguments (8.6%) was collected by

expert annotators who worked closely with the researchers. The annotators were presented with the following task: given a topic, they were asked to come up with one supporting and one contesting argument for it. Furthermore, the arguments were required to be high-quality, and written in original language.

The collected arguments then were annotated for quality, again by using crowd annotators on the Figure Eight platforms. For each argument, 10 annotators were presented with the binary question of whether they would recommend this argument *as is* to a friend preparing a speech supporting/contesting the topic, regardless of their personal opinion. Additionally, they had to mark if the argument was for or against the given topic. In order to ensure a high quality annotation process, they selected a group of 600 crowd annotators, which had high Annotator-reliability scores on previous tasks. Furthermore, before 1/5 of all arguments were labeled, the annotators were presented with hidden test questions to ensure that they were reading the sentences properly. If they failed more than 20% of the test questions, their judgments were ignored in the overall results.

Using the obtained binary annotations, they then derived a likelihood between 0 (bad quality) and 1 (good quality) for each argument. They utilized the previously proposed **MACE probability (MACE-P)** [12] and also proposed a new scoring function called **Weighted Average (WA)**. Both scoring functions try to incorporate annotator reliability and decrease the influence of non-reliable annotators on the final quality score. The main difference between these two functions becomes clear when looking at their score distribution in figure 3.1. For WA, values are skewed towards 1 with an almost linear decrease. MACE-P on the other hand assigns probabilities to both labels, leading to a U-shaped histogram.



(a) MACE-P (b) WA

Figure 3.1: The two scoring functions of the IBM-Rank-30k dataset.

3.2 IBM Evidences Sentences

Next, we consider the IBM Evidences Sentences dataset, a sentence-level evidence dataset. The dataset consists of a wide variety of topics extracted from Wikipedia articles, which were obtained using a novel end-to-end argument retrieval model. In total, there are 29,429 sentences from 221 different topics. While previous work often focused on document-level retrieval systems, this approach was inspired by Levy et al. (2017) [16], who were the first to propose a sentence-level claim detection framework.

The researchers put a heavy emphasis on two concepts, *Motion* and *Evidence*. A *Motion* is a high-level claim implying some clearly positive or negative stance towards a (debate's) topic, and, optionally, some policy or action that should be taken as a result. For example, for the motion *Guns should be banned,* the topic is *guns* and the proposed action is *ban*. In the context of a motion, they define *Evidence* as a single sentence, which clearly supports or contests the motion, yet is not merely a belief or claim. Rather, it provides an indication whether a belief or a claim is true. For the data acquisition process, one major hurdle in argument collection is the acquisition of a well-balanced dataset. In order to combat this, they proposed an end-to-end retrieval system, which aims to be able to retrieve a higher amount of true-positive examples. They start with a small set of manually collected labeled sentences, which is then used to train a simple classifier. Then, additional sentences are queried from a large corpus, using keywords such as the topic, action, or evidence-related

terms. These sentences are then passed to the model, which assigns a score to all samples. The top predictions of the classifier are then manually annotated by using crowdsourcing. This process can be executed iteratively to enrich the dataset with a sufficient amount of true-positive examples, and also "hard" negative examples. Once a large enough well-balanced dataset is obtained, the same process of prediction and annotation can be repeated using a more powerful classifier. The resulting classifier can then be used to generate scores for a dataset. A visualization of this process can be seen in figure 3.2.

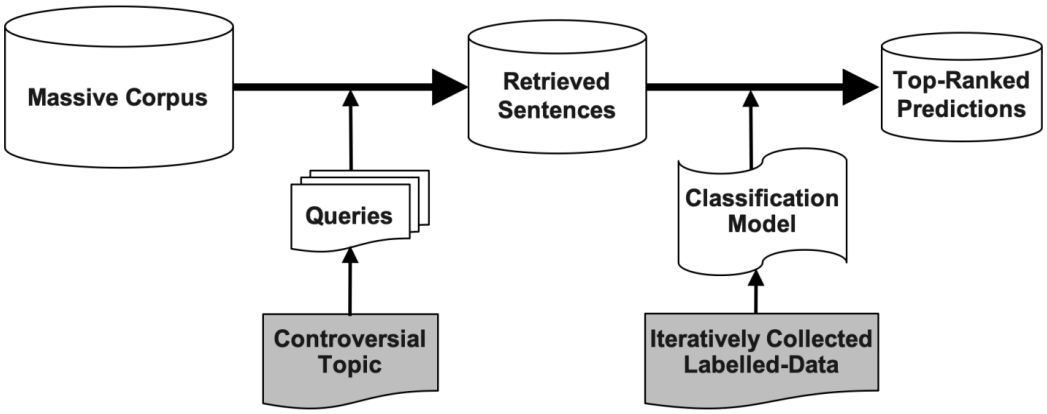


Figure 3.2: Overview of the end-to-end evidence retrieval system from the authors. [7]

In the case of the released dataset, each sentence and motion pair was annotated by 10 different annotators on the Figure Eight platform. The annotators were asked to assign a binary label, based on whether the sentence is an Evidence of a certain type. Annotators with a low agreement score were discarded, and the final label was determined using a majority vote. This process was run on two large corpora, a large one consisting of newspaper and journal articles, and a relatively small one consisting of Wikipedia articles. The latter was publicly released and is the one we are using for our research. A distribution of the final evidence score or acceptance rate can be seen in figure 3.3.

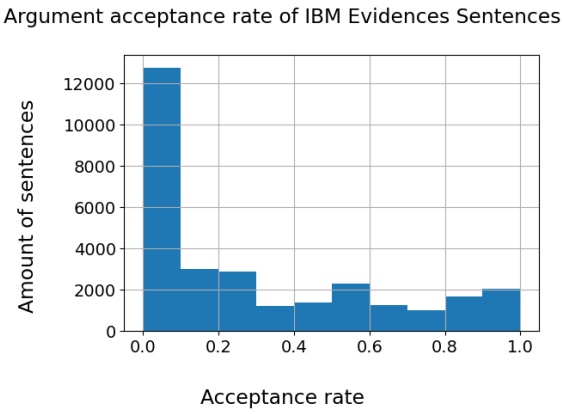
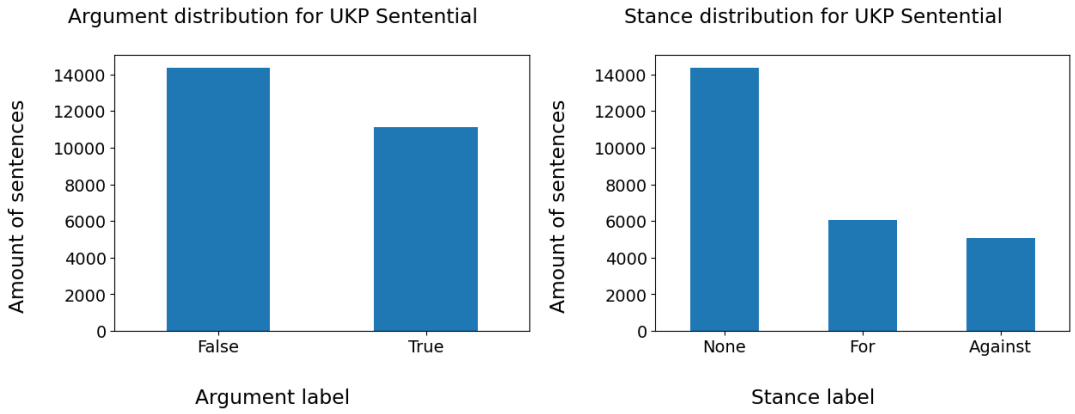


Figure 3.3: The evidence score distribution of the IBM Evidences Sentences dataset.

3.3 UKP Sentential

Finally, we consider UKP Sentential, an argument identification dataset released by the Ubiquitous Knowledge Processing Lab. The dataset is the smallest of the three, encompassing only 8 topics with 25,492 sentences extracted from text sources found on the Web. A main contribution in this work is a simple argument annotation scheme to extract arguments on a sentence-level from heterogeneous sources. Previous approaches often relied on consistent text types, which is usually unsuitable when extracting arguments from the Web. [4] However, their approach is quite restricted in a few regards. Instead of using high level motions like in 3.2, they use the lower level concept of a *topic*, which they define as a matter of controversy for which there is an obvious polarity to the possible outcomes. Additionally, they restrict themselves to topics that can be concisely and implicitly expressed through keywords. Furthermore, they define an *argument* as a span of text expressing evidence or reasoning that can be used to either support or oppose a given topic. In their work, they assume that an argument only consists of a single sentence. These restrictions reduce the variety of topics and complexity of arguments, however, in turn they are capable of handling a variety of text types. They applied this theory by randomly selecting eight controversial topics and collecting the top 50 Google results for each topic name. These results included a variety of text types such as news reports, editorials, blogs, debate forums, and encyclopedia articles. The collected documents were then preprocessed using standard tools like Apache Tika and Stanford CoreNLP tools. For the annotation, they simply presented the annotator a sentence alongside its topic with the option

to mark it as either a supporting argument, an opposing argument, or not an argument with respect to the topic. First, they used expert annotators to determine an annotator reliability baseline. After some experimentation, they were able to achieve reliability scores comparable to the expert annotations with 7 Amazon crowdsource annotators per argument. Using this configuration, they then proceeded to annotate the entire dataset. The resulting dataset is fairly balanced, since the relevant topics were generally controversial. The distribution of classes can be observed in figure 3.4.



(a) Argument label distribution

(b) Stance label distribution

Figure 3.4: The distribution of the UKP Sentential dataset.

Chapter 4

Methods

For this chapter, we provide details for how we structure our experiments. First, we describe the usage of Topic Information for our work. Next, we briefly highlight some issues when fine-tuning the BERT model, and describe how to mitigate them. Then, we explain the training configuration for all of our models, and outline our experiments with them. Last, we show the importance of the loss function in multi-task learning and describe our approach to multi-task models.

4.1 Topic Information

We already introduced the concept of a *Motion* in section 3.2 when discussing the IBM Evidences Sentences dataset. As a reminder, a *Motion* is a high-level claim implying some clearly positive or negative stance towards a (debate's) topic, and, optionally, some policy or action that should be taken as a result. When evaluating their dataset, they experimented with various input configurations and achieved the best results using the *Sentence+Motion* setting. Instead of only providing the model with a sentence, it was also given additional topic information in the form of a *Motion*. We are interested in further examining if topic information can be beneficial for other tasks as well. We therefore propose two input configuration for our models: 1) *Sentence* denoted as $\text{BERT}_{\mathbf{S}}$ and 2) *Sentence+Motion* denoted as $\text{BERT}_{\mathbf{S+M}}$. The motion is separated from the sentence using the BERT [SEP] token. Additionally, another perspective to consider for the topic information is the distribution of topics across different splits. We want to measure how well a model can handle topics it has not encountered during its training, or in other words, how well a model can generalize. We consider two scenarios here: 1) a cross-topic scenario, denoted as $\text{BERT}^{\mathbf{Cross}}$, where there is no intersection for the topics of the train, validation, and test splits and 2) an in-topic scenario, denoted as

$\text{BERT}^{\mathbf{In}}$, where we have roughly the same distribution of topics within each split. The former scenario of the two is generally seen as the more difficult one. In total, this results in four combinations of models we consider for each task:

- $\text{BERT}^{\mathbf{In}}_{\mathbf{S}}$: In-topic without motion
- $\text{BERT}^{\mathbf{In}}_{\mathbf{S+M}}$: In-topic with motion
- $\text{BERT}^{\mathbf{Cross}}_{\mathbf{S}}$: Cross-topic without motion
- $\text{BERT}^{\mathbf{Cross}}_{\mathbf{S+M}}$: Cross-topic with motion

4.2 Fine-Tuning BERT

When fine-tuning BERT on a downstream task, we find that BERT can yield a wide range of scores across different seeds. The BERT authors themselves noted this problem for smaller datasets on $\text{BERT}_{\text{LARGE}}$. [6] In recent years, this problem has been noted by other researchers as well, and there have been various attempts at improving the fine-tuning stability [13] [17]. In the original work which introduced BERT, the authors recommended the following hyperparameters:

- **Batch size:** 16, 32
- **Learning rate (Adam)**: 5e-5, 3e-5, 2e-5
- **Number of epochs:** 2, 3, 4

Since GPUs have become increasingly more performant over the last years, we instead opted for a higher batch size of 64 for decreased learning time. Initially, we experimented with a learning rate of 5e-5, but quickly saw that the model diverged in some of our runs. We find that a lower learning rate and the addition of warm-up steps, as recommended by Mosach et al. [17] results in much more stable training. For our experiments, we settle for a learning rate of 1e-5 and a warm-up period for the first 0.1 epochs. Additionally, we found that using epoch-based evaluations, BERT often over-fitted on the training data, which lead to non-optimal results. We argue that multiple evaluations per epoch allow us to obtain more optimal results, albeit at the cost of a longer training time. In our training configuration, we opt for evaluations every 0.1 epochs, resulting in 10 evaluations per epoch. As for our train/validation/test split, we utilize a fairly standard 70/10/20 split. Furthermore, we calculate the

99th percentile of the max length of all sentences inside the train and validation split, and truncate them to that length. This further decreases the required learning time, due to a reduced input dimension, without losing a significant amount of information. For all other relevant hyperparameters, we use the default settings provided by Huggingface[1], our library for implementation. Finally, to further reduce variance in training we use multiple seeds for our experiments, and calculate the mean and std for all of our results.

4.3 Single-Task Models

In the following section, we want to provide a brief overview of the architecture for all the single-task models we train, and the experiments we conduct with them. We will first consider an Argument Identification model (AId). Next, we take a look at an Argument Quality model (AQ). Last, we describe an Evidence Detection model (ED). For each setting our architecture is compromised of a simple $\text{BERT}_{\text{LARGE}}$ pre-trained model, followed by a dropout layer with 10% dropout probability feeding into a linear layer mapping from 768 dimensions to either 1 or 2 dimensions and a corresponding loss function depending on the problem type.

4.3.1 Argument Identification Model

Our Argument Identification model AId-BERT can assign any input a label indicating whether the given input is an argument. Since this is a binary classification task, our linear layer maps to 2 dimensions. A softmax function is then applied to the output, yielding probability values for each label. For the loss function, we use the Binary Cross-Entropy loss. For a given prediction probability/confidence p and a label $y \in \{0, 1\}$ we define it as

$$\text{BSE} = -(y \log(p) + (1 - y) \log(1 - p))$$

Additionally, for measuring our performance, we utilize a macro f1-score. For this task, we use the UKP Sentential dataset introduced in section 3.3. The dataset only provides split assignments for the in-topic scenario, so we also create a cross-topic version by manually assigning each topic to a split. This is slightly problematic, since the dataset only has 8 distinct topics. However, we find that by splitting the topics in a 5/1/2 distribution, we are able to achieve a roughly 70/10/20 distribution of the respective splits. We also map the "supportive" and "opposing" label included in this dataset to a simple "argumentative" label.

[1] https://huggingface.co/docs/transformers/master/en/main_classes/trainer

4.3.2 Argument Quality Model

Our Argument Quality model AQ-BERT can assign any input a quality score $q \in [0, 1]$, with 0 being the lowest quality score and 1 being the highest quality score. Our aforementioned model architecture uses a linear layer to map the 768 dimensions to 1 for this model. Since this is a regression problem, we can directly use this output and apply a simple Mean Squared Error (MSE) loss to measure how well our model approximates the target.

$$\text{MSE} = \sum_{i=1}^{D} |x_i - y_i|$$

Additionally, we use the pearson correlation to measure the linear correlation between the gold labels and our predictions. For this task, we use the IBM-Rank-30k dataset introduced in section 3.1. The dataset only provides split assignments for the cross-topic scenario, so we also create an in-topic version by randomly assigning each sentence with respect to the previously defined 70/10/20 set distribution. Our gold labels are represented by the Weighted Average metric contained in this dataset.

4.3.3 Evidence Detection Model

Our Evidence Detection model ED-BERT can assign any input an evidence score $e \in [0, 1]$, with 0 being the lowest score and 1 being the highest score. Evidence refers to a single sentence clearly supporting or contesting a motion, without being merely a belief or claim. Since this is again a regression problem, we use the same model architecture, the Mean Squared Error (MSE) as our loss function, and the pearson correlation as a performance measure. For this task, we use the IBM Evidences Sentences dataset introduced in section 3.2. The dataset does not come with any split assignments, so we create an in-topic version by again randomly assigning each sample according to our 70/10/20 distribution. Additionally, we create a cross-topic version with respect to both the topic-split assignments of the other datasets, and again the aforementioned distribution. Our gold labels are represented by the evidence score.

4.4 Transfer Zero-Shot Learning

We create all the models introduced in section 4.3 using the described model architectures. For each setting, we train four different versions with the topic configurations we mentioned in section 4.1. These models are then used to each conduct experiments on the datasets used for the other tasks. We are interested in finding out how well each model performs on the related tasks.

4.4.1 Correlation Measurement

In cases where two tasks share the same model architecture, the measuring of the performance for one task on a model trained on another is quite trivial. However, an issue arises if the model output has a different structure. We describe the two different scenarios we encounter for our experiments, and explain how we measure the correlation and performance there.

4.4.1.1 Classification Model on Continuous Label

Let us consider a binary classification model, which predicts a binary label $l \in \{0, 1\}$ for a given input. However, let us assume the data points we want to experiment with are annotated for a score $s \in [0, 1]$. Therefore, we either need to convert the model output to a single numeric value, so we can use the pearson correlation, or we need to find a threshold mapping the score to a label 0 or 1, so we can measure the f1-score. For the conversion to a numeric value, we train a multi-layer perceptron network with one hidden layer of size 100. For the input value to the network, we experiment with multiple different settings:

1. **Raw output:** the raw two-dimensional negative and positive model output
2. **Raw positive output:** the raw positive model output
3. **Raw negative output:** the raw negative model output
4. **Softmax positive output:** the softmax probability of the positive model output
5. **Softmax negative output:** the softmax probability of the negative model output

For settings 2 to 5 we also experiment with simply measuring the correlation on the raw unmapped value instead. The other setting we experiment with is the threshold mapping of the target score. Here, we compute all possible thresholds t mapping all data points with a continuous value x to the label 0 if $t < x$ and to the label 1 if $t >= x$. For each possible threshold, we then compute a macro f1-score and choose the threshold with the maximum f1-score.

4.4.1.2 Regression Model on Binary Label

Let us consider a regression model, which can predict a continuous score $s \in [0, 1]$ for a given input. Now, let us assume we want to measure a correlation on data points annotated with a binary label $l \in \{0, 1\}$. Unlike the scenario described in 4.4.1.1, we only have the option of applying a threshold mapping. So we again compute the optimal macro f1-score for all possible thresholds, as described in the previous section.

4.4.2 Experiment Outline

Using the aforementioned techniques, we investigate the correlation and performance of each models on the other tasks. We start with our AId model, a classification model. We evaluate it on the AQ and ED datasets, which are both labeled using continuous scores $s \in \{0, 1\}$. Therefore, we conduct our experiments on both of them using the methods described in section 4.4.1.1. Next, we look at the AQ model, a regression model. Here we have to use different methods for the other tasks. For evaluating the ED task on the AQ model, we can simply calculate the pearson correlation between the predictions and the gold labels. However, for the AId task, we need to again map the data. Since we have a regression model, the only viable option here is to convert our predictions to binary labels using the threshold mapping described in section 4.4.1.2 and calculate the macro f1-score. Lastly, the way we handle the ED model is very similar to the AQ model. Since we again are dealing with a regression model, we simply take the pearson correlation on the AQ task, and use threshold mapping on our predictions on the AId task to determine a macro f1-score.

4.5 Multi-Task Models

Now we take a look into combining all the aforementioned tasks into a multi-task model. We first introduce describe our general model architecture for our multi-task model and discuss the importance of choosing a proper loss function in a multi-task setting. Finally, we outline our training configuration for our multi-task models.

4.5.1 Model Architecture

While a single-task model consists of a simple BERT encoder and one task head, a multi-task model uses a simple BERT encoder with multiple task-heads. Multi-task learning is a concept derived from transfer learning. The

idea here is that different tasks can learn relevant information from each other. In theory, this way, we can help our model generalize better on each task. In a simple single-task setting, the choice of a loss function is often trivial. For regression problems one can simply use a standard Mean Squared Error (MSE) loss, while for classification problems Cross-Entropy (CE) loss is a popular choice. However, in a multi-task setting this is a more complex issue. Let us consider a multi-task model with two task heads, one AId task head with a binary CE loss, and one AQ task head with a MSE loss. Furthermore, let us suppose that in each training batch we have one AQ and one AId training sample. When gathering the gradients before calculating a backward propagation step, we need to combine the losses. A naive approach would be to take the sum of all loss functions l_t for each task $t \in T$.

$$l_{\text{naive}} = \sum_{t \in T} l_t$$

The issue with this approach is, that each loss generally has a different range of values for each task. For example, let's suppose the optimal MSE loss we can achieve for a single-task AQ model is $l_{\text{AQ}} = 0.01$, and the optimal CE loss for a single-task AId model is $l_{\text{AId}} = 0.09$. If we were to combine the losses with a naive sum, we would end up with a total loss of $l_{\text{naive}} = 0.10$. In a multi-task model, this loss function would be problematic. We essentially end up with unevenly weighted loss, which highly encourages the model towards the optimal solution for the AId head, while partly neglecting the AQ head. Ideally, we want our model to weigh both tasks equally, so they can both be learned in parallel. An improved approach would be to instead take a *weighted* sum with weights w_t for all tasks $t \in T$.

$$l_{\text{weighted}} = \sum_{t \in T} w_t l_t$$

For our example, let us use the weights $w_{\text{AQ}} = 9$ and $w_{\text{AId}} = 1$. This way we can achieve an effective loss of 0.9 for both of our loss functions, and a total loss of 1.8. However, determining these weights is an expensive and time-intensive process, so it's far from ideal. Instead, Kendall et al. suggest to use *Uncertainty* for weighting each class during training. [14] More specifically, they suggest the use of homoscedastic uncertainty to weight the combined loss function. The term "homoscedastic uncertainty" is a specific type of uncertainty. In Bayesian modelling, there are two main types of uncertainty one can model [14]

- *Epistemic* uncertainty is uncertainty in the model, which captures what our model does not know due to lack of training data. It can be explained away with increased training data.

- *Aleatoric* uncertainty captures our uncertainty with respect to information which our data cannot explain.

Aleatoric uncertainty can be explained away with the ability to observe all explanatory variables with increasing precision. Aleatoric uncertainty can again be divided into two subcategories.

- *Data-dependent* or *Heteroscedastic* uncertainty is aleatoric uncertainty, which depends on the input data and is predicted as a model output.
- *Task-dependent* or *Homoscedastic* uncertainty is aleatoric uncertainty, which is not dependent on the input data. It is not a model output, rather it is a quantity which stays constant for all input data and varies between different tasks. It can therefore be described as task-dependent uncertainty.

Using this idea, the researchers show that one can easily derive a multi-task loss function based on maximizing the Gaussian likelihood with homoscedastic uncertainty. For each task, they introduce a new parameter σ_t. This is the model's observation noise parameter, which captures how much noise there is in the outputs. They prove that both the CE and MSE loss can be written in the format required for the multi-task loss, which is defined as follows.

$$l_{\text{uncertainty}} = \sum_{t \in T} \frac{1}{2\sigma_t^2} l_t + \log \sigma_t$$

Effectively, each σ_t represents the weight for each loss function and is automatically learned in the learning process. In practice, we simply add a new parameter $s := \log \sigma_t^2$ to the model for each task, which is then used in the $l_{\text{uncertainty}}$ formula. [14] However, a small issue arises with this approach. Over time, the loss weights will continuously decrease as the model converges towards the training data. This is problematic, as we want our loss function to work for our validation dataset independently of the training data. If we simply were to use $l_{\text{uncertainty}}$ for validation, we would easily end up with an over-fitted model. Therefore, we opt to use l_{weighted} for our validation set instead, with the label weights W being based on the optimal loss values for each single-task model. Additionally, to avoid larger values of σ_t, we also apply the loss weight before in addition to the homoscedastic uncertainty.

$$l_{\text{uncertainty+weighted}} = \sum_{t \in T} \frac{w_t}{2\sigma_t^2} l_t + \log \sigma_t$$

4.5.2 Training Configuration

Using the architecture and loss function outlined in 4.5.1, we train all possible combinations of multi-task models using the task heads introduced in section 4.3. In total, this results in four different task head configurations:

- **AId-AQ-BERT**: Argument Identification and Argument Quality task heads
- **AId-ED-BERT**: Argument Identification and Evidence Detection task heads
- **AQ-ED-BERT**: Argument Quality and Evidence Detection task heads
- **AId-AQ-ED-BERT**: Argument Identification, Argument Quality and Evidence Detection task heads

Additionally, for each model, we again train the four different topic information settings outlined in section 4.1. Our experiments consist of simple zero-shot learning on the same tasks the model is trained on.

Chapter 5

Evaluation

In this chapter, we list and discuss all of our results for the experiments outlined in chapter 4.

5.1 Argument Identification Model

For our AId model, the results on the test split are as follows.

Model	f1-score
AId-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$79.99\% \pm 0.49\%$
AId-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$68.19\% \pm 6.57\%$
AId-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$80.51\% \pm 0.99\%$
AId-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$66.70\% \pm 3.76\%$
Average	$73.85\% \pm 7.48\%$

Table 5.1: Results of the AId model on the UKP Sentential test split.

We can observe that both in-topic settings perform substantially better than the cross-topic settings. This is somewhat expected, since the cross-topic scenario is more difficult and requires a more generalized model. It is likely our cross-topic AId model struggles with generalization across topics, as there are only five topics in our training split. Next, we examine the performance of this model on the other tasks using the methods described in section 4.4.1.

Model	f1-score	threshold
AId-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$55.16\% \pm 0.08\%$	0.73 ± 0.02
AId-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$54.72\% \pm 0.14\%$	0.79 ± 0.03
AId-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$58.45\% \pm 0.38\%$	0.46 ± 0.04
AId-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$61.03\% \pm 0.21\%$	0.65 ± 0.05
Average	$57.34\% \pm 2.58\%$	0.66 ± 0.13

Table 5.2: Correlation of the AId model with the threshold-mapped AQ task using the IBM-Rank-30k test split.

The performance of the AId model on the AQ task seems to be subpar. An average f1-score of 57.34% seems to only indicate a slight correlation between these tasks. The AId task does not seem to transfer very well to the AQ task. However, the models which were trained on the **S+M** setting perform generally better than the ones trained on the **S** setting. The inclusion of a motion in the input does seem to help the model with correctly classifying instances from an unknown topic.

Model	pearson
AId-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$18.24\% \pm 2.46\%$
AId-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$14.05\% \pm 1.16\%$
AId-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$31.88\% \pm 1.33\%$
AId-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$30.04\% \pm 1.35\%$
Average	$23.55\% \pm 7.76\%$

Table 5.3: Correlation of the AId model with the MLP-mapped AQ task using the IBM-Rank-30k test split.

We can observe the same findings using the MLP mapping instead of the threshold-mapping. In general, there is only a slight correlation, but the inclusion of a motion yields a significant increase in the correlation score. Table 5.3 only lists the MLP trained on the **Raw output** setting described in section

4.4.1.1, since we observe very similar pearson scores for all settings. Using the same methods, we also evaluate the ED task on our AId model.

Model	f1-score	threshold
AId-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$65.34\% \pm 0.65\%$	0.51 ± 0.05
AId-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$67.34\% \pm 2.06\%$	0.51 ± 0.05
AId-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$70.63\% \pm 0.29\%$	0.22 ± 0.06
AId-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$71.12\% \pm 0.88\%$	0.43 ± 0.06
Average	$68.61\% \pm 2.65\%$	0.42 ± 0.13

Table 5.4: Correlation of the AId model with the threshold-mapped ED task using the IBM Evidences Sentences test split.

We note a strong positive correlation for all of our models. In particular, for the cross-topic settings ED task, our AId model achieves similar performance to the AId test split. Additionally, the motion in the **S+M** setting again seems to yield an increase of a couple percent points for the f1-score. Again, we also consider the MLP-mapping alongside the threshold-mapping.

Model	pearson
AId-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$47.27\% \pm 1.17\%$
AId-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$48.11\% \pm 2.18\%$
AId-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$51.44\% \pm 0.79\%$
AId-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$55.28\% \pm 1.63\%$
Average	$50.53\% \pm 3.51\%$

Table 5.5: Correlation of the AId model with the MLP-mapped ED task using the IBM Evidences Sentences test split.

Here, we again find a strong positive correlation. We can also confirm our previous findings, that the setting with the motion achieves slightly higher correlation scores. As noted before, we again only consider the **Raw output** setting, as the other settings yielded comparable results. Summarizing, we find

a strong positive correlation with the ED task, and a slight positive correlation with the AQ task. In general, our in-topic setting seems to outperform the cross-topic setting, and including the motion alongside the argument in the input yields slight improvements in model performance.

5.2 Argument Quality Model

For our AQ model, the results on the test split are as follows.

Model	pearson
AQ-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$54.49\% \pm 0.12\%$
AQ-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$52.92\% \pm 0.27\%$
AQ-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$55.81\% \pm 0.35\%$
AQ-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$52.91\% \pm 0.86\%$
Average	$54.03\% \pm 1.31\%$

Table 5.6: Results of the AQ model on the IBM-Rank-30k test split.

We achieve a strong positive pearson score for all of our settings. However, the cross-topic setting is slightly less performant again than the in-topic setting. Additionally, there seem to be only marginal differences between the **S** and **S+M** settings. However, in general, our model achieves strong positive correlation on the AQ test split. Based on the results of the AId model, we expect there to be a slight positive correlation when evaluating the AId dataset on the AQ model.

Model	f1-score	threshold
AQ-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$70.75\% \pm 0.77\%$	0.69 ± 0.04
AQ-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$70.24\% \pm 1.44\%$	0.69 ± 0.04
AQ-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$71.25\% \pm 0.66\%$	0.69 ± 0.02
AQ-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$71.27\% \pm 0.74\%$	0.65 ± 0.04
Average	$70.88\% \pm 1.05\%$	0.68 ± 0.04

Table 5.7: Correlation of the threshold-mapped AQ model with the AId task using the UKP Sentential test split.

To our surprise, we find that our AQ model seems to generalize well enough to also solve the AId task to a sufficient degree. We achieve f1-scores of more than 70% for all settings. Although, unlike in the AId model, we only observe a slight increase in score in the **S+M** setting. Interestingly, we outperform the AId model in the cross-topic scenarios. A reason for this could be the poor generalization capabilities of the AId model across topics, since it is only trained on five topics in this setting.

Model	pearson
AQ-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$30.55\% \pm 0.91\%$
AQ-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$36.83\% \pm 3.42\%$
AQ-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$35.87\% \pm 3.19\%$
AQ-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$40.07\% \pm 2.65\%$
Average	$35.83\% \pm 4.37\%$

Table 5.8: Correlation of the AQ model with the ED task using the IBM Evidences Sentences test split.

For the ED task, we note a moderate positive correlation. We can again observe that all the models trained on the **S+M** seem to greatly benefit of the extra topic information provided by the motion. Interestingly, we can also observe, that the cross-topic setting outperforms the in-topic setting. This is the opposite of what we would expect, however, we have a possible explanation for

this effect. Assuming our AQ model generalizes well independently of topics, we would expect it to do well on topics it has previously not encountered. We saw this effect in the analysis of the AId task on the AQ model. However, another angle to consider is that arguments belonging to certain topics can be harder to classify. We test this theory by conducting our experiments on all splits instead of just the test split. This way, we remove the bias of topic distribution among the splits.

Model	pearson
AQ-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$31.76\% \pm 0.55\%$
AQ-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$32.65\% \pm 2.46\%$
AQ-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$36.28\% \pm 3.42\%$
AQ-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$35.42\% \pm 1.40\%$
Average	$34.03\% \pm 2.92\%$

Table 5.9: Correlation of the AQ model with the ED task using all splits in IBM Evidences Sentences.

We find that with the inclusion of all splits in evaluation, the differences between the in-topic and cross-topic settings become negligible. This supports our theory that some topics may be harder to classify than others. The reason behind this could be that the pre-trained model encountered certain topics not as often as others during its training process, making BERT less familiar with certain topics. To conclude our findings, we find that the AQ model has done well on both the AId and the ED task. This suggests that our AQ model has strong task-independent generalization capabilities.

5.3 Evidence Detection Model

For our ED model, the results on the test split are as follows.

Model	pearson
ED-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$80.03\% \pm 1.00\%$
ED-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$70.47\% \pm 0.26\%$
ED-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$83.93\% \pm 0.60\%$
ED-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$79.24\% \pm 0.30\%$
Average	$78.42\% \pm 4.96\%$

Table 5.10: Results of the ED model on the IBM Evidences Sentences test split.

We observe a very strong positive correlation across all settings. We can again recognize that the **S+M** setting yields significant performance improvements. Especially for the cross-topic setting, we find a 9% increase in pearson scores with the inclusion of the motion. Based on our previous results, we expect a strong positive correlation on the AId task, and a moderate positive correlation on the AQ task.

Model	f1-score	threshold
ED-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$72.08\% \pm 0.72\%$	0.20 ± 0.03
ED-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$75.01\% \pm 0.59\%$	0.24 ± 0.03
ED-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$72.01\% \pm 0.49\%$	0.24 ± 0.02
ED-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$75.16\% \pm 0.71\%$	0.22 ± 0.03
Average	$73.56\% \pm 1.65\%$	0.22 ± 0.03

Table 5.11: Correlation of the threshold-mapped ED model with the AId task using the UKP Sentential test split.

As expected, we find a strong positive correlation on the AId task. We notice that the results for the **S** and **S+M** setting are practically identical. When evaluating this task both on the AQ and AId model, we also noticed only a marginal increase in f1-score. This indicates that the inclusion of a motion might not play as much of a role in the AId task. We also observe that the

cross-topic setting seemingly performs better than the in-topic setting, so we again consider the scores on all splits.

Model	f1-score	threshold
ED-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$70.90\% \pm 0.71\%$	0.18 ± 0.01
ED-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$70.73\% \pm 0.25\%$	0.26 ± 0.02
ED-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$71.57\% \pm 0.45\%$	0.23 ± 0.02
ED-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$71.64\% \pm 0.59\%$	0.25 ± 0.02
Average	$71.21\% \pm 0.66\%$	0.23 ± 0.03

Table 5.12: Correlation of the threshold-mapped ED model with the AId task using all splits in UKP Sentential.

Here, the difference between the cross-topic and in-topic setting becomes marginal, indicating good generalization capabilities for our model. Additionally, in all cross-topic settings we outperform the AId model on its own task, further supporting that our AId model may suffer from poor generalization capabilities across topics. Lastly, we also evaluate the ED model on the AQ task.

Model	pearson
ED-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$24.25\% \pm 0.89\%$
ED-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$18.57\% \pm 2.42\%$
ED-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$29.76\% \pm 1.43\%$
ED-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$30.16\% \pm 1.10\%$
Average	$25.68\% \pm 4.98\%$

Table 5.13: Correlation of the ED model with the AQ task using the IBM-Rank-30k test split.

We find that the ED model yields similar scores to the AId model for solving the AQ task. As in a lot of the other settings, we notice a huge improvement with the inclusion of the motion in the **S+M** setting. However, there is only a slight positive correlation, instead of the moderate positive correlation we

observed for the ED task on the AQ model. This suggests that the AQ model generalizes better across tasks than the ED model. Summarizing, we can say that the ED model seems to translate very well to the AId task, but seems to be lacking when attempting to do the same for the AQ task.

5.4 Multi-Task Models

In the following, we present and analyze the results of all four multi-task models outlined in section 4.5.2. While both the AQ model and ED model performed well on their respective tasks, we saw in section 5.1 that the AId model exhibited poor performance in the cross-topic setting. We hope this is something we can improve in the multi-task setting. First, we consider the combination of the AId and AQ task.

Model	f1-score (AId)	pearson (AQ)
AId-AQ-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$79.04\% \pm 1.08\%$	$53.70\% \pm 0.39\%$
AId-AQ-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$72.36\% \pm 2.40\%$	$52.00\% \pm 0.72\%$
AId-AQ-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$81.12\% \pm 0.63\%$	$55.06\% \pm 0.41\%$
AId-AQ-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$80.07\% \pm 1.16\%$	$52.92\% \pm 0.60\%$
Average	$78.15\% \pm 3.73\%$	$53.42\% \pm 1.25\%$

Table 5.14: Results of the AId-AQ model on the respective test splits for each task.

While the results for the AQ task are very comparable to the ones in the single-task model, we notice large improvements for the AId task in the cross-topic settings. We believe that additionally learning the AQ task, helped create a more generalized model. We now investigate if the same improvements for the AId task can be found in the combined AId and ED model.

Model	f1-score (AId)	pearson (ED)
AId-ED-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$79.78\% \pm 0.30\%$	$79.38\% \pm 0.72\%$
AId-ED-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$69.90\% \pm 4.30\%$	$68.58\% \pm 1.09\%$
AId-ED-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$81.18\% \pm 0.43\%$	$82.46\% \pm 0.75\%$
AId-ED-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$74.86\% \pm 3.38\%$	$79.41\% \pm 0.36\%$
Average	$76.43\% \pm 5.22\%$	$77.46\% \pm 5.33\%$

Table 5.15: Results of the AId-ED model on the respective test splits for each task.

The performance on the ED head is again mostly comparable to the ones in the single-task setting, albeit marginally worse by one to two percentage points. For the AId head, we notice slight improvements to the scores in the cross-topic setting, but the improvement is not as drastic as observed for the combined AId and ED model. This is despite the fact that the ED dataset contains 221 topics, while the AQ dataset only contains 71. We think this provides further evidence, that training on the AQ task seems to provide a more generalized model. For both the ED and AQ task, we notice a slight decrease in scores when trained in conjunction with the AId task. We are interested in finding out if the same can be observed when training a model for the AQ and ED task.

Model	pearson (AQ)	pearson (ED)
AQ-ED-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$53.31\% \pm 1.61\%$	$79.77\% \pm 0.67\%$
AQ-ED-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$52.79\% \pm 0.43\%$	$70.58\% \pm 0.37\%$
AQ-ED-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$55.60\% \pm 0.37\%$	$83.88\% \pm 0.56\%$
AQ-ED-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$52.56\% \pm 0.49\%$	$78.82\% \pm 0.45\%$
Average	$53.56\% \pm 1.50\%$	$78.26\% \pm 4.85\%$

Table 5.16: Results of the AQ-ED model on the respective test splits for each task.

While the model was able to learn both the ED and AQ task, we notice slight

score decreases when comparing the results to the ones of the single-task models. We can infer from this, that for both of these tasks, the single-task model can already provide a good solution, so the individual tasks may not benefit much from the information of unrelated tasks. Lastly, we also want to evaluate the model with all three tasks.

Model	f1-score (AId)	pearson (AQ)	pearson (ED)
AId-AQ-ED-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$80.08\% \pm 0.37\%$	$54.10\% \pm 0.15\%$	$80.48\% \pm 0.31\%$
AId-AQ-ED-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$76.12\% \pm 1.97\%$	$51.99\% \pm 0.59\%$	$70.29\% \pm 0.90\%$
AId-AQ-ED-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$80.94\% \pm 0.80\%$	$55.28\% \pm 0.69\%$	$83.28\% \pm 0.87\%$
AId-AQ-ED-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$78.91\% \pm 3.17\%$	$53.82\% \pm 0.87\%$	$79.12\% \pm 0.20\%$
Average	$79.01\% \pm 2.64\%$	$78.29\% \pm 4.90\%$	$53.80\% \pm 1.34\%$

Table 5.17: Results of the AId-AQ-ED model on the respective test splits for each task.

We again observe that both the AQ and ED task are marginally worse than they are for their single-task setting. However, we can notice even further improvements in the AId tasks. We now find an average of 79.01% f1-score for this task, with every individual score being higher than in the single-task AId model. To summarize, we find major improvements for the AId task in multi-task learning. However, both the AQ and ED tasks can already be sufficiently solved by their respective single-task model, and unfortunately do not see any improvements in the multi-task setting.

Chapter 6

Conclusion and Future Work

We conclude with providing a summary of our findings and an interpretation of the results. Finally, we end with touching upon some potential future research directions.

6.1 Conclusion

We demonstrated that we can train models for the AId, AQ, and ED task using the datasets and methods outlined in chapters 3 and 4. We showed in chapter 5 that we can reproduce the results obtained by previous approaches using the same datasets. Furthermore, we demonstrated that both the model of the ED and AId task are capable of solving each other's task. We were able to show the same for the AQ model, which seems to generalize well enough to also solve the ED and AId task to a certain degree. However, we found that both the ED and AId model struggle with solving the AQ task, indicating a lack of task-independent generalization. We highlighted the importance of topic information with our different settings. We noticed that including a motion in the input alongside the argument, can significantly improve a model's performance. Even in cases without improvements, the motion also did not negatively impact the scores. Additionally, we highlighted the major score differences between the in-topic and cross-topic datasets, further proofing the importance of topic information. We also showed how the use of multi-task learning can significantly improve the results of certain tasks. While a single-task model can already sufficiently solve both the AQ and ED task, we noted major improvements in the AId task when training it alongside another task.

6.2 Future Work

We touch upon many subjects in this thesis, many of which can be further investigated. We want to give three possible future research directions, which we think are interesting to explore. First, we only explored transfer zero-shot learning for our single-task models. However, the same idea could be explored with multi-task models as well. For example, it could be quite interesting, to see how the AId task would perform on the AQ-ED-BERT model. Another area we think is interesting to explore, is the motion in the model input. For some models, we observed only marginal improvements when including the motion in our input to the model. However, for others, we observed quite significant improvements in terms of f1-score or pearson correlation. We believe it could be interesting to research, which tasks benefit more from motions than others, and why that is the case. Furthermore, we could also explore training a model on the **S+M** setting, and later evaluate it on the **S** setting, or vice versa. Last, our ideas could be extended to other tasks beyond the AQ, ED and AId tasks.

Bibliography

[1] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

[2] Elena Cabrio and Serena Villata. Five years of argument mining: a data-driven analysis. In *IJCAI*, volume 18, pages 5427–5433, 2018.

[3] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML '08*, 2008.

[4] Johannes Daxenberger, Steffen Eger, Ivan Habernal, Christian Stab, and Iryna Gurevych. What is the essence of a claim? cross-domain claim identification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2055–2066, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[5] Li Deng, Geoffrey E. Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: an overview. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8599–8603, 2013.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[7] Liat Ein-Dor, Eyal Shnarch, Lena Dankin, Alon Halfon, Benjamin Sznajder, Ariel Gera, Carlos Alzate, Martin Gleize, Leshem Choshen, Yufang Hou, Yonatan Bilu, Ranit Aharonov, and Noam Slonim. Corpus wide argument mining – a working solution, 2019.

[8] Ross B. Girshick. Fast r-cnn. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[9] Shai Gretz, Roni Friedman, Edo Cohen-Karlik, Assaf Toledo, Dan Lahav, Ranit Aharonov, and Noam Slonim. A large-scale dataset for argument quality ranking: Construction and analysis, 2019.

[10] Kathrin Grosse, Maria P Gonzalez, Carlos I Chesnevar, and Ana G Maguitman. Integrating argumentation and sentiment analysis for mining opinions from twitter. *AI Communications*, 28(3):387–401, 2015.

[11] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[12] Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[13] Hang Hua, Xingjian Li, Dejing Dou, Cheng-Zhong Xu, and Jiebo Luo. Noise stability regularization for improving bert fine-tuning. *arXiv preprint arXiv:2107.04835*, 2021.

[14] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics, 2018.

[15] John Lawrence and Chris Reed. Argument mining: A survey. *Computational Linguistics*, 45(4):765–818, December 2019.

[16] Ran Levy, Shai Gretz, Benjamin Sznajder, Shay Hummel, Ranit Aharonov, and Noam Slonim. Unsupervised corpus–wide claim detection. In *Proceedings of the 4th Workshop on Argument Mining*, pages 79–84, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[17] Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines, 2021.

[18] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *ArXiv*, abs/1706.05098, 2017.

[19] Sebastian Ruder, Matthew E Peters, Swabha Swayamdipta, and Thomas Wolf. Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 15–18, 2019.

[20] Claudia Schulz, Steffen Eger, Johannes Daxenberger, Tobias Kahse, and Iryna Gurevych. Multi-task learning for argumentation mining in low-resource settings. In *NAACL*, 2018.

[21] Mike Schuster and Kaisuke Nakajima. Japanese and korean voice search. *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152, 2012.

[22] Andrés Segura-Tinoco and Iván Cantador. On the extraction and use of arguments in recommender systems: A case study in the e-participation domain (long paper). In *KaRS/ComplexRec@RecSys*, 2021.

[23] Christian Stab, Tristan Miller, and Iryna Gurevych. Cross-topic argument mining from heterogeneous sources using attention-based neural networks, 2018.

[24] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

[25] Assaf Toledo, Shai Gretz, Edo Cohen-Karlik, Roni Friedman, Elad Venezian, Dan Lahav, Michal Jacovi, Ranit Aharonov, and Noam Slonim. Automatic argument quality assessment – new datasets and methods, 2019.

[26] Nhat Tran and Diane Litman. Multi-task learning in argument mining for persuasive online discussions. In *Proceedings of the 8th Workshop on Argument Mining*, pages 148–153, 2021.

[27] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

[28] Yonghui Wu, Mike Schuster, Z. Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason R. Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv*, abs/1609.08144, 2016.

[29] Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and

the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41:2251–2265, 2019.

[30] Xing Xu, Fumin Shen, Yang Yang, D. Zhang, Heng Tao Shen, and Jingkuan Song. Matrix tri-factorization with manifold regularizations for zero-shot learning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2007–2016, 2017.

[31] Xiaodong Yu and Yiannis Aloimonos. Attribute-based transfer learning for object categorization with zero/one training example. In *ECCV*, 2010.

Appendix A

Combined result table

Model	f1-score (AId)	pearson (AQ)	pearson (ED)
AId-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$79.99\% \pm 0.49\%$	$18.24\% \pm 2.46\%$	$47.27\% \pm 1.17\%$
AId-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$68.19\% \pm 6.57\%$	$14.05\% \pm 1.16\%$	$48.11\% \pm 2.18\%$
AId-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$80.51\% \pm 0.99\%$	$31.88\% \pm 1.33\%$	$51.44\% \pm 0.79\%$
AId-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$66.70\% \pm 3.76\%$	$30.04\% \pm 1.35\%$	$55.28\% \pm 1.63\%$
AQ-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$70.75\% \pm 0.77\%$	$54.49\% \pm 0.12\%$	$30.55\% \pm 0.91\%$
AQ-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$70.24\% \pm 1.44\%$	$52.92\% \pm 0.27\%$	$36.83\% \pm 3.42\%$
AQ-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$71.25\% \pm 0.66\%$	$55.81\% \pm 0.35\%$	$35.87\% \pm 3.19\%$
AQ-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$71.27\% \pm 0.74\%$	$52.91\% \pm 0.86\%$	$40.07\% \pm 2.65\%$
ED-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$72.08\% \pm 0.72\%$	$24.25\% \pm 0.89\%$	$80.03\% \pm 1.00\%$
ED-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$75.01\% \pm 0.59\%$	$18.57\% \pm 2.42\%$	$70.47\% \pm 0.26\%$
ED-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$72.01\% \pm 0.49\%$	$29.76\% \pm 1.43\%$	$83.93\% \pm 0.60\%$
ED-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$75.16\% \pm 0.71\%$	$30.16\% \pm 1.10\%$	$79.24\% \pm 0.30\%$

Table A.1: Results of all single-task models on the respective test splits for each task.

Model	f1-score (AId)	pearson (AQ)	pearson (ED)
AId-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$79.99\% \pm 0.49\%$	$18.24\% \pm 2.46\%$	$47.27\% \pm 1.17\%$
AId-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$68.19\% \pm 6.57\%$	$14.05\% \pm 1.16\%$	$48.11\% \pm 2.18\%$
AId-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$80.51\% \pm 0.99\%$	$31.88\% \pm 1.33\%$	$51.44\% \pm 0.79\%$
AId-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$66.70\% \pm 3.76\%$	$30.04\% \pm 1.35\%$	$55.28\% \pm 1.63\%$

Table A.2: Results of all single-task models on the respective test splits for each task.

Model	f1-score (AId)	pearson (AQ)	pearson (ED)
AQ-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$70.75\% \pm 0.77\%$	$54.49\% \pm 0.12\%$	$30.55\% \pm 0.91\%$
AQ-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$70.24\% \pm 1.44\%$	$52.92\% \pm 0.27\%$	$36.83\% \pm 3.42\%$
AQ-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$71.25\% \pm 0.66\%$	$55.81\% \pm 0.35\%$	$35.87\% \pm 3.19\%$
AQ-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$71.27\% \pm 0.74\%$	$52.91\% \pm 0.86\%$	$40.07\% \pm 2.65\%$

Table A.3: Results of all single-task models on the respective test splits for each task.

Model	f1-score (AId)	pearson (AQ)	pearson (ED)
ED-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$72.08\% \pm 0.72\%$	$24.25\% \pm 0.89\%$	$80.03\% \pm 1.00\%$
ED-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$75.01\% \pm 0.59\%$	$18.57\% \pm 2.42\%$	$70.47\% \pm 0.26\%$
ED-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$72.01\% \pm 0.49\%$	$29.76\% \pm 1.43\%$	$83.93\% \pm 0.60\%$
ED-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$75.16\% \pm 0.71\%$	$30.16\% \pm 1.10\%$	$79.24\% \pm 0.30\%$

Table A.4: Results of all single-task models on the respective test splits for each task.

Model	f1-score (AId)	pearson (AQ)	pearson (ED)
AId-AQ-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$79.04\% \pm 1.08\%$	$53.70\% \pm 0.39\%$	-
AId-AQ-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$72.36\% \pm 2.40\%$	$52.00\% \pm 0.72\%$	-
AId-AQ-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$81.12\% \pm 0.63\%$	$55.06\% \pm 0.41\%$	-
AId-AQ-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$80.07\% \pm 1.16\%$	$52.92\% \pm 0.60\%$	-
AId-ED-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$79.78\% \pm 0.30\%$	-	$79.38\% \pm 0.72\%$
AId-ED-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$69.90\% \pm 4.30\%$	-	$68.58\% \pm 1.09\%$
AId-ED-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$81.18\% \pm 0.43\%$	-	$82.46\% \pm 0.75\%$
AId-ED-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$74.86\% \pm 3.38\%$	-	$79.41\% \pm 0.36\%$
AQ-ED-BERT$^{\mathbf{In}}_{\mathbf{S}}$	-	$53.31\% \pm 1.61\%$	$79.77\% \pm 0.67\%$
AQ-ED-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	-	$52.79\% \pm 0.43\%$	$70.58\% \pm 0.37\%$
AQ-ED-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	-	$55.60\% \pm 0.37\%$	$83.88\% \pm 0.56\%$
AQ-ED-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	-	$52.56\% \pm 0.49\%$	$78.82\% \pm 0.45\%$
AId-AQ-ED-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$80.08\% \pm 0.37\%$	$54.10\% \pm 0.15\%$	$80.48\% \pm 0.31\%$
AId-AQ-ED-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$76.12\% \pm 1.97\%$	$51.99\% \pm 0.59\%$	$70.29\% \pm 0.90\%$
AId-AQ-ED-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$80.94\% \pm 0.80\%$	$55.28\% \pm 0.69\%$	$83.28\% \pm 0.87\%$
AId-AQ-ED-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$78.91\% \pm 3.17\%$	$53.82\% \pm 0.87\%$	$79.12\% \pm 0.20\%$

Table A.5: Results of all multi-task models on the respective test splits for each task.

Model	f1-score (AId)	pearson (AQ)	pearson (ED)
AId-AQ-ED-BERT$^{\mathbf{In}}_{\mathbf{S}}$	$80.08\% \pm 0.37\%$	$54.10\% \pm 0.15\%$	$80.48\% \pm 0.31\%$
AId-AQ-ED-BERT$^{\mathbf{Cross}}_{\mathbf{S}}$	$76.12\% \pm 1.97\%$	$51.99\% \pm 0.59\%$	$70.29\% \pm 0.90\%$
AId-AQ-ED-BERT$^{\mathbf{In}}_{\mathbf{S+M}}$	$80.94\% \pm 0.80\%$	$55.28\% \pm 0.69\%$	$83.28\% \pm 0.87\%$
AId-AQ-ED-BERT$^{\mathbf{Cross}}_{\mathbf{S+M}}$	$78.91\% \pm 3.17\%$	$53.82\% \pm 0.87\%$	$79.12\% \pm 0.20\%$

Table A.6: Results of all multi-task models on the respective test splits for each task.