

Transformer-Encoder and Decoder Models for Questions on Math

Anja Reusch¹, Maik Thiele² and Wolfgang Lehner¹

¹Database Systems Group, Technische Universität Dresden, Germany

²Hochschule für Wirtschaft und Technik Dresden, Germany

Abstract

This work summarizes our submission to ARQMath-3. We pre-trained Transformer-Encoder-based Language Models for the task of mathematical answer retrieval and employed a Transformer-Decoder Model for the generation of answers given a question from a mathematical domain. In comparison to our submission to ARQmath-2, we could improve the performance of our models regarding all three metrics nDCG, mAP and p@10 by refined pre-training and enlarged fine-tuning data. In addition, we improved our p@10 results even further by additionally fine-tuning on annotated test data from ARQMath-2. In summary, our findings confirm that Transformer-based models benefit from domain adaptive pre-training in the mathematical domain.

Keywords

Mathematical Language Processing, Information Retrieval, Transformer-based Models

1. Introduction

With a rising number of scientific publications, retrieval of information from documents containing mathematical notation has recently received more attention. The task of Mathematical Information Retrieval (MIR) deals with finding relevant documents for a query, where both document and query may include mathematical notation such as \LaTeX expressions beside natural language. As for many text-based tasks, Transformer-based models have demonstrated great potential for MIR. They could even be applied as a stand-alone model when adapted to the domain, since models like BERT [1] or ALBERT [2] were originally pre-trained on documents that did not contain mathematical notation. Recent research has therefore focused on adapting these models to the domain of mathematics by additional pre-training on the Mathematics StackExchange (MathSE).

However, several base models such as BERT and ALBERT were further pre-trained and evaluated using different methods or data sets [3, 4, 5]. Hence, a fair comparison which base model is best suited for MIR is not possible. In order to evaluate their impact under the same conditions, we start our submission by pre-training and fine-tuning three Transformer-Encoder models, namely ALBERT, BERT and RoBERTa, on MIR.

CLEF 2022: Conference and Labs of the Evaluation Forum, September 5–8, 2022, Bologna, Italy


✉ anja.reusch@tu-dresden.de (A. Reusch); maik.thiele@htw-dresden.de (M. Thiele);

wolfgang.lehner@tu-dresden.de (W. Lehner)

🌐 <https://wwwdb.inf.tu-dresden.de> (W. Lehner)

🆔 0000-0002-2537-9841 (A. Reusch); 0000-0002-1665-977X (M. Thiele); 0000-0001-8107-2775 (W. Lehner)

© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

In the core of this work, we refine our ALBERT-based approach from ARQMath-2. Here, the model was pre-trained on MathSE serving as in-domain data and fine-tuned using a classification task with the objective to predict whether a given post answers the question. The resulting classification probability assigned by the model was used to rank the answers.

The success of models like BERT is attributed to pre-training on large diverse corpora. To evaluate the influence of the corpus, we experiment with pre-training using the AMPS corpus [6] which consists of 23GByte of question and answer pairs. Compared to 2021, we also enlarged our fine-tuning corpus by 152%. Furthermore, we also rely on the annotated test data of last year which we are leveraging for more informed training data for MIR. We will also study the impact of this new training data on our models.

Finally, ARQMath-3 includes the new task of generating answers instead of retrieving them from the corpus. Our team fine-tuned a GPT-2 model [7] on the AMPS as well as on the provided MathSE corpora to generate solutions for the questions from the retrieval task.

In summary, our submission to the ARQMath Lab 2022 Task 1 Answer Retrieval and Task 3 Answer Generation focuses on three areas:

- A comparison of three pre-trained Transformer-Encoder models: BERT, ALBERT, RoBERTa,
- The impact of pre-training and fine-tuning data on MIR,
- The application of Transformer-Decoder Models to Mathematical Answer Generation.

Our evaluation shows that all our submitted models could outperform the best models of ARQMath-2. Our ALBERT model trained on three different versions of the MathSE data and the enlarged fine-tuning data demonstrated the best performance. We could improve $p@10$ even more by fine-tuning using annotated data from the run of 2021. Training models based on RoBERTa also shows to result in a promising approach, even though longer computing time is needed. All our models are made publicly available as part of the Huggingface Model Hub¹.

The remainder of this submission document is structured as follows: Section 2 will review related work in the field of MIR and related generation tasks. We will introduce the tasks of the lab in Section 3 and the models in Section 4. Section 5 describes our model setup and the results of Task 1, while Section 6 will summarize our efforts for Task 3. The final section concludes this work.

2. Related Work

Deep learning models based on Encoders or Decoders of the Transformer architecture [8] have been widely introduced in several Natural Language Processing and Information Retrieval Tasks in recent years. Encoder-Models like BERT [1], ALBERT [2], and RoBERTa [9] have been applied to various domains including scientific literature [10], medical documents [11, 12] or source code [13, 14]. Decoder-Models are typically used to generate text, where the most prominent examples being GPT-1, GPT-2, and GPT-3 [15, 7, 16].

Transformer-Encoder-based models for mathematical domains have also been studied with one example being MathBERT [17]. Here, mathematical formulas in form of operator trees

¹<https://huggingface.co/AnReu/albert-for-arqmath-3>

are used as an input for pre-training. During the ARQMath Lab in 2020 and 2021, five teams submitted systems based on BERT, RoBERTa, and SentenceBERT [18, 19, 20, 21, 22, 23] where the models were used without domain adaption for downstream tasks. Only [4] pre-trained their submissions on mathematical documents. [3] fine-tuned a BERT model for notation prediction tasks based on scientific documents by enlarging the vocabulary of BERT with additional \LaTeX tokens. [5] followed a similar procedure for mathematical documents.

Generative Models have also found their way into the mathematical domain with GPT- f , a Transformer-based proof-solver [24]. [6] introduced two new data sets, one for measuring the performance of generative mathematical language models and one for pre-training. Along with it, benchmarks based on GPT-2 and GPT-3 were published. All of these only used an exercise-level data set and not a community-data set like the MathSE data in the task at hand.

3. ARQMath 2022 Lab

The overall goal of ARQMath Lab 2022 (ARQMath-3) [25] is to accelerate the research in mathematical Information Retrieval. The lab consists of three tasks offering three different scenarios. Task 1 of the lab involves mathematical answer retrieval for a question asked on the Mathematics StackExchange², which is a platform for users to post questions related to mathematical topics to be answered by the community. The goal of this task is the retrieval of an answer post from 2010 - 2018 to questions that were posted in 2019. The evaluation data of ARQMath-1 contain 99, while ARQMath-2 and 3 provided 100 query topics each, which are question posts including title, text and tags. In the 2020 test set, 77 queries were evaluated for Task 1, while its evaluation in 2021 included 71 queries. ARQMath-3 evaluated 78 queries. The optimal answers retrieved by the participants are expected to answer the complete question on their own. The relevance of the question-answer pairs was assessed by reviewers during the evaluation process. This relevance assessment was performed by pooling after the teams submitted their results.

For each topic the participating teams submitted a ranked list of 1,000 documents retrieved by their systems, which were scored by Normalized Discounted Cumulative Gain, but with unjudged documents removed before assessment (nDCG'). The graded relevance scale used for scoring ranged from 0 (not relevant) to 3 (highly relevant). Two additional measures, mAP' and p@10, were also reported using binarized relevance judgments (0 and 1: not relevant, 2 and 3: relevant).

Task 2 of the ARQMath is built on top of the same data as Task 1, but with a different goal in mind: Participants are expected to retrieve relevant formulas given a query formula in the context of its post. This task is related to the formula browsing task of NTCIR-12 [26].

ARQMath-3's new Task 3 presents an open-domain question/answering scenario instead of finding the most relevant answers for a given question. For each of the 100 topics of Task 1 in the 2022 test set the participants are asked to extract or generate a single answer. These answers contributed to the pool of answers which were judged for Task 1. Any knowledge source - except for the MathSE data from 2019 to today - was allowed as training data. The evaluation is carried out by evaluating the average relevance (AR) of the answers and the

²<https://math.stackexchange.com>

Precision at 1 (p@1) for each topic.

Apart from the task definitions and the evaluation data, ARQMath provides data from the Mathematics StackExchange including question and answer posts from 2010 to 2018. In total, the collection contains 1M questions and 1.4M answers. Furthermore, users may use mathematical formulas to clarify their posts. These formulas written in \LaTeX notation were extracted and parsed into Symbol Layout Trees and Operator Trees. Apart from this corpus of posts and formulas that are available for training and evaluating models, the organizers of ARQMath also released a test set of queries.

4. Transformer-based Models

The Transformer Architecture as introduced by [8] consists of Encoder and Decoder layers. Encoder models in Natural Language Processing typically apply several of these encoder layers on top of each other resulting in a model that reads a sequence of tokens and outputs contextualized embeddings for each token as well as for the entire input. These embeddings can then be further processed for classification tasks among others. In contrast, decoder models are designed to generate the next output token given a sequence of previous tokens (context tokens). Training both types of models usually includes two phases: A pre-training phase and a fine-tuning phase. While pre-training consists of training the model on relatively simple self-supervised tasks on a large amount of data, fine-tuning does not necessarily need large annotated data.

In the following sections we will describe the pre-training of both model types, differences in the concrete model instances we applied and our fine-tuning for the respective task.

4.1. Encoder Models

Encoder models are trained to capture the meaning of natural language by self-supervised pre-training tasks. The most important and widely applied task is the Masked Language Model. The model is presented with the embeddings E_i for each token i from the input sentence:

$$CU_1U_2\cdots U_N = \text{BERT}(E_{CLS}E_1E_2\cdots E_N)$$

, where E_{CLS} and C are the input and output embeddings of the $\langle CLS \rangle$ token. A classifier is then applied to predict the original word given the input:

$$P(w_j|S) = \text{softmax}(U_i \cdot W_{MLM} + b_{MLM})_j,$$

where w_j is the j -th word from the vocabulary. This determines the probability that the i -th input word was w_j given the input sentence S . The weight matrix W_{MLM} and its bias b_{MLM} are only used for this pre-training task and are not reused afterwards.

RoBERTa uses only the MLM task, while BERT and ALBERT also employ a second pre-training task on the same input data, which is a sequence classification task applied on top of the contextualized embeddings of the $\langle CLS \rangle$ token:

$$P(\text{label} = i|S) = \text{softmax}(C \cdot W_{SOP} + b_{SOP})_i,$$

where the matrix W_{SOP} and the bias b_{SOP} are only used for pre-training and are not re-used otherwise later. In practice, this task is used to learn coherence between two input sequences given to the model. In this work, we pre-train using the Sentence Order Prediction (SOP), where $\text{label} = 1$ denotes that the two input sequences are in correct order, while $\text{label} = 0$ denotes that they were swapped.

Fine-Tuning

In order to predict whether an answer $A = A_1A_2 \cdots A_M$ is relevant to a question $Q = Q_1Q_2 \cdots Q_N$ a classifier is trained on top of the pre-trained Transformer-Encoder model. The input string $\langle CLS \rangle Q_1Q_2 \cdots Q_N \langle SEP \rangle A_1A_2 \cdots A_M$, with $\langle CLS \rangle$ being the classification token and $\langle SEP \rangle$ the separation token, is presented to the model:

$$CU_1U_2 \cdots U_N = \text{LM}(E_{CLS}E_1E_2 \cdots E_{N+M}),$$

where E_i and E_{CLS} are the input embeddings for each input token and the $\langle CLS \rangle$ token, respectively, calculated as explained in the previous section. After the forward pass through the model, the output vector of the $\langle CLS \rangle$ token C is given into a classification layer:

$$P(\text{label} = i|Q, A) = \text{softmax}(C \cdot W_{MIR} + b_{MIR})_i,$$

where the label 1 stands for a matching or correct answer for the query and label 0 otherwise. During evaluation, the resulting probability of the classification layer for label 1 is the assigned a similarity score s for the answer A to a question Q and is then used to rank all answers in the corpus: $s(Q, A) = p(\text{label} = 1|Q, A)$.

4.2. Decoder Models

Decoder models are trained on the casual language modeling objective, i.e., given some input tokens, generate the most probable next token. In other words, the objective is to maximize the probability over the corpus consisting of a sequence of n tokens $C = \{t_0, t_1, \dots, t_n\}$:

$$P(C) = \prod_{i=0}^n P(t_i|t_{i-k}, \dots, t_{i-1}),$$

where the conditional probability $P(s_t|t_{i-k}, \dots, t_{i-1})$, ranging over a context window of size k , is estimated by the Decoder model. The input is embedded and given to the Transformer Decoder layers resulting in the last layer's output h_{last} , which is used to calculate the probability:

$$P(u) = \text{softmax}(h_{\text{last}} \cdot W_e^T),$$

with W_e^T being the embedding matrix.

Decoder Models such as GPT or GPT-2 are also fine-tuned in a supervised fashion to predict

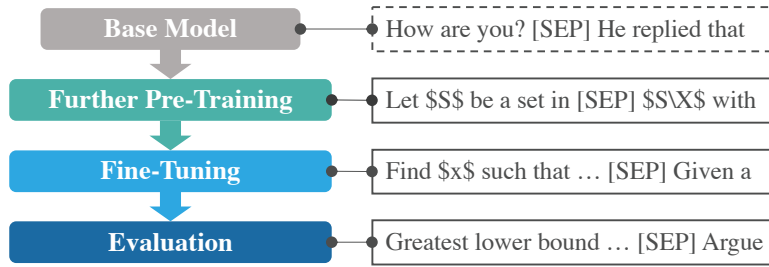


Figure 1: Overview of our approach for Task 1 - Mathematical Answer Retrieval including examples for training and evaluation data.

labels from an annotated corpus. Since we only use GPT-2 to generate tokens and not to predict labels, we will omit the details of this fine-tuning here.

Fine-Tuning

To generate answers given a question, we provide the model with the question tokens and prompt it to complete the text by filling in the answer. During training, the model is prompted using the following pattern:

$$\text{PROBLEM: } Q_1 Q_2 \cdots Q_N \text{ SOLUTION: } ,$$

where Q_i are the question tokens. The model is then optimized to complete the prompt by generating the answer tokens A_i :

$$\text{PROBLEM: } Q_1 Q_2 \cdots Q_N \text{ SOLUTION: } A_1 A_2 \cdots A_M.$$

During evaluation, the model is presented with the same pattern to generate an answer.

5. Contribution to Task 1

Task 1 deals with retrieving the most relevant answers from the MathSE corpus given 100 questions that were not seen during training. For this task we pre-trained and fine-tuned several models using the base models BERT, ALBERT, and RoBERTa, applying different corpora for pre-training and fine-tuning on three different sets of question-answer pairs. An overview of our approach for Task 1 is depicted in Figure 1. In the following, we will first describe the data we used, then our experiments including hyper-parameter settings, and finally present our results.

Pre-Training Data

Prior to pre-training, we applied the official tool provided by ARQMath to read the posts, wrapped formulas in \$ and removed other HTML markup, yielding a list of paragraphs for each post. BERT and ALBERT models rely on data which is separated into sentences during pre-processing for the SOP task. We combined three different strategies: (1) split the text

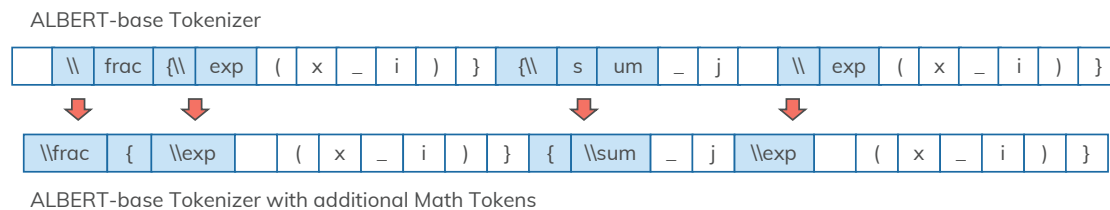


Figure 2: An example of tokenizing the \LaTeX expression $\frac{\exp(x_i)}{\sum_j \exp(x_j)}$, important changes are highlighted.

into sentences, (2) split text into chunks of natural language and formulas and (3) split the mathematical equations on relation symbols (e.g., =, ∈) into parts. The SOP task is designed to work on sentences level granularity to facilitate the modeling of inter-sentence-coherency. Hence, (1) is usually used in various NLP tasks. At the same time, our goal was to increase the model’s understanding of formulas. Therefore, strategy (2) splits a paragraph first into sentences. These sentences are then further split at a formula (with more than three \LaTeX tokens to avoid splitting at e.g., definitions of symbols). In case the remaining text is too short (less than ten characters), it is concatenated to the formula before, separated by a \$ sign. Strategy (3) only uses formula data without natural language. The three strategies will be denoted by MathSE (1), MathSE (2), and MathSE (3), respectively.

Apart from the MathSE corpus provided by the ARQMath Lab, we also pre-processed the Auxiliary Mathematics Problems and Solutions (AMPS) corpus containing questions and answers relating to mathematical problem-solving [6]. Since the data was already split in chunks, we used these data sets as the base for the sentence order task of ALBERT and BERT. The data set contains two parts: the Khan data set consisting of 100,000 exercise questions and answers from the Khan Academy and the Mathematica data set containing 5 M similar questions that are generated using Mathematica Scripts. The questions from both data sets range from topics like simple geometry to multivariate calculus. Both questions and answers use \LaTeX to convey mathematical notation. We used this data only for pre-training, but not for fine-tuning due to its structure.

Tokenizing, creating the pre-training data for each task, i.e., masking tokens and assembling pairs of sentences, and further pre-processing was performed using Huggingface’s libraries transformers and datasets [27, 28]. For our models, we used the released *sentencepiece* vocabulary, but added 501 additional tokens³ to the tokenizer to cover \LaTeX [29]. The list of tokens was taken from the \LaTeX parser by Approach⁴. An example of the impact of the new tokenizer on the expression $\frac{\exp(x_i)}{\sum_j \exp(x_j)}$ can be seen in Figure 2. After we added the \LaTeX tokens to the vocabulary, typical tokens like `\sum` or `\frac` did not get torn apart into multiple tokens, but remain together. Input sequences whose length after tokenization exceeded the maximum number of input tokens were truncated to the maximum length of 512 tokens.

³Our list of additional tokens can be found here: https://github.com/AnReu/ALBERT-for-Math-AR/blob/main/untrained_models/latex_tokens.txt

⁴<https://github.com/approach0/search-engine/blob/master/tex-parser/lexer.template.l>

Fine-Tuning Data

In order to fine-tune our models, we paired each question with up to N correct answers and the same number of incorrect answers. Up to N correct answers were randomly chosen from the answers of the question. Each question in the corpus comes along with tags, i.e. categories indicating the topic of a question such as *sequences-and-series* or *limits*. As an incorrect answer for each question, we picked a random answer from one question sharing at least one tag with the original question by chance. This way, we chose up to N incorrect answers independently from another.

This procedure yields 1.9M examples for $N = 1$ and 2.8M examples for $N = 10$, of which 90% were used as training data for the fine-tuning task. We presented to the model the entire text of the questions and answers using the structure introduced in the previous section. In addition, we pre-trained an ALBERT Model on MathSE (1) and fine-tuned it on $N = 1$. We then let this model predict 1,000 answers to the 2021 test set. We evaluated the answers against the publicly available test set from last year and paired each correct answer with a randomly selected incorrect answer from the model’s results. These question-answer pairs were used as an additional fine-tuning set which we denote by ANNOTATED.

Evaluation Data

To evaluate the trained models, we paired each question of the ARQMath-1 to 3 test sets with each of the answer posts from 2010 to 2018. The question-answer pairs are pre-processed in the same way as the fine-tuning data. Note, that we do not apply pre-filtering or a first-ranking stage as it would be usually done for this kind of cross-encoder design. Instead, we are ranking the entire set of answers. This is possible because we are using a GPU with a greater memory size compared to our submission to ARQMath-2. For the longest queries, ranking the entire set of answers takes around 3h.

5.1. Experimental Setup

In the previous sections, we have introduced several base models, pre-training, and fine-tuning data sets which lead to many combinations for MIR. A summary of our devised models can be found in Table 1. Our submission includes five models which were fine-tuned using the $N = 10$ fine-tuning data set. For these models, we added their official identifiers to the table. The other models are used as baselines and for comparison of our setup. The models MATH_10 and MATH_10_ADD were first pre-trained on MathSE (1), then on MathSE (2), and finally, on MathSE (3). We refer to this pre-training as *mathematical pre-training*. Six other models did not incorporate these two additional data sets for pre-training but were only pre-trained using the first strategy MathSE (1). One model was trained only on the Khan part of the AMPS data set, while two models used a mix of samples from Khan and MathSE. Here, both corpora were combined into a single data set and shuffled. We experimented with the same approach on the entire AMPS corpus and MathSE. For fine-tuning, we denoted on which data set each model was trained. Two models were trained first on the $N = 10$ data. After this training was completed, a second fine-tuning was conducted using ANNOTATED.

All twelve models were trained using eight A100 GPUs with 40 GB GPU memory each. For

Official Identifier	Base Model	Pre-Training Data	Fine-Tuning Data
roberta_10	BERT	MathSE (1)	$N = 1$
	RoBERTa	MathSE (1)	$N = 1$
	RoBERTa	MathSE (1)	$N = 10$
base_10	ALBERT	MathSE (1)	$N = 1$
	ALBERT	MathSE (1)	$N = 10$
	ALBERT	MathSE (1)	$N = 10 + \text{ANNOTATED}$
math_10	ALBERT	MathSE (1) - (3)	$N = 10$
math_10_add	ALBERT	MathSE (1) - (3)	$N = 10 + \text{ANNOTATED}$
Khan_SE_10	ALBERT	Khan	$N = 1$
	ALBERT	Khan + MathSE mixed	$N = 1$
	ALBERT	Khan + MathSE mixed	$N = 10$
	ALBERT	AMPS + MathSE mixed	$N = 1$

Table 1
Model Configurations for Task 1.

pre-training, a batch size of 16 samples per GPU was used. We pre-trained the models for 13 epochs using MathSE (1) and 9 epochs on MathSE (2). MathSE (3) added additional 20 epochs to the model. Fine-tuning on $N = 1$ and $N = 10$ used a batch size of 32 examples per device, and 200 warm-up steps with a learning rate of $2e^{-05}$. ANNOTATED used the same hyperparameters, but a batch size of 32 in total. Pre-training and fine-tuning were performed using Huggingface’s library transformers [27].

5.2. Evaluation

This section summarizes our results using the different setups. We start by presenting our overall results of the models submitted to the lab and then discuss the details of choosing the base model, the pre-training, and fine-tuning data.

5.2.1. Overall Results

The results of our runs submitted to Task 1 of the ARQMath Lab 2022 are presented in Tables 2 and 3. Regarding nDCG’ and mAP’, MATH_10, our model using mathematical pre-training performs the best in all three years. The model which was fine-tuned using ANNOTATED received the highest scores for p’@10, but its performance on the other two metrics degraded. Since it was fine-tuned on the ARQMath 2021 test set, the scores on this set are naturally much higher than the models which were not fine-tuned on this data. The other three models of our submission are on par even though they were trained on different data and with a different base architecture. Nevertheless, our models for the submission to ARQMath-3 outperform even the best models from ARQMath-2 in all three metrics. In comparison to other participants of ARQMath-3, our MATH_10_ADD received the highest p’@10 scores among all automatic runs. In the following, we will analyze different aspects of improvements of our submission.

	Official Identifier	ARQMath 2020			ARQMath 2021		
		nDCG'	mAP'	p'@10	nDCG'	mAP'	p'@10
Submissions 2022	math_10_add	0.421	0.264	0.405	(0.566)	(0.445)	(0.589)
	math_10	0.446	0.268	0.392	0.454	0.228	0.321
	roberta_10	0.438	0.254	0.372	0.446	0.224	0.309
	Khan_SE_10	0.437	0.254	0.357	0.437	0.214	0.309
	base_10	0.438	0.252	0.369	0.434	0.209	0.299
ARQMath 2021 Participants							
TU_DBS (2021)	primary	0.380	0.198	0.316	0.377	0.158	0.227
Math Dowser (2021)	primary	0.433	0.191	0.249	0.434	0.169	0.211
DPRL (2021)	QASim	0.417	0.234	0.369	0.388	0.147	0.193

Table 2
Results of Task 1.

	Official Identifier	ARQMath 2022		
		nDCG'	mAP'	p'@10
Submissions 2022	math_10_add	0.379	0.149	0.278
	math_10	0.436	0.158	0.263
	roberta_10	0.413	0.150	0.226
	Khan_SE_10	0.426	0.154	0.236
	base_10	0.423	0.154	0.228

Table 3
Results of Task 1.

5.2.2. Base Model

We evaluated models trained on three base architectures: BERT, ALBERT, and RoBERTa. The results can be found in Table 4. Even though ALBERT and RoBERTa are considered to be advancements over BERT, their performance on our downstream task is not necessarily higher. RoBERTa receives the highest scores for nDCG' and p'@10, while BERT scores highest using the metric mAP'. Overall, the improvements of the three architectures over each other are rather minimal. However, the training time should also be considered:

Pre-training ALBERT on (1) MathSE took 24h, while BERT and RoBERTa needed on average 25% more time. To fine-tune each model, ALBERT was the fastest with 8h on the $N = 1$ data set. The fine-tuning of BERT and RoBERTa on the same data set took 11h. Evaluation takes the same time on average for each of the three models because the data is processed by the same number of layers since ALBERT's layer sharing is only beneficial during training and BERT and RoBERTa share the same underlying architecture.

5.2.3. Additional Pre-Training Data

Transformer-Encoder models are known to benefit from more pre-training data which is why we evaluate the ALBERT model on four different data set configurations whose results are

	ARQMath Lab 2020		
	nDCG'	mAP'	p'@10
BERT	0.4068	0.2411	0.3560
ALBERT	0.4122	0.2335	0.3587
RoBERTa	0.4157	0.2328	0.3676

Table 4

Comparison of results of BERT, ALBERT and RoBERTa as base models.

presented in Table 5. Interestingly, the model trained on a mixed data set consisting of data from the Khan Academy and the MathSE scores best, receiving slightly better scores on nDCG' and mAP'. For p'@10 the model trained only on MathSE outperforms the other models, indicating that it is able to place relevant documents better within the top 10 documents, while the first model ranked relevant documents better in the long run.

The model trained only on data from Khan Academy scored worst in this evaluation demonstrating the shortcomings of out-of-domain data. A reason for this behavior could be that the questions from Khan are designed to serve as exercises. Therefore, each sentence is relevant for solving the question and does not contain any irrelevant information that could be included by question authors on MathSE (e.g. "Dear community, I have a question ..."). After training on Khan data only, it could be harder for the model to deal with these irrelevant information in questions.

Base Model	Dataset	ARQMath Lab 2020		
		nDCG'	mAP'	p'@10
ALBERT	MathSE (1)	0.4122	0.2335	0.3587
ALBERT	Khan	0.3716	0.1852	0.2947
ALBERT	Khan+MathSE (1)	0.4164	0.2356	0.3373
ALBERT	AMPS+MathSE (1)	0.4052	0.2256	0.3400

Table 5

Comparison of results for pre-training using different data sets.

5.2.4. Fine-Tuning Data

When comparing the amount of fine-tuning data needed for the answer retrieval task, we can see in Table 6 that more data is clearly beneficial. In both cases, for ALBERT and RoBERTa, we see an increase on all three metrics when fine-tuning on $N = 10$ instead of $N = 1$ is applied. With an additional training on ANNOTATED, only p'@10 is increased, while the other two metrics deteriorate. This indicates that the model can differentiate better between relevant and non-relevant answers in the top 10, but fails to place other relevant documents in good positions afterwards.

We also report the scores on nDCG' for the three categories 'Both', 'Math', and 'Text' indicating which of these parts are most crucial for answering the question. For example, a question based in the category 'Text' would require to understand the written text of the question over the

Base Model	Fine-Tuning Data	ARQMath Lab 2020					
		nDCG'	mAP'	p'@10	Both	Math	Text
ALBERT	$N = 1$	0.4122	0.2335	0.3587	0.4202	0.4033	0.4140
ALBERT	$N = 10$	0.4377	0.2519	0.3693	0.4391	0.4437	0.4184
ALBERT	$N = 10+\text{ANNOTATED}$	0.3988	0.2435	0.3853	0.3837	0.4220	0.3789
RoBERTa	$N = 1$	0.4157	0.2328	0.3676	0.4175	0.4200	0.3999
RoBERTa	$N = 10$	0.4376	0.2543	0.3720	0.4293	0.4511	0.4246

Table 6
Comparison of results for fine-tuning using different data sets.

mathematical formulas. Models which were trained using $N = 10$ data showed improvements in all three categories, but the 'Math' category benefited the most. An explanation for this observation could be that for $N = 1$ the model saw fewer irrelevant examples that shared the same notation (same mathematical symbols) as the question, but which were used in a different and therefore irrelevant way for the question. In the $N = 10$ data set, it was more probable that an irrelevant question still used the same mathematical symbol. Therefore, using this data set, the model needed to learn the semantics of the usage of the symbols, rather than their mere appearance. However, a similar observation should be possible for fine-tuning on ANNOTATED, but by adding this fine-tuning, the model's performance on all three categories degrades.

6. Contribution to Task 3

Task 3 was first introduced to the ARQMath Lab in 2022 and has the goal of generating answers given a question instead of retrieving them from a corpus. The questions are identical to the ones for Task 1. All include at least one formula.

In the following, we will introduce our approach of generating answers using GPT-2 by fine-tuning it on two corpora. An overview of the approach is illustrated in Figure 3. We start by describing the data which we used, then our experimental setup and finally, present our results.

6.1. Data

For fine-tuning GPT-2, we use the same two data sets as for Task 1: the MathSE data set as provided by the ARQMath Lab and the AMPS data set consisting of questions-answer pairs

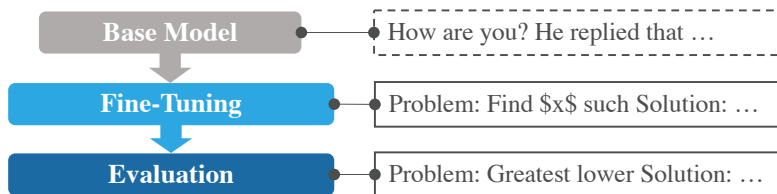


Figure 3: Overview of our approach for Task 3 - Mathematical Answer Generation including examples for fine-tuning and evaluation data.

Official Identifier	Training Setup	Beam Size	Hints	Length Penalty	Sampling
amps3_se1_hints	3 ep. AMPS + 1 ep. MathSE	5	True	1	False
se3_len_pen_10	3 ep. MathSE	10	False	2	False
amps3_se1_len_pen_20_sample_hint	3 ep. AMPS + 1 ep. MathSE	20	True	2	True
shortest	*	*	*	*	*

Table 7

Model Configurations for Task 3, ep. denotes the number of epochs.

from the Khan Academy and generated questions with step-by-step answers using Mathematica. In total, we fine-tuned our models on 1,445,487 question-answer pairs from MathSE, where for each question a single answer was chosen by chance. In addition, the AMPS data set consists of 627,795 question-answer pairs. For pre-processing, we used the tokenizer provided by the authors of AMPS, which is based on the original GPT-2 tokenizer, but separated compounds of digits into single digits. We also experimented with the original GPT-2 tokenizer, but found the adapted one to perform better.

6.2. Experimental Setup

A summary of our experiments can be found in Table 7. We experimented with fine-tuning the models on two data sets for a different number of epochs. We tested to train only on the MathSE data for three epochs, while another model was first fine-tuned on the AMPS data set for three epochs and afterwards for one epoch on MathSE. In addition, also smaller numbers of epochs were tested but did not yield better results. For training on AMPS, we sub-sampled the amount of training data for the Mathematica part to 0.5 and for Khan to 5 following the procedure of [6].

For decoding, we varied the length penalty between 1 and 2 and applied beam search with a beam size of 5, 10, and 20. We also experimented with top-k sampling. Apart from these modifications, we followed the training and evaluation recommendations reported by [6]. Because the length of the generated answers exceeded the allowed maximum length for the submission to the ARQMath Lab 2022 in several cases, we tried to force the model to generate shorter, but still relevant answers by adding the word 'HINT' to the beginning of the solution during decoding. The data set for the training was not altered. These combinations in total led to three experiments. The fourth one is a combined run which includes the shortest generated answer of each of the three runs. This run is denoted by 'shortest'.

For all experiments for Task 3, we adapted the code by [6] for our data set which is based on Huggingface Transformers [27].

Official Identifier	AR	p@1
amps3_se1_hints	0.325	0.078
se3_len_pen_10	0.244	0.064
amps3_se1_len_pen_20_sample_hint	0.231	0.051
shortest	0.205	0.026

Table 8
Results of Task 3.

6.3. Results

Table 8 displays our efforts for Task 3 of ARQMath-3. Our best model was trained on the AMPS data and afterward on MathSE. During decoding, we use the word 'HINT' to force the model to generate shorter answers. Surprisingly, this run scored better in both metrics than the one that was trained in the same way but used a higher length penalty, sampling, and a beam size of 20. The model which was trained on MathSE only ranks in second place for both metrics. The lowest scores received the run that consists of the shortest answers of our models for each topic. This indicates that shorter answers may be insufficient to convey enough relevant information in the post. Since an automatic evaluation of answer generation is challenging, we will not analyze the impact of different aspects of our submission but instead report in the following some results in the context of a qualitative evaluation.

Below, we present two examples of questions from the test set with their generated answers by our primary submission. In general, the model is able to pick up the topic from the questions and generate meaningful, syntactically correct answers in most cases. However, whether the answers are relevant for solving the questions needs further evaluation. For example, the generated answer for Question 322 is simply a copy of a part of the question which does not seem to provide any additional information. This behavior of copied information can also be seen in other answers. Especially, when there was an edit of the author where they already provided a correct answer, the generated answer would only copy this solution without commenting. A similar issue can be seen in the generated solution for Question 340. The model only arranges the question differently in several steps but does not comment on the process. Whether and why this equation holds, can not directly be inferred from the model's answer. More examples of model outputs can be seen in the Appendix.

Post ID 322:

Title: How do I calculate the sum of sum of triangular numbers?

Question: As we know, triangular numbers are a sequence defined by $\frac{n(n+1)}{2}$. And it's first few terms are 1, 3, 6, 10, 15.... Now I want to calculate the sum of the sum of triangular numbers. Let's define

$$a_n = \frac{n(n+1)}{2}$$

$$b_n = \sum_{x=1}^n a_x$$

$$c_n = \sum_{x=1}^n b_x$$

And I want an explicit formula for c_n . After some research, I found the explicit formula for $b_n = \frac{n(n+1)(n+2)}{6}$. Seeing the patterns from a_n and b_n , I figured the explicit formula for c_n would be $\frac{n(n+1)(n+2)(n+3)}{24}$ or $\frac{n(n+1)(n+2)(n+3)}{12}$.

Then I tried to plug in those two potential equations,

If $n = 1$, $c_n = 1$, $\frac{n(n+1)(n+2)(n+3)}{24} = 1$, $\frac{n(n+1)(n+2)(n+3)}{12} = 2$. Thus we can know for sure that the second equation is wrong.

If $n = 2$, $c_n = 1 + 4 = 5$, $\frac{n(n+1)(n+2)(n+3)}{24} = 5$. Seems correct so far.

If $n = 3$, $c_n = 1 + 4 + 10 = 15$, $\frac{n(n+1)(n+2)(n+3)}{24} = \frac{360}{24} = 15$

Overall, from the terms that I tried, the formula above seems to have worked. However, I cannot prove, or explain, why that is. Can someone prove (or disprove) my result above?

Generated Solution: HINT: $\frac{n(n+1)(n+2)(n+3)}{24} = \frac{360}{24} = 15$

Post ID 340:

Title: I have the following problem: Let $|x_{n+1} - x_n| < 1/3^n$. Show that (x_n) is a Cauchy sequence.

Question: We have that (x_n) is a sequence of real numbers. And the relation on the title:

$$|x_{n+1} - x_n| < \frac{1}{3^n}.$$

We must prove that this is a Cauchy sequence.

I know that an Cauchy sequence follows the definition:

given $\epsilon > 0$, exists $n_0 > 0$, such that $m, n > n_0 \Rightarrow |x_m - x_n| < \epsilon$

But I don't know how to use both informations to prove the exercise.

If someone please may help me, I'd be very thankful.

Generated Solution: $|x_{n+1} - x_n| = \left| \frac{x_{n+1} - x_n}{3^{n+1}} \right| = \frac{|x_{n+1} - x_n|}{3^{n+1}} < \frac{1}{3^{n+1}}$.

7. Conclusion

In our contribution to this lab, we demonstrated the use of Transformer-Encoders and Transformer-Decoders for two mathematical question-answering tasks. We pre-trained and fine-tuned twelve models based on BERT, RoBERTa, and ALBERT for the retrieval of answers given a mathematical question for the ARQMath Lab 2022. Our results show significant improvements compared to the 2021 edition of this lab, which can be attributed to better pre-training

and the enlarged fine-tuning data. In addition, we analyzed several pre-training data sets and found that the Khan data set showed slight improvements in two out of three metrics. Finally, we improved our $p@10$ results even further by additionally fine-tuning on annotated test data from ARQMath-2. For Task 3, a GPT-2 model was fine-tuned on two data sets. The results for this task are not yet published, but first analyses showed that the model is able to capture the topic of the question and can generate syntactically correct answers. The limited length of the generated answers is still an issue, which should be addressed in future research.

Acknowledgments

This work was supported by the DFG under Germany’s Excellence Strategy, Grant No. EXC-2068-390729961, Cluster of Excellence “Physics of Life” of TU Dresden. Furthermore, the authors are grateful for the GWK support for funding this project by providing computing time through the Center for Information Services and HPC (ZIH) at TU Dresden. We would also like to thank the reviewers for their helpful comments and recommendations.

References

- [1] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).
- [2] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut, Albert: A lite bert for self-supervised learning of language representations, arXiv preprint arXiv:1909.11942 (2019).
- [3] H. Jo, D. Kang, A. Head, M. A. Hearst, Modeling mathematical notation semantics in academic papers, in: Findings of the Association for Computational Linguistics: EMNLP 2021, 2021, pp. 3102–3115.
- [4] A. Reusch, M. Thiele, W. Lehner, Tu_dbs in the arqmath lab 2021, clef, in: CEUR Workshop Proceedings, 2021. <http://ceur-ws.org/Vol-2936/paper-07.pdf>.
- [5] W. Zhong, J.-H. Yang, J. Lin, Evaluating token-level and passage-level dense retrieval models for math information retrieval, arXiv preprint arXiv:2203.11163 (2022).
- [6] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, J. Steinhardt, Measuring mathematical problem solving with the math dataset, arXiv preprint arXiv:2103.03874 (2021).
- [7] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., Language models are unsupervised multitask learners, OpenAI blog 1 (2019) 9.
- [8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in Neural Information Processing Systems 30 (2017) 5998–6008.
- [9] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, arXiv preprint arXiv:1907.11692 (2019).
- [10] I. Beltagy, K. Lo, A. Cohan, Scibert: A pretrained language model for scientific text, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing

and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 3606–3611.

- [11] E. Alsentzer, J. R. Murphy, W. Boag, W.-H. Weng, D. Jin, T. Naumann, W. Redmond, M. B. McDermott, Publicly available clinical bert embeddings, NAACL HLT 2019 (2019) 72.
- [12] K. Huang, J. Altsosaar, R. Ranganath, Clinicalbert: Modeling clinical notes and predicting hospital readmission, arXiv preprint arXiv:1904.05342 (2019).
- [13] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, et al., Codebert: A pre-trained model for programming and natural languages, arXiv preprint arXiv:2002.08155 (2020).
- [14] A. Kanade, P. Maniatis, G. Balakrishnan, K. Shi, Learning and evaluating contextual embedding of source code, in: International Conference on Machine Learning, PMLR, 2020, pp. 5110–5121.
- [15] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, Improving language understanding by generative pre-training (2018). <https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf>.
- [16] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, Advances in neural information processing systems 33 (2020) 1877–1901.
- [17] S. Peng, K. Yuan, L. Gao, Z. Tang, Mathbert: A pre-trained model for mathematical formula understanding, arXiv preprint arXiv:2105.00377 (2021).
- [18] S. Rohatgi, J. Wu, C. L. Giles, Psu at clef-2020 arqmath track: Unsupervised re-ranking using pretraining, in: CEUR Workshop Proceedings. Thessaloniki, Greece, 2020. http://ceur-ws.org/Vol-2696/paper_121.pdf.
- [19] V. Novotný, P. Sojka, M. Štefánik, D. Lupták, Three is better than one, in: CEUR Workshop Proceedings. Thessaloniki, Greece, 2020. http://ceur-ws.org/Vol-2696/paper_235.pdf.
- [20] S. Rohatgi, J. Wu, C. L. Giles, Ranked list fusion and re-ranking with pre-trained transformers for arqmath lab (2021). <http://ceur-ws.org/Vol-2936/paper-08.pdf>.
- [21] V. Novotný, M. Štefánik, D. Lupták, M. Geletka, P. Zelina, P. Sojka, Ensembling ten math information retrieval systems (2021). <http://ceur-ws.org/Vol-2936/paper-06.pdf>.
- [22] P. Dadure, P. Pakray, S. Bandyopadhyay, Bert-based embedding model for formula retrieval, CLEF, 2021. <http://ceur-ws.org/Vol-2936/paper-03.pdf>.
- [23] B. Mansouri, D. W. Oard, R. Zanibbi, Dprl systems in the clef 2021 arqmath lab: Sentencebert for answer retrieval, learning-to-rank for formula retrieval (2021). <http://ceur-ws.org/Vol-2936/paper-04.pdf>.
- [24] S. Polu, I. Sutskever, Generative language modeling for automated theorem proving, arXiv preprint arXiv:2009.03393 (2020).
- [25] B. Mansouri, V. Novotný, A. Agarwal, D. W. Oard, R. Zanibbi, Overview of ARQMath-3 (2022): Third CLEF lab on Answer Retrieval for Questions on Math (Working Notes Version), in: G. Faggioli, N. Ferro, A. Hanbury, M. Potthast (Eds.), Working Notes of CLEF 2022 - Conference and Labs of the Evaluation Forum, 2022.
- [26] R. Zanibbi, A. Aizawa, M. Kohlhase, I. Ounis, G. Topic, K. Davila, Ntcir-12 mathir task overview., in: NTCIR, 2016. <https://www.cs.rit.edu/~rlaz/files/ntcir12-mathir.pdf>.
- [27] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao,

$$\begin{aligned}
\int_a^x f(t) dt + \int_a^x f(t) dt - \int_a^x f(t) dt &= \int_a^x f(t) dt + \int_a^x f(t) dt - \int_a^x f(t) dt = \\
\int_a^x f(t) dt + \int_a^x f(t) dt - \int_a^x f(t) dt &= \int_a^x f(t) dt + \int_a^x f(t) dt - \int_a^x f(t) dt = \\
\int_a^x f(t) dt + \int_a^x f(t) dt - \int_a^x f(t) dt &= \int_a^x f(t) dt + \int_a^x f(t) dt -
\end{aligned}$$