TU_DBS in the ARQMath Lab 2021, CLEF

Anja Reusch, Maik Thiele and Wolfgang Lehner

Database Systems Group, Technische Universität Dresden, Germany

Abstract

Mathematical Information Retrieval (MIR) deals with the task of finding relevant documents that contain text and mathematical formulas. Therefore, retrieval systems should not only be able to process natural language, but also mathematical and scientific notation to retrieve documents. The goal of this work is to review the participation of our team in the ARQMath 2021 Lab where two different approaches based on ALBERT and ColBERT were applied to a Question Answer Retrieval task and a Formula Similarity task. The ALBERT-based classification approach received competitive results for the first task. We found that by pre-training on data separated in chunks of text and formulas, the model performed better on formula data. This way of pre-training could also be beneficial for the Formula Search task.

Keywords

Mathematical Language Processing, Information Retrieval, BERT-based Models

1. Introduction

With the rising number of scientific publications and mathematics-aware online communities available Mathematical Information Retrieval has become more important since many of these documents and posts not only use natural language, but also mathematical notation to communicate. Only interpreting natural language is not sufficient for retrieval in such documents anymore since the usage of mathematical notation is crucial to understand the information conveyed by the author. Hence, in order to search or retrieve information from these platforms, a retrieval system needs to understand the notation of mathematical expressions.

The ARQMath Labs 2020 [1] and 2021 [2] have two related aims: Task 1 deals with the retrieval of relevant answers given a question from the Mathematics StackExchange Community. This task involves understanding the problem of the question poster in terms of natural language in combination with mathematical notation in form of LaTeX, Symbol Layout Trees (SLTs) or Operator Trees (OPTs). For Task 2 on the other hand, the participants were required to develop a system that returns relevant formulas given a query formula.

For Natural Language based Information Retrieval systems based on large pre-trained language models such as BERT [3] have been found to be effective and out-performed previous, traditional IR systems that were based on string matching methods [4]. In a previous work, we showed that our approach of using an ALBERT-based classifier as a similarity measure is

CLEF 2021 – Conference and Labs of the Evaluation Forum, September 21–24, 2021, Bucharest, Romania

anja.reusch@tu-dresden.de (A. Reusch); maik.thiele@tu-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)

CEUR Workshop Proceedings (CEUR-WS.org)

beneficial for Mathematical Question Answering when answers depend on the written text [5]. However, when answering the question depends on proper interpretation of the formulas, traditional methods are still more suitable.

The second disadvantage is that the average query latency of BERT-based systems is a few orders of magnitude larger compared to non-neural methods [6, 7], due to the fact that for each query-document pair an entire forward pass through the deep network needs to be performed. A recent advance in terms of speed without neglecting performance is ColBERT [7], where the authors applied a late interaction mechanism to assess the relevance of a document given a query. This approach made offline indexing of the collection and a faster evaluation possible, since only one forward pass of the query is necessary.

Furthermore, our ALBERT-based models have only been applied to Task 1, the retrieval of answers given a textual query question. But the application of this approach to formula retrieval, such as in Task 2, has not been tested yet.

Therefore, with our participating in this year's ARQMath Lab we would like to address the following three areas:

- Pre-training adjustments in order to increase the models' performance on formula understanding
- Faster evaluation by using ColBERT
- Application of our ALBERT-based approach to formula retrieval

This work is structured as follows: We will first introduce the ARQMath 2021 Lab and then review relevant literature for Information Retrieval and BERT-based systems for natural language and multi-modal tasks. In Section 4 the overall architecture of our approach will be explained. Section 5 and Section 6 introduce Task 1 and Task 2, respectively. We will describe the data set we used to pre-train and fine-tune the different models including a description of the experiments and discuss their results. Finally, the last section summarizes our work.

2. ARQMath 2021 Lab

The aim of ARQMath Lab 2021 [2] is to accelerate the research in mathematical Information Retrieval and includes two related, but different tasks: Task 1 involves the retrieval of relevant answer posts for a question asked on the Mathematics StackExchange, which is a platform where users post questions to be answered by the community. The questions should be related to mathematics topics at any level[1]. Users have the possibility to add mathematical formulas to their post to clarify their questions. These formulas are written in LaTeX notation. Task 2 is built on top of the same data, but with a different goal in mind: Participants are expected to retrieve relevant formulas given a query formula in context of its post. This task is related to the formula browsing task of NTCIR-12 [8]. The participating teams submitted for each topic 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

[1] https://math.stackexchange.com

(0 and 1: not relevant, 2 and 3: relevant). The relevance assessment was performed by pooling after the teams submitted their results.

ARQMath 2021 provides data from the Mathematics StackExchange including question and answer posts from 2010 to 2018. In total, the collection contains 1 M questions and 1.4 M 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. Each formula got assigned a formula id and a visual ids. Formulas sharing the same visual appearance received the same visual id. Apart from this corpus of posts and formulas that are available for training and evaluating models, also a test set of queries is released by the organizers of ARQMath. The query topics of 2020 and 2021 contain 99 and 100 topics, repectively, which are question posts including title, text and tags. In the 2020 test set 77 queries were evaluated for Task 1 and 45 formula queries for Task 2, while the evaluation of Task 1 in 2021 included 71 queries and 58 for Task 2.

3. Related Work

Bidirectional Encoder Representations from Transformers (BERT) is an architecture based on the encoder of a Transformer model which was designed for language modelling [3]. Due to the success of this and other pre-trained, Transformer-based language models, BERT has been a basis in many systems for Natural Language Understanding (NLU) tasks and applications in Information Retrieval. Hence, there exist several advanced versions such as RoBERTa [9] or ALBERT [10] with the goal to optimize BERT's performance.

The influence of in-domain pre-training has been analyzed by Gururangan et al.[11] who found that this is especially valuable when the domain vocabulary has a low overlap with the pre-training data. As a consequence, various models for different domains have been developed, such as BioBERT [12], ClinicalBERT [13, 14] or SciBERT [15] for scientific domains or CuBERT [16] and CodeBERT [17]. A notable difference between these models is that BioBERT, ClinicalBERT and CodeBERT use the original vocabulary that their base model was pre-trained on while SciBERT and CuBERT trained their own domain specific vocabulary. However, each of these models could demonstrate its improvements compared to the original models without domain specific pre-training.

BERT-based models for mathematical domains have also been studied with the most recent example being MathBERT [18]. In addition, during the last ARQMath Lab 2020, two teams submitted systems based on BERT and RoBERTa [19, 20]. Both teams used the models to generate post embeddings for a given question and all answers. Their similarity is calculated by comparing the vectors using cosine similarity.

Shortly after BERT outperformed previous approaches in various NLU tasks, it was also successfully applied to Information Retrieval. The model by Nogueira et al. classified its input consisting of a query and one document for their relevance resulting in a score that can be used to rank multiple documents [4]. This approach achieved state-of-the-art performance, but was much slower and computationally expensive than previous systems, because one forward pass through the entire deep neural network was necessary to score one query-document pair. Nevertheless, this approach has also been proven to be effective for the multi-modal retrieval of

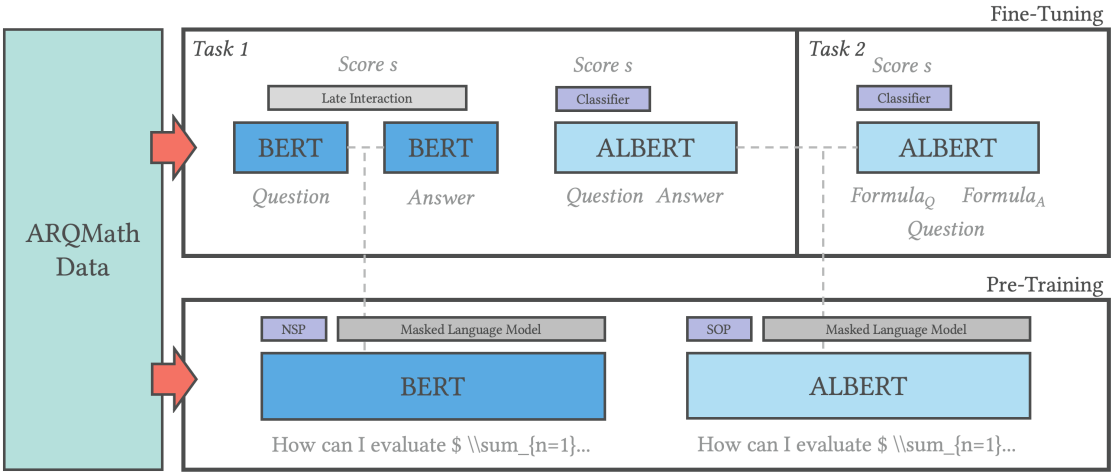

Figure 1: Overview of our pre-training (details in Figure 2) and fine-tuning procedure (details in Figure 3 and 4).

source code [17] and was also applied to Mathematical Question Answering using an ALBERT model trained and evaluated on the ARQMath Lab 2020 test set [5]. The evaluation results were also broken down to the categories determining which part of the question influenced answering it the most. The model showed the best performance when answering the question depended on the written text. But for questions relying on formulas the results were worse than systems based on non-neural methods. Therefore, the modeling capability of formulas needs to be improved to also be able to capture their semantics in a better way.

Due to the fact that the ranking model by Nogueira et al. came with a steep increase in computational cost, recent research focused on improving the evaluation time without neglecting its performance gains. Despite there is more than one model dealing with this challenge, we will focus in this work on the approach by Khattab et al. called ColBERT [7]. ColBERT uses a BERT model to separately encode a query and a document and then apply a novel late interaction mechanism to calculate the similarity. This way they achieved competitive results when re-ranking on the popular MSMARCO data set [21] with a latency of 61 ms compared to 107,000 ms using the BERT-based approach by Nogueira et al.

4. Model Architecture

BERT-based models have proven to be effective in Natural Language Understanding and Information Retrieval tasks. Their strength was also shown in scenarios where not only natural language plays an important role, such as Code Retrieval or Mathematical Language Processing as in this lab [17, 5, 18]. Building on top of these achievements, we apply two deep neural models based on the popular BERT in our submission: ALBERT and ColBERT. ALBERT is an even more recent model based on BERT, which is optimized by factorization of the embeddings and parameter sharing between layers. The general idea of our first approach is to employ the

ALBERT model to determine the similarity score between two snippets, for Task 1 a question and an answer and for Task 2 two formulas with context. This is achieved by fine-tuning a classifier on top of the pre-trained ALBERT model which predicts how well the two snippets match. We apply ALBERT for this approach, because its optimizations result in less training parameters and therefore a lower memory consumption and accelerated training speed compared to BERT. The second method that we apply for Task 1 uses a BERT model as a basis of ColBERT. The query and each document are passed through BERT separately in order to encode their respective content. This way an offline computation of the representations of each document is possible beforehand. The late interaction mechanism in form of the L2 distance is applied to aggregate and compare the contextualized embeddings. Finally, the documents are ranked by this computed L2 distance.

The success of BERT and BERT-based models is attributed to its transformer architecture [22] and also to the self-supervised pre-training on large amounts of data. In this work, we will focus on the latter aspect and pre-train models on different data highlighting its influence. The overall process of our approach is depicted in Figure 1. We will present details about the pre-training and fine-tuning in the next sections.

5. Task 1

The goal of Task 1 is the retrieval of an answer post from 2010 - 2018 to questions that were posted in 2019. The ARQMath Lab 2021 added a second set of 100 questions asked in 2020. The optimal answers retrieved by the participants are expected to answer the complete question on its own. This relevance of each question was assessed by reviewers during a pooling process. In the following sections we will present our two approaches for this mathematical question answering task. We will first explain the models we used, then how we processed the data corpus for pre-training and fine-tuning. Finally, we give details on our experiments, the results and a comparison to other models.

5.1. Pre-Training

As mentioned previously, BERT and also ALBERT rely on pre-training on rather simple tasks. BERT is pre-trained using two objectives to obtain general understanding of language: the masked language model (MLM) and the next sentence prediction (NSP).

Pre-training is performed on a sentence-level granularity. Each sentence S is split into tokens: $S = w_1 w_2 \cdots w_N$. Before inputting the sentence into the model, each token w_i is embedded using a sum of three different embeddings, the word embedding t_i encoding the semantic of the token, the position embedding p_i denoting its position within the sentence, and the segment embedding s_i in order to discern between the first and the second segment when the model is presented e.g., two sentences as for the NSP task. The segment embeddings will also help our model to differentiate between the query and document as the two segments later. All three embeddings are added up to form the input embedding E_i for each token:

$$E_i = t_i + p_i + s_i.$$

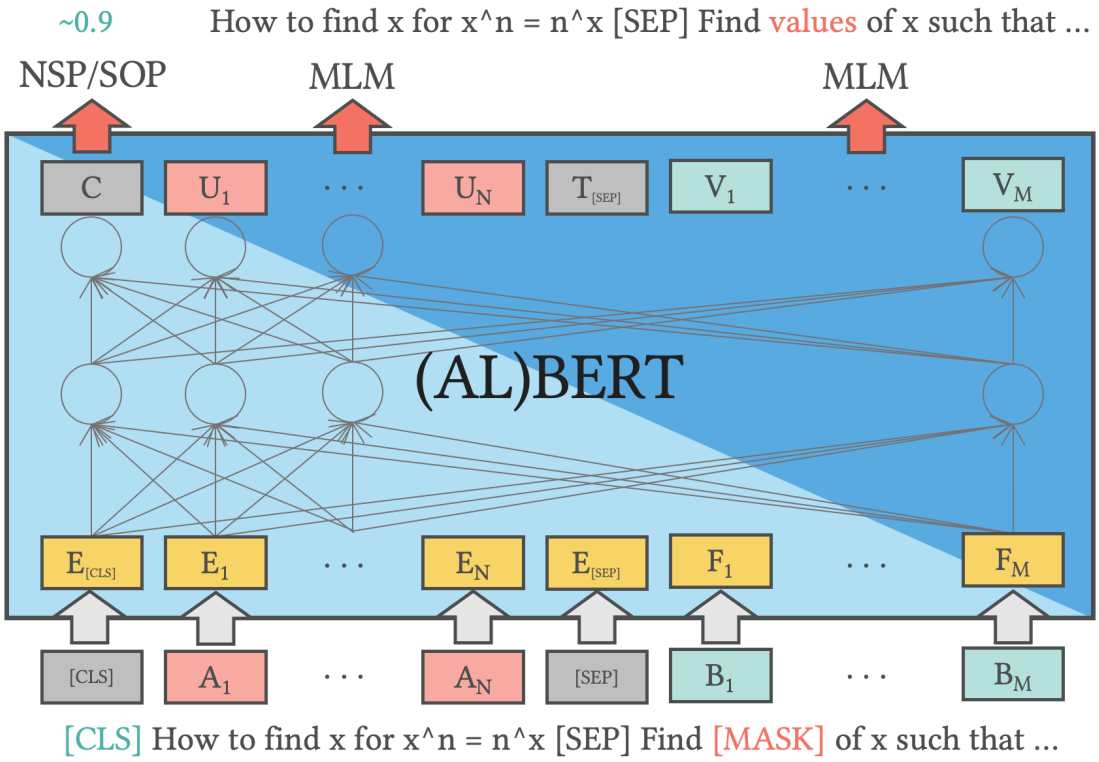

Figure 2: BERT's and ALBERT's pre-training process, 0.9 symbolizes the NSP or SOP score for the two sentences, the red word 'values' is the predicted word for the masked token.

In order to obtain a representation of the entire input, we prepend the sentence S with a classification token $w_S = \langle CLS \rangle$. It is embedded in the same way as the other tokens and will be used for the NSP task and also for fine-tuning tasks that rely on a representation of the input such as classification.

The first pre-training task is the masked language model meaning tokens from the input sentence are randomly replaced by a $\langle MASK \rangle$ token, a different token or is not changed at all. After embedding the input, it is feed into the BERT model, consisting of 12 layers of transformer encoder blocks, resulting in a contextualized output embedding U_i for each input token:

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

where E_{CLS} and C are the input and output embeddings of the $\langle CLS \rangle$ token. Afterwards, a simple classifier is applied in order to predict the original word from 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.

The next sentence prediction objective predicts whether the sentence given to the model as the first segment S_A appears in a text before the sentences given to the model as the second segment S_B (label 1) or whether the second sentence is a random sentence from another document (label 0). This task is performed as a binary classification using the output embedding C as its input:

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

where the matrix W_{NSP} and the bias b_{NSP} are only used for the NSP and are not used otherwise later.

ALBERT also makes use of the MLM objective, but it has been found that NSP, predicting whether the second sentence in the input is swapped with a sentence from another document from the corpus, is a relatively challenging task and was changed to the sentence order prediction (SOP). Here, the model is asked to determine what the correct order of two presented sentences is. Hence, the model is presented with two sentences and performs their classification in the same way as BERT's NSP. Therefore, the formulas for NSP as introduced above apply as well.

The pre-training process of BERT and ALBERT is depicted together in Figure 2. Note, that BERT applies a classification on the output embedding C for the NSP objective, while ALBERT does the same for the SOP objective. Both models use the MLM objective.

Pre-Training Data

Before 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 sentence separated data during pre-processing for the NSP and SOP tasks. Two different strategies were tested: (1) split the text into sentences, (2) split it into chunks of text and formulas. The SOP task is designed to work with sentences. Hence, (1) is usually used in various NLP tasks. On the other hand, our goal was to increase the model's understanding of formulas. Therefore, strategy (2) splits a paragraph first into sentences, but also when a sentences contains 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. Before inputting the data into the models, tokenizing, creating the pre-training data for each task, i.e., masking tokens and assembling pairs of sentences, and further pre-processing was performed by the pre-processing scripts provided in the official BERT and ALBERT repositories[2]. For the models that started from official checkpoints, we used the released sentencepiece vocabulary [23]. For the models that started from scratch, we trained our own sentencepiece model using the parameters recommended in the ALBERT repository which had a vocabulary overlap of 32.1% compared to the released sentencepiece vocabulary for ALBERT. Sentencepiece tokenizes the input into subwords using byte-pair-encoding[24], e.g., the sentence 'how can i evaluate $ \sum_{n=1}^\infty \frac{2n}{3^{n+1}} $?' would be tokenized into 'how can i evaluate $ \ sum _ { n = 1 } ^ \ in ##ft ##y \ fra ##c { 2 ##n } { 3 ^ { n + 1 } } $?' by the BERT tokenizer, where single tokens are separated by spaces. Input sequences whose length after tokenization exceeded the maximum number of input tokens where truncated to the maximum length. In case two

[2] https://github.com/google-research/bert, https://github.com/google-research/ALBERT

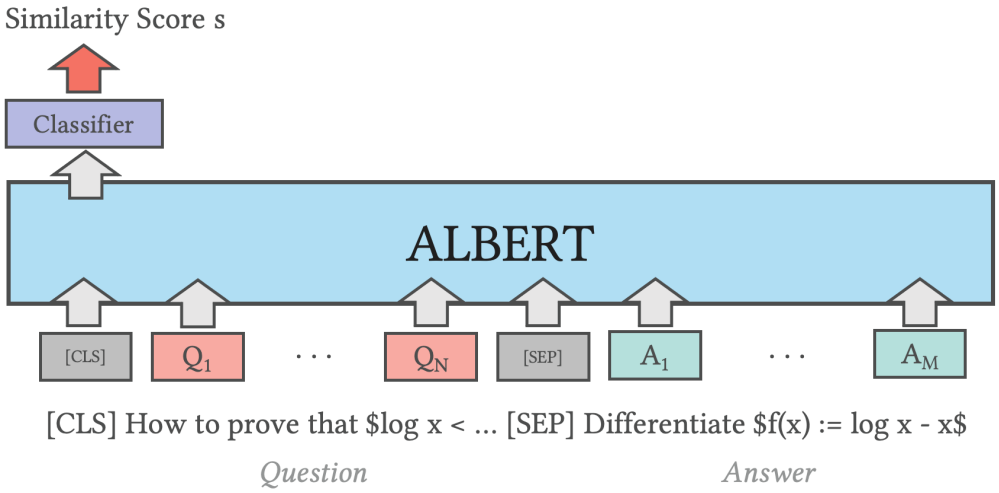

Figure 3: Architecture of Task 1's Fine-Tuning.

segments together exceeding the maximum length during e.g., NSP or fine-tuning, token by token was deleted from the longest sequence until the sum of the number of both segments equaled the maximum length.

5.2. ALBERT Model

In order to predict whether an answer $A = A_1 A_2 \cdots A_M$ is relevant to a question $Q = Q_1 Q_2 \cdots Q_N$ a classifier is trained on top of the pre-trained ALBERT model as depicted in Figure 3. The input string $\langle CLS \rangle Q_1 Q_2 \cdots Q_N \langle SEP \rangle A_1 A_2 \cdots A_M$, with $\langle CLS \rangle$ being the classification token and $\langle SEP \rangle$ the separation token, is presented to the model:

$$CU_1 U_2 \cdots U_N = \text{ALBERT}(E_{CLS} E_1 E_2 \cdots E_{N+M}),$$

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 Section 5.1. After the forward pass through the model, the output vector of the $\langle CLS \rangle$ token C is given into a classification layer:

$$p(label = i | Q, A) = \text{softmax}(C \cdot W_{Task1} + b_{Task1})_i,$$

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 similarity score s for the answer A to a question Q and is then used to rank the answers in the corpus:

$$s(Q, A) = p(label = 1 | Q, A).$$

Fine-Tuning Data

In order to fine-tune the ALBERT models for Task 1, we paired each question with one correct answer and one incorrect answer. The correct answer was 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. Following this procedure, we yielded 1.9M examples, of which 90% were used as training data for the fine-tuning task. We presented the model the entire text of the questions and answers using the structure introduced in the previous section.

5.3. ColBERT Model

Our second approach was to train ColBERT on top of a pre-trained BERT model. In each training step, the model is presented the query Q and two answers: one being a relevant answer A, the second being an answer B that should be regarded as non-relevant by the model. All three strings, Q, A and B are prepended with a token denoting the string as either question (query), $\langle Q \rangle$ or answer (document) $\langle D \rangle$, and are passed through the BERT model individually to create contextualized embeddings for each post:

$$C_Q Q U_1 U_2 \cdots U_N = \mathrm{BERT}(E_{CLS} E_Q E_1 E_2 \cdots E_N),$$

$$C_D D V_1 V_2 \cdots V_M = \mathrm{BERT}(E_{CLS} E_D F_1 F_2 \cdots F_M),$$

where E_i, F_i, E_{CLS}, E_Q, and E_D are the input embeddings for each input token, the $\langle CLS \rangle$ token, $\langle Q \rangle$ token and the $\langle D \rangle$ token, respectively, calculated as explained in Section 5.1. Using the late interaction mechanism as specified by Khattab et al. [7] a relevance or similarity score is calculated for each question-answer pair and optimized by applying softmax cross-entropy loss over the scores:

$$s(Q, A) = \sum_{i=1}^{N} \max_{j \in \{1,\ldots,M\}} U_i V_j^T.$$

More implementation specific detail can be found in work by Khattab et al. [7].

Fine-Tuning Data

We use the same procedure to generate training data for the ColBERT-based models, but with the difference that we used up to $N_{answers} = 10$ correct and incorrect answers in case a question had that many submitted answers. If less answers were present, the minimum of correct and incorrect answers was used such that the number of correct and incorrect answers matched. We paired each correct answer with all incorrect answers, generating at most $10 \times 10 = 100$ samples for each question. We experimented with $N_{answers} = 1$ and $N_{answers} = 5$, but we achieved best results with $N_{answers} = 10$.

5.4. Evaluation Data

During evaluation we exploited the tag information from the queries in order to rank only the answers that shared at least one tag with the query question. In this way, we saved large amounts of computation time for the ALBERT-based models. Each question and the answers were pre-processed and paired in the same way as during fine-tuning.

Table 1
Overview of Pre-Training Configurations of ALBERT models

Model	Pre-Training Data	Steps
Base 750k	(1) sentence split	750k
Base 250k	(1) sentence split	250k
Base Combined	(1)+(2) combined	135k
Scratch 1M	(1) sentence split	1M
Scratch 2M	(1) sentence split	2M
Scratch Separated	(2) separated	1M

For ColBERT, we generated an index based on all answers whose question had at least one tag that was associated with at least one query question.

For each query the organizers of the Lab annotated whether answering the question mostly depends on its text, its formulas or both. We used these categories for the interpretation of our results.

5.5. Experiments

We tested various scenarios for training ALBERT of which we report six in this work: The models Base 750k, Base Combined and Base 250k are initialized from the official weights of the ALBERT base model released by the ALBERT authors and were further pre-trained on ARQMath data using strategy (1), i.e., sentence split text (see Section 5.1). The data pre-processed with strategy (2), i.e., data split into text and LaTeX, was mixed with the aforementioned data to pre-train Base Combined. The weights of Scratch 1M, Scratch Separated and Scratch 2M were initialized randomly. Scratch 1M and Scratch 2M used the sentence split data (1) while Scratch Separated was only pre-trained on the separated data of strategy (2). All six models followed the recommendations for hyperparameter configuration during pre-training, with 12M parameters, using the LAMP optimizer [25], 3,125 warm up steps, maximum sequence length of 512 and a vocabulary size of 30,000. Furthermore, we used a batch size of 32 and a learning rate of 0.0005. The models were trained for different numbers of steps: Base 750k was trained for 750k steps while the training of Base 250k was already stopped after 250k steps. Scratch 1M and Scratch Separated pre-trained for 1M steps. This amount was doubled for Scratch 2M. Finally, Base Combined could only be trained for 135k steps before the final submission of the results. A summary of the different combination of pre-training data and number of steps for each model can be found in Table 1. After pre-training, each classification model was fine-tuned for 125k steps using a batch size of 32, a learning rate of 2e-5 and 200 warm-up steps. Both pre-training and fine-tuning was performed using the code published in the official ALBERT repository. We submitted results of four ALBERT-based models to the ARQMath 2021 Lab and evaluated Base 250k and Scratch 2M using the official evaluation tools.

ColBERT can be seen as an extension of BERT whose performance depends on its pre-training [7]. Therefore, we apply three differently pre-trained models as the basis for ColBERT: ColBERT uses the weights of the original BERT-base, ColSciBERT uses SciBERT [15], which was trained on a large corpus of scientific publications from multiple domains and finally, we pre-trained

our own BERT model for ColARQBERT. The last model was initialized using the original BERT weights and was then further pre-trained on the sentence split data (1) described earlier. The hyperparameters recommended by the BERT authors in their repository were used to pre-train this model: The learning rate was set to 2e-05, one batch contained 16 samples and the models were trained for 500k steps. In contrast to the recommendations we set the maximum length of the input to 512, because we did not start to train the model from scratch, where the initial sequence length was set to 128, but rather further trained the already fully pre-trained model on additional data. The training of all three ColBERT models made use of the same hyperparameter configuration. We optimized the L2 similarity between 128-dimensional vectors with a batch size of 128 for 75k steps. Other parameters kept their default values. Punctuation tokens were masked, but we also experimented with models that did not mask them, but we could not see a significant difference in the results. We also started to incorporate ALBERT as a base model for ColBERT, but did not yet find a configuration for a successful training. The pre-training of ColARQBERT was performed using the code published by the BERT authors, while the ColBERT repository was slightly adapted to support different checkpoints than BERT base in order to train the other models. Finally, ColSciBERT model was submitted to the competition, while ColBERT and ColARQBERT were evaluated later using the official evaluation guide.

5.6. Evaluation

The results of our ALBERT and ColBERT-based models are shown in Table 2 together with additional experiments that were not submitted and results of other models from the ARQMath 2021 Lab for comparison. We report the scores of the 2020 test set and the new 2021 test set. In addition, we break down the nDCG' score results of 2020 by the categories on which part answering the question depends. These categories are either text, formula or both in combination and were annotated by the organizers of the lab. The scores for each category are reported in Table 3.

Pre-Training Adjustments

In general, our results can be seen as competitive. Regarding nDCG', all ALBERT-based models could outperform the baseline systems in both years. On the 2020's test set, one team with three systems received the highest scores for mAP', while our ALBERT-based models are all in the range of the top four teams. In 2021, our best model ranks second among all teams regarding mAP'. Our results for p'@10 are not as high as the best baseline, but there was not a single system from any of the teams that could beat the baseline results for p'@10. Comparing to the other participants, our system receives the highest score for p'@10 in 2021.

The reason why our Precision is relatively high, but the nDCG' is lower compared to the other teams that received higher scores could be that our systems do not rank all answers for each topic due to the too time consuming evaluation. Possibly, our results would have been better if all answers would have been scored for their relevance.

We will now take a deeper look at the differences between the models we trained. When comparing Base 750k and Base 250k, the overall score is slightly increased by the longer pre-training. In Table 3 we see that with longer pre-training the model learned a better understanding

Table 2
Results of Task 1

Model	Official Identifier	2020 nDCG'	2020 mAP'	2020 p'@10	2021 nDCG'	2021 mAP'	2021 p'@10
Base 750k	TU_DBS_P	0.380	0.198	0.316	0.377	0.158	**0.227**
Scratch 1M	TU_DBS_A1	0.362	0.178	0.304	0.353	0.132	0.180
Base Combined	TU_DBS_A3	0.359	0.173	0.299	0.357	0.141	0.194
Scratch Separated	TU_DBS_A2	0.356	0.173	0.291	0.367	0.147	0.217
ColSciBERT	TU_DBS_A4	0.045	0.016	0.071	0.028	0.004	0.009
Additional Experiments							
Base 250k	-	0.375	0.193	0.311			
Scratch 2M	-	0.359	0.177	0.297			
ColARQBERT	-	0.225	0.073	0.131			
ColBERT	-	0.183	0.053	0.110			
ARQMath Competitors							
Best '20&'21	MathDowsers-primary	**0.433**	0.191	0.249	**0.434**	**0.169**	0.211
Best '20	DPRL-RRF	0.422	**0.247**	**0.386**	0.347	0.101	0.132
Best Baseline	linked_results	0.279	0.194	**0.386**	0.203	0.120	**0.282**

Table 3
nDCG' Scores of Task 1 by Category, 2020 Test Set

	Base 750k	Scratch 1M	Base Combined	Scratch Separated	Base 250k
Both	0.365	0.365	0.364	0.321	**0.370**
Formula	**0.382**	0.354	0.338	0.367	0.366
Text	0.411	0.375	0.399	**0.421**	0.408

of text and formulas on their own, but for category 'both' the results decreased. On the other hand, pre-training for too many steps shows effects of over-fitting as the scores start to decrease again as we see in the difference between Scratch 1M and Scratch 2M.

The comparison of Scratch 1M and Scratch Separated shows that the separation of text and mathematical formulas leads to better nDCG' scores for queries dependent on formulas and text separately, but the performance degrades on question-answers pairs that depend on both, which is expected since the model was not pre-trained on data that involved both in one example. Base Combined has a much lower nDCG' value for the formula category in comparison to the other models. This can be explained by the fact that it was pre-trained for a much lower number of steps. The same effect is visible when viewing Base 750k and Base 250k. Therefore, we hypothesize that a pre-training of 750k or even more steps could even outperform Base 750k and Scratch Separated in all three categories.

BERT-base models generally benefit from a long pre-training on a large corpus. In our experiments, we could not observe this behavior. We experimented with models trained for 2M steps on data from 41 StackExchange communities supporting LaTeX, but the results are worse than the ones presented in Table 2.

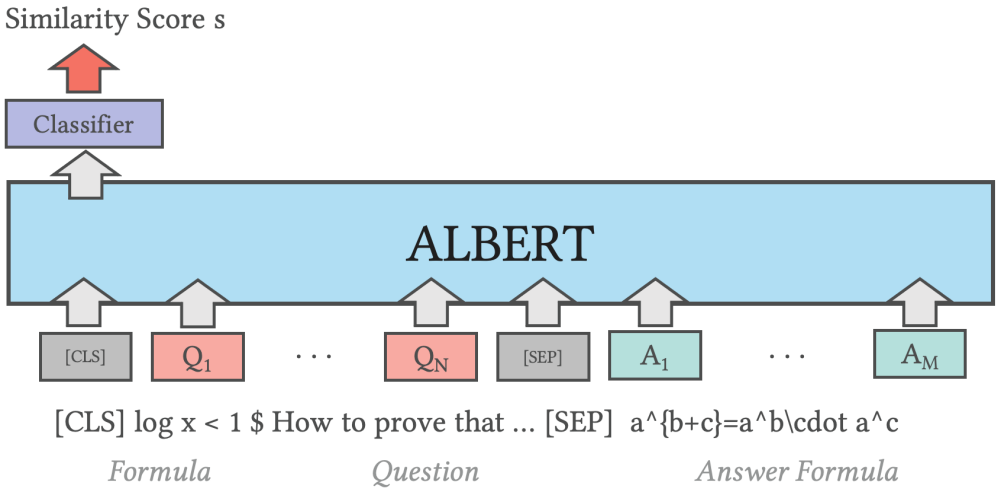

Figure 4: Architecture of Task 2's Fine-Tuning.

ColBERT

ColSciBERT is the fifth model we submitted for the 2020 ARQMath Task 1 and it was trained using ColBERT. As can be seen from the results table, its performance is not optimal hinting at a substantial problem during training or evaluation. This could be caused by using SciBERT as the basis for ColBERT. Two other models that were not officially submitted to the Lab received higher scores, but are still not on par with our other ALBERT-based approaches regarding all three metrics. This confirms the hypothesis that SciBERT is not suitable in this scenario.

Nevertheless, with ColBERT the time required to evaluate all 100 topics of 2020 took around six minutes using two NVIDIA GTX 1080 while evaluating one query using our ALBERT-based classification approach took between ten minutes and one hour on one NVIDIA V100. Therefore, further research in this direction is worthwhile for speeding up the evaluation while receiving competitive scores at the same time. Future work here should further analyze the performance and determine the best training scenario for a ColBERT-based system.

6. Task 2

Task 2 deals with the retrieval of relevant formulas given a query formula together with the post in which it appeared. As for Task 1 we will start from a pre-trained ALBERT model, which was already introduced in Section 5.1. In the following section, we will therefore only highlight how the fine-tuning and data processing was performed and present the results of the application of an ALBERT-based model for the task of formula similarity search.

6.1. Fine-Tuning Model

For formula similarity search our approach slightly differs from the one presented in Section 5.2 for Task 1. Instead of presenting the model the two formulas as the query and answer to classify

whether they are relevant to each other, we add the question in which the first formula appears as additional context since the same formula and especially its variables can have different interpretations depending on its context, e.g., $P(X)$ can be a probability of a random variable or a polynomial depending on its context. Each query formula was concatenated with its question forming the first part of the input $Q_1 Q_2 \cdots Q_N$, separated by \$. The formula that should be assessed for its similarity to the first formula makes up the second part of the input $A_1 A_2 \cdots A_M$. Analogously to Task 1, the classification token $\langle CLS \rangle$ is added at the beginning of the input and both parts are separated by the separation token $\langle SEP \rangle$. The output of the classifier is the similarity score that is optimized during training and used for the ranking of candidates during evaluation. The process of fine-tuning ALBERT for Task 2 is depicted in Figure 4.

Fine-Tuning Data

Fine-tuning was performed on formulas in context with the post in which they appeared in order to provide the model with information on how the formula was used by the author. First, we removed formulas that contained less than three LaTeX token and filtered the ARQMath corpus for posts that had at least one formula remaining. For each post in the corpus one formula was chosen either by chance (denoted as random) or the longest formula was used as the query formula (denoted as longest). Formulas from the title of the post were preferred when the title contained any formulas, because the title can be seen as a summary of the post that should include the formula with the most meaning to the post. We chose this procedure because we faced the problem that the posts contained many formulas (on average 9.41 formulas per post) not all of which were directly relevant to the post or the given answers, such as, definitions of variables or examples.

For each query formula we determined positive and negative examples. The positive examples, i.e., the ones that should be classified as relevant formulas by the model, originate in the answers given a post. This assumes that the formulas in the answers are relevant to the formulas in its question post. We chose the negative examples from answer posts where their questions had at least one common tag with the query post. For each query we used a maximum of five positive and negative examples, where the number of positive examples and negative examples were equal. Hence, when a question had $N_{formulas}$ with $N_{formulas} < 5$ formulas in its answers or we found only $N_{formulas}$, $N_{formulas} < 5$ formulas in other posts with the same tags, then only $N_{formulas}$ formulas for positive and negative examples would be used. In total, we yielded 5,812,412 question-answer formula pairs of which 90% were used as training data. The training data was presented to the model in the way that was described above.

6.2. Evaluation Data

Due to time and hardware constraints, it was not possible to evaluate our models on the entire collection of formulas. Therefore, we limited the corpus to visually distinct formulas with more than three LaTeX tokens which appeared in questions and answers that shared either one or all tags with the query post, denoted by 'one' or 'all' in the results table, respectively. The ARQMath collection provided each formula with its visual id and the post id in which it appeared. We

Table 4
Results of Task 2

Fine-Tuning Data	Eval Tags	Official Identifier	2020 nDCG'	2020 mAP'	2020 p'@10	2021 nDCG'	2021 mAP'	2021 p'@10
random	one	TU_DBS_A3	0.426	0.298	0.386	-	-	-
longest	one	TU_DBS_A1	0.396	0.271	0.391	-	-	-
random	all	TU_DBS_A2	0.157	0.085	0.122	0.154	0.071	0.217
longest	all	TU_DBS_P	0.152	0.080	0.122	0.153	0.069	0.216
Best 2020		DPRL-ltrall	**0.738**	**0.525**	**0.542**	0.445	0.216	0.333
Best 2021		Approach0-P300	0.507	0.342	0.441	**0.555**	**0.361**	**0.488**
Baseline		TangentS_Res	0.691	0.446	0.453	0.492	0.272	0.419

determined the tags of the posts from the post's corpus and aggregated for each visual id the tags of all posts in which formulas with this visual id appeared. For each visual id that remained in the corpus for a given query we provided the model with the query formula, its posts and the first formula that was associated with this visual id. The post id that was reported as our Task 2 result was the post id corresponding to the formula in the model input, i.e., the post id of the first formula for each visual id.

6.3. Experiments

In total, we trained two classifiers on different fine-tuning data and evaluated each of them in two settings as described above. These four configurations can be found in Table 4. Both models are based on the pre-trained ALBERT-base model that was used for the Task 1 Model Base 750k. The fine-tuning hyperparameters are the same as for the models of Task 1: we used a learning rate of 2e-5, batch size of 32 and 200 steps for warm-up. Both classifiers were trained for 125k steps.

6.4. Evaluation

The results of our experiments for Task 2 can be seen in Table 4. Our best model on the 2020 topics is fine-tuned on the random formulas as queries. It is evaluated on all distinct question and answer formulas that shared at least one tag with the query post. In general, the two models trained on data with a random query formula showed better results than the two models always using the longest formula. Including all formulas that had at least one similar tag increases the search space and therefore the results for the retrieved formulas from this search space are better. This suggests that the performance could be even more increased if all formulas from the entire corpus were used for the classification. This was not done in this work due to our hardware constraints leading to long evaluation times.

Generally speaking, our ALBERT-based model is a promising approach, but the comparison to other participants of the lab demonstrates that it is not on par with state-of-the-art models or the baseline system. Hence, future work should explore better methods of representing formulas using ALBERT. In this work, only LaTeX-based representations have been used, but ARQMath also

provides tree-based data for each formula. One possible improvement could be the prediction of relationships between nodes in these trees as it was done in a similar way for Programming Language Understanding using data flow graphs [26]. Furthermore, as seen in the evaluation of Task 1 in Section 5.6, pre-training on data that separated text and formulas improved the scores on formula dependent questions pointing to a better understanding of mathematical formulas compared to models trained on non-split data. Therefore, these pre-trained models could also be beneficial as basis for Task 2 fine-tuning.

7. Conclusion

Mathematical Information Retrieval deals with the retrieval of documents from a corpus, which are relevant to a query, where documents and queries may include both, natural language and mathematical formulas. Two instances for such an objective are Task 1 and Task 2 of the ARQMath Lab, whose goal is to either retrieve answers given a question or formula retrieval using a formula in its context. Since this challenge includes not only text written in English, but also formulas, approaches from Natural Language Processing and Information Retrieval have to be adapted in order to be able to interpret also the semantics of mathematical formulas. This has also been demonstrated in our previous work, where we showed that ALBERT has to be further pre-trained on relevant data in order to better handle formulas in MIR tasks. In this work we further analyzed this claim and showed that our previous results for Task 1 could even be improved by a longer pre-training on the data provided by ARQMath. Furthermore, we showed that separating large chunks of natural language text and LaTeX notation in one sentence increased the model's performance on formula-only and text-only dependent questions, respectively. The second contribution of this work was to explore the application of ColBERT in order to accelerate the evaluation of queries, because our classification-based approach is too time-consuming. Thereby, we trained and evaluated a ColBERT model and showed that further improvements are necessary before this approach can reach state-of-the-art performance. We also applied our ALBERT-based approach to the formula retrieval objective of Task 2 and showed that there is still work necessary in order to improve the model's understanding of formula similarity. In conclusion, we showed promising approaches for Mathematical Answer Retrieval and Formula Similarity Search by applying differently pre-trained and fine-tuned ALBERT models and one ColBERT model. In order to improve the modeling capabilities of mathematical formulas, we recommended strategies involving several pre-training methods that include syntactical features of formula that we have not yet taken into account. To facilitate research based on our work, we release the code for data pre-processing and the training of the models in the project's repository[3]. The source code for training the ColBERT-based models was forked from the official ColBERT repository and slightly adjusted[4].

[3] https://github.com/AnReu/ALBERT-for-Math-AR
[4] https://github.com/AnReu/ColBERT-for-Formulas

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] B. Mansouri, A. Agarwal, D. Oard, R. Zanibbi, Finding old answers to new math questions: the arqmath lab at clef 2020, in: European Conference on Information Retrieval, Springer, 2020, pp. 564–571.

[2] B. Mansouri, A. Agarwal, D. Oard, R. Zanibbi, Advancing math-aware search: The arqmath-2 lab at clef 2021 (2021) 631–638.

[3] 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).

[4] R. Nogueira, K. Cho, Passage re-ranking with bert, arXiv preprint arXiv:1901.04085 (2019).

[5] A. Reusch, M. Thiele, W. Lehner, An albert-based similarity measure for mathematical answer retrieval, in: Proceedings of the 44rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, to appear.

[6] S. MacAvaney, A. Yates, A. Cohan, N. Goharian, Cedr: Contextualized embeddings for document ranking, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 1101–1104.

[7] O. Khattab, M. Zaharia, Colbert: Efficient and effective passage search via contextualized late interaction over bert, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 39–48.

[8] R. Zanibbi, A. Aizawa, M. Kohlhase, I. Ounis, G. Topic, K. Davila, Ntcir-12 mathir task overview., in: NTCIR, 2016.

[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] 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).

[11] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, N. A. Smith, Don't stop pretraining: Adapt language models to domains and tasks, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 8342–8360.

[12] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, J. Kang, Biobert: a pre-trained biomedical language representation model for biomedical text mining, Bioinformatics 36 (2020) 1234–1240.

[13] 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.
[14] K. Huang, J. Altosaar, R. Ranganath, Clinicalbert: Modeling clinical notes and predicting hospital readmission, arXiv preprint arXiv:1904.05342 (2019).
[15] 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.
[16] 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.
[17] 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).
[18] S. Peng, K. Yuan, L. Gao, Z. Tang, Mathbert: A pre-trained model for mathematical formula understanding, arXiv preprint arXiv:2105.00377 (2021).
[19] 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.
[20] V. Novotnỳ, P. Sojka, M. Štefánik, D. Lupták, Three is better than one, in: CEUR Workshop Proceedings. Thessaloniki, Greece, 2020.
[21] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, L. Deng, Ms marco: A human generated machine reading comprehension dataset, in: CoCo@ NIPS, 2016.
[22] 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.
[23] T. Kudo, J. Richardson, Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing, arXiv preprint arXiv:1808.06226 (2018).
[24] R. Sennrich, B. Haddow, A. Birch, Neural machine translation of rare words with subword units, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Berlin, Germany, 2016, pp. 1715–1725. URL: https://www.aclweb.org/anthology/P16-1162. doi:10.18653/v1/P16-1162.
[25] Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, C.-J. Hsieh, Large batch optimization for deep learning: Training bert in 76 minutes, in: International Conference on Learning Representations, 2019.
[26] D. Guo, S. Ren, S. Lu, Z. Feng, D. Tang, S. LIU, L. Zhou, N. Duan, A. Svyatkovskiy, S. Fu, M. Tufano, S. K. Deng, C. Clement, D. Drain, N. Sundaresan, J. Yin, D. Jiang, M. Zhou, Graphcode{bert}: Pre-training code representations with data flow, in: International Conference on Learning Representations, 2021. URL: https://openreview.net/forum?id=jLoC4ez43PZ.