Investigating the Usage of Formulae in Mathematical Answer Retrieval

Anja Reusch[0000−0002−2537−9841], Julius Gonsior[0000−0002−5985−4348], Claudio Hartmann[0000−0002−5334−059X] and Wolfgang Lehner[0000−0001−8107−2775]

Dresden Database Research Group, Technische Universtität Dresden, Germany
{firstname.lastname}@tu-dresden.de

Abstract. This work focuses on the task of Mathematical Answer Retrieval and studies the factors a recent Transformer-Encoder-based Language Model (LM) uses to assess the relevance of an answer for a given mathematical question. Mainly, we investigate three factors: (1) the general influence of mathematical formulae, (2) the usage of structural information of those formulae, (3) the overlap of variable names in answers and questions. The findings of the investigation indicate that the LM for Mathematical Answer Retrieval mainly relies on shallow features such as the overlap of variables between question and answers. Furthermore, we identified a malicious shortcut in the training data that hinders the usage of structural information and by removing this shortcut improved the overall accuracy. We want to foster future research on how LMs are trained for Mathematical Answer Retrieval and provide a basic evaluation set up[1] for existing models.

Keywords: Mathematical Information Retrieval · Transformer-Encoders.

1 Motivation

Mathematical Answer Retrieval (Math AR) deals with the task of ranking a set of answers for their relevance to a given mathematical question. As in general Information Retrieval, Transformer-Encoder-based Language Models (LMs) are part of the most successful approaches for Math AR [7, 21, 32], but are usually applied as black box models. Recently, research has found that LMs adapted to mathematics encode mathematical parse trees, namely Operator Trees, in their contextualized embeddings [20]. However, when the model is probed for these tree structures after it was fine-tuned on Math AR, the performance degrades. In Fig. 1, we visualize how much structural information can be extracted from the embeddings of each of the Transformer layers following the methodology of the original authors. After the model was fine-tuned for Math AR (blue line), the performance on the structural probing task was significantly lower in all layers greater than 2. This finding demonstrates that the probe was less successful in extracting structural information from the model's embeddings after fine-tuning,

[1] Link to repository: https://github.com/AnReu/math_analysis

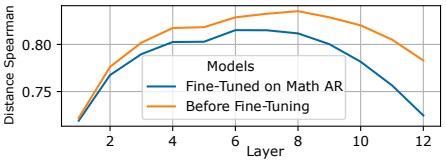

Fig. 1: Layerwise results of an LM before and after fine-tuning on Math AR.

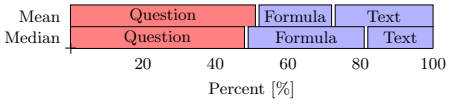

Fig. 2: IG scores aggregated on question tokens (red), and answer tokens (blue).

which indicates that the structural information got lost during the fine-tuning process, since the model "overwrote" it with other information about formulae. Van Aken et al. [27] showed in a similar experiment that factual knowledge that is not required for solving a fine-tuning task is lost after fine-tuning, while relevant information is more extractable by their probe. Therefore, our question here is which information on mathematical formulae instead got reinforced by the model during fine-tuning on Math AR. Hence, the goal of this work is to study how the LM that demonstrated the best ability for extracting structural information of mathematical formulae, `AnReu/math_pretrained_bert`, is using formulae when assessing the relevance of a mathematical answer. These insights are then used to identify and verify shortcomings in the training and optimize the model's retrieval performance.

Before we begin, we motivate our methodology by performing an initial experiment where we investigate whether the model considers formula token at all during the retrieval process. Hereby, we utilize the Integrated Gradient (IG) attribution method [25], which measures how much of the model's prediction can be attributed to which token. We use `AnReu/math_pretrained_bert` after it was fine-tuned in a Cross-Encoder setup for Math AR using the ARQMath data set [11] following the methodology of Reusch et al. [21]. Since cross-encoders simply perform a classification on top of the [CLS] embedding of the LM, IG calculates the importance of each token with regard to the class denoting that the answer is relevant for the mathematical question. Similarly to prior work [31], we break up the scores by query (here: question) and document (answer), but further differentiate between token types of the answer to study whether the respective formula parts are used for the relevance assessment. Transformer-interpret's implementation[2] of IG is applied and we aggregated (mean and median) the top 50 answers for each topic for tokens that belong to the mathematical question, the text part of the answer, and the mathematical tokens of the answer. The results can be found in Fig. 2 where the proportion of IG scores for each part of the input is plotted. It is visible that the model's decision for the relevance score is in part based on the answer and in part also based on the question, which is as expected. The most important observation can be made when looking at the IG scores on the formula tokens of the answer. Here, we can see that the model indeed considers the formulae along with the textual part of the answer. Thus,

[2] https://pypi.org/project/transformers-interpret/

we can conclude that the model uses information from formulae when assessing the relevance of an answer, but it is still an open question how the formulae are used and to which extent the model is relying on this information.
Therefore, this work begins by further analyzing what influence formulae in the answer have when the model assesses the relevance of an answer. Here, we investigate how much the retrieval performance degrades when the model has no or limited access to formula information in the answers. Afterwards, we compare the ability of the LM before and after fine-tuning on Math AR on the task of predicting variables that occur in the token overlap of a question and an answer. This task is motivated by the observation that several results, which had the same structure as the formula in the question but no common variables, were ranked low by the model, even though they were considered relevant by experts. In contrast, non-relevant results, which shared the same variable names with a different structure, were ranked high by the model. Hence, the model seems to be biased towards answers that have a higher number of common variables with the question. During training, it could have learned a malicious shortcut of only considering the variable overlap and not the actual semantics of a formula. To verify the insights from the first experiments described here, we construct two additional training sets for fine-tuning. The first data set artificially adds the mentioned shortcut, while the second completely removes it such that the model has to focus on other features than the shortcut. By comparing the results of the model on these two fine-tuning sets to the baseline data set, we can conclude that the model indeed relied on this shortcut.
To summarize, this paper aims to contribute to the following: Our evaluation shows that the model `AnReu/math_pretrained_bert` uses information from the formulae when assessing the relevance of an answer given a question (Sec. 3), but probing revealed that during fine-tuning, the importance of overlapping variables between question and answers is reinforced (Sec. 4). We further demonstrate that substituting variables in the fine-tuning set leads to an overall improved performance in the Math AR task (Sec. 5).

2 Preliminaries

2.1 Related Work

To interpret the knowledge and skills Transformer-Encoder models learn during training, several *post-hoc* techniques have been introduced. One of these techniques is the probing classifier or the probe where a simple classifier is trained in order to predict a certain property from the model's learned contextualized embeddings. If the classifier successfully predicts the property, sufficient information about this property has to be encoded in the contextualized embeddings. Therefore, one concludes that the model encoded or learned this property during the initial training. For more details on probing, we refer the reader to this survey [2]. The application of these probing classifiers for knowledge of retrieval models was studied in previous research: Fan et al. [6] evaluated 16 different tasks for natural language understanding while Van Aken et al. [27] probed for

layer-wise information in question answering. Also, Zhan et al. [31] applied a probing classifier for a model's ranking performance on different layers. Wallat et al. [29] studied which information can be extracted by a probing classifier designed for factual knowledge before and after it was fine-tuned on a fine-tuning task and found that the fine-tuning task forces the model to forget and reinforce certain information.
In the context of mathematical models, [20] trained a structural probe to evaluate how much information on mathematical parse trees can be extracted from the contextualized embeddings of math-adapted Transformer-Encoder models. While several studies cover new data sets to inspect how well Transformer models solve mathematical questions [8, 24, 5, 18], no research was specifically conducted to analyze the layer-wise information of Transformer-Encoder models for Math AR. In a similar way, previous research analyzed the IG attribution and the attention patterns of BERT models for document ranking [31, 19]. However, since the interpretability of attention is disputed [28], we refrain from using it. The removal and scrambling of words have been used to study their effects on LMs [16, 17]. However, we apply these ideas to formulae in a mathematical context.

2.2 Fine-tuning Setup

Model Choice The recent series of ARQMath Labs [11–13] has moved more attention to the development of Transformer-Encoder models that are adapted for Math AR. During the course of the lab series, several teams adapted exisiting Transformer-Encoder models that were originally pre-trained on natural language, to the domain of mathematical documents [23, 22, 21, 15]. These models use LaTeX as the form of representing formulae. In contrast, other works make use of (linearized) tree structures of mathematical parse trees to represent formulae in their models' input [32, 14]. Even though several Transformer-Encoder models for mathematical documents have been proposed, this work analyzes the model `AnReu/math_pretrained_bert`. We chose this model because the goal of this work is to study which mathematical information is used when a model assesses the relevance of an answer given a question, and this particular model displayed the best results when extracting syntactic information from its contextualized embeddings [20]. In addition, this is also the BERT-based version of the successful `AnReu/math_albert`, which was the best performing cross-encoder model in the ARQMath Lab 3 [21].

Training Details In order to perform Math AR, we train the models in a cross-encoder setup [9], where a binary classification is used to assess if a given answer addresses a mathematical question. Question and answer tokens are concatenated and provided as an input to the LM up to a limit of 512 tokens. A classification head (linear classifier) on top of the [SEP] token embedding is trained to decide for either "relevant answer" or "non-relevant answer". The question-answer pairs we use are based on the official ARQMath 2020 corpus [11]. Relevant answers are answers that a user posted in response to a mathematical question. For each question, we use up to ten relevant answers or as many as the questions

had if less were present. Non-relevant answers are sampled by chance from the set of answers with at least one similar topic. Both classes, "relevant" and "non-relevant", are equally distributed among the 2.7 M pairs, of which 90% are used for training and 10% for validation. The training and evaluation was performed using Huggingface's transformers library [30]. We followed Reusch et al. [21] for pre-processing and hyperparameter settings, because this set up demonstrated the most promising results in a cross-encoder set up.
The evaluation of our models is performed in the same way as for the ARQMath Lab that provides 77 questions from 2019 with relevance judgment for several answers annotated using pooling. Each of these questions is paired with every answer from the corpus, then each pair is provided as input to the model. The classification score for label "relevant answer" is used to rank the answers. We follow ARQMath and evaluate the top 1,000 ranking answers for each topic using nDCG and the top 10 using precision. For each question from the evaluation corpus, the organizers of the ARQMath Lab 2020 also divided the questions into the three categories *Formulae*, *Text*, and *Both* indicating whether answering a question depended on understanding mainly the question's formulae, the natural language text or both, respectively. During evaluation we will break down the performance of our models by these categories.

Judged Evaluation The ARQMath Lab does not provide relevance judgments for every answer regarding each question, but only for a small subset that was annotated during the pooling procedure. Hence, there exists a large number of answers that do not contain scores. When evaluating the performance of a model, non-judged answers are removed and the nDCG and Precision are calculated only on basis of the judged answers. However, evaluating the models on the entire set of answers is costly. Therefore, in most cases evaluations on the Math AR test set were performed only using judged answers meaning that for each question in the test set only answers whose relevance was judged during the ARQMath Lab were considered as candidates. When only comparing the approaches and models in this paper, this is a valid approach since only using the judged answers does not influence the ranking of the answers nor the relations between the scores and the models. But we cannot compare these model results to the results of the original ARQMath Lab because their ranking contains non-judged results which were removed but still influence the ranks of relevant answers when computing the metrics. Therefore, we provide an evaluation of the final models on the non-judged test set using the original metrics as used in the ARQMath Lab (nDCG$'$ and p$'$@10) at the end of the work.

Significance Testing In order to compare models, we use the Almost Stochastic Order (ASO) test [3, 4] as implemented by Ulmer et al. [26]. We compared all pairs of models based on five random seeds each using ASO with a confidence level of $\alpha = 0.05$ (before adjusting for all pair-wise comparisons using the Bonferroni correction). We report almost stochastic dominance ($\epsilon_{min} < \tau$ with $\tau = 0.2$) in all results sections.

3 Usage of Mathematical Formulae for Answer Retrieval

First of all, we study if models fine-tuned for Mathematical Answer Retrieval incorporate mathematical information at all, when determining the relevance of an answer given a question. We start with three simple experiments where we evaluate the results on the ARQMath 2020 test set when removing all mathematical formulae (1) or when replacing each formula with a dummy expression (2). Thereby, we identify if the models use the formulae at all when differentiating between answers. If the results on the modified two test sets are lower than on the original test set, the models relies on the formula information to judge the relevance. If the scores are higher than on the original test set, the formula information might confuse the model or lead it in the wrong direction. Additionally, each formula in the test set is replaced by a string containing the same tokens, but sorted (3), evaluating the usage of structural information. Thus, we end up with three variants (1) - (3) of the test set, each enabling us to show a specific aspect of the trained knowledge.

3.1 Experiment Setup

In these three experiments, we use the models trained by applying our baseline setup as detailed in Sec. 2.2 and evaluate them in the same way. The only difference compared to experiment (1), (2), and (3) is that we evaluate them on a slightly modified test set as we will detail in the following section.
Each test set contains the same questions from the ARQMath 2020 test set along with the same set of answers. Each question is paired with each answer from the set of judged answers, which are provided by the organizer of the ARQMath Lab. The unprocessed data is in XML format while each post is formatted in HTML. The formulae are present in ⟨math⟩ containers and therefore easily removable. For test set (1), we remove each mathematical formula entirely. Since this could cause a break in the coherence of a post, we create test set (2) where each formula is replaced by $a+b$. Because each formula contains now the same information, the model cannot differentiate between two answers only on basis of these formulae. For test set (3), we tokenized each formula using a LaTeX tokenizer and sorted the tokens. This way structural information can not be identified by the models. An illustration of the applied test sets can be seen in Tab. 1.

3.2 Results

The results of the experiments on the three modified test sets as well as for the default test set are displayed in Tab. 1. There is a visible and significant drop on both metrics when comparing the default test set with each of the modified test sets. The differences between test sets (1) and (2) are not significant. In addition, we break the score down regarding the three dependency categories *Formulae*, *Text* and *Both* (not in the table) for a more detailed analysis: The largest drop in performance is as one would expect in the *Formulae* category (on average -0.23 on nDCG@1,000 on *Formulae*, -0.11 on *Both* and -0.04 in *Text*). This indicates

Table 1: Results for all applied test sets including an example.

Test Set	Example	nDCG@1,000	p@10
Default	How to simplify $\int_a^b f(x)dx$+...	0.6990	0.3733
No Math (1)	How to simplify	0.5438	0.1349
Dummy Math (2)	How to simplify $a + b$	0.5309	0.1434
Sorted formulae (3)	How to simplify $?(()) + \int ab$...	0.5917	0.2221

that the model focuses more on mathematical formulae for questions where these formulae are also more important to retrieve a relevant answer.

From these results, it can be inferred that formula information is used by the models to determine the relevance of an answer since on both test sets the metrics deteriorate in comparison to the default test set that contained formulae. Because the formula information was not available in the two modified test sets, the models had to rely on the textual information, which in some cases was not sufficient to rate the answer accordingly. Since we now know that formulae are actually beneficial to answer a question and are not random by-standing tokens, we can investigate how they are used by the models.

Here, we can look at the scores for test set (3), which are between the ones of the default test set and test set (1) and (2). This indicates that the model does - in contrast to what Fig. 1 suggests - use some information on the structure of formulae as the scores drop when the structure is removed. However, after sorting the tokens of the formulae some information is still recovered and used to assess the relevance of the answers given the questions. This can be inferred because the model is able to better rank the test set (3) in contrast to test set (1) and (2). A possible reason could be that the sorting destroys also simple relationships such as operator-argument relationships, which could be used by a model to identify common patterns such as $f(x)$ or $sin(\pi)$. Here, further investigations are necessary to evaluate cases where the model uses structural information. We have seen that some structural information is used, but the results suggest that the model is also relying on other information, which is visible by the gap in the scores between test set (3) and (2). Therefore, the next section will evaluate another candidate for information the models might rely on: Variable Overlap.

4 Variable Overlap Prediction

The observation that question and answers, which are retrieved by the model, share more common tokens than relevant answers from the test set in general led us to investigate the variable overlap in answers that were ranked high by the model in comparison to the overlap of variables in actual relevant answers. The overlap of variables in the top 1,000 answers for ARQMath 2020 ranked by `AnReu/math_pretrained_bert` was 50.0% while when only considering relevant answers it is only 34.5%. The models seem to be biased towards answers that

have a higher number of common variables with the question which it could have learned during the fine-tuning. Hence, it is reasonable to investigate the model's ability of capturing which variables occur in both the question and the answer. We train a probing classifier to detect common tokens between the two input segments of the models and call this task *Variable Overlap Prediction*. As Sec. 1, we evaluate the model's performance before and after fine-tuning and compare them to see if it learned to reinforce this ability or replace it by something else. In the following sections, we will provide details on the data set and the training of the probing classifier and analyze the results of our experiment.

4.1 Training Details

Our data set comprises pairs of formulae as input data and the labels indicating the variables they have in common. The formulae are taken from the train and test split of the MATH data set [8]. We randomly paired two formulae and intersected their tokens. These intersection was then filtered for variables. We chose the set of single letters in the Latin alphabet as variables, since Greek letters or other variable names occurred only in less than 20 examples. Each variable denotes a single class. If the variable is included in the overlap of the two formulae, its class gets the label 1; if it is not present in the overlap, the label for the class is set to 0. If the overlap of the two formulae contains another token except of the variables, it was added as an instance of the fallback class "other". This class is needed when no variable overlap between the formulae is present, but other tokens still overlap, e.g. $a + 1$ and $b + 1$. Therefore, each sample from the data set consists of two formulae in LaTeX and one Boolean vector with 27 dimensions denoting the labels of the 26 variables and the fallback class. The training examples were selected from the train split of the MATH data set, while the examples in the test set were selected from its test split and deduplicated since formulae can occur in both splits. In total, the data set consists of 5k formula pairs with their labels for training, 2k for validation and 10k for the test set. As before, training and evaluation were performed using the transformers library [30]. We conducted hyperparameter tuning using Optuna [1].
In order to probe the LMs for their information on variable overlap, we input a concatenation of the two formulae in the model. On top of the [CLS] token embedding of the respective layer, we train a classification head which consists of a single linear layer. Identifying the variable overlap between the two formulae is a multi class, multi label classification with 27 classes. The output vector $o \in [0, 1]^{27}$ of the model is trained to contain a value between 0 and 1 for each of the 27 classes. $o_i = 0$ if the ith variable is not present in the overlap, and $o_i = 1$ if it is present.

4.2 Results

Fig. 3 shows the results for the [CLS] embedding of each layer, when evaluated on the Variable Overlap Prediction task (the results for this section are shown in solid lines, dashed are the results for Sec. 5). For each layer except the first two,

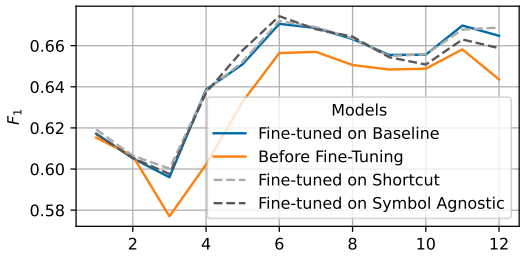

Fig. 3: F_1 Scores on Variable Overlap Prediction for each layer before and after fine-tuning on the different Math AR data sets.

the difference between the fine-tuned model and the model before fine-tuning is significant. It was possible to extract the information about the variable overlap with the highest F_1 score in Layer 6 and 11. The lowest scores receive Layer 3 for both setups.

This significant gap in almost all layers demonstrates that the information on variable overlap between both segments is more easily extractable from the [CLS] embedding. That indicates that fine-tuning reinforced this knowledge as it was more useful or important for the model. It could be the case that the relevance of a formula in an answer is determined only by judging the overlapping variables between question and answer. This way the model would not learn to rely on the structure of the formula, but only on the shallow feature of the names of the variables. A probable reason for the model to learn such a shortcut, which would make it easier for the model to receive a low loss during training, is that the fine-tuning data is constructed in such a way. The data contains the same amount of correct and wrong answers for each question. The correct answers are the answers to that question provided by users of StackExchange. These answers are likely to contain the same variables as the question since they refer to what was written in the question. Wrong answers are sampled from the corpus. The user who wrote these answers referred to other questions which in most cases contained other variables. Therefore, the model only needs to learn to look at the variables in question and answer. If the overlap is high, the answer is probably one of the correct ones, while it is a wrong answer when the variables do not match. However, we base this claim only on the observation of the gap in the Variable Overlap results. In the next section, we will train two more models that model the variable overlap between question and answer explicitly or not at all to verify our claim.

5 Symbol-agnostic Training

To verify whether our models heavily rely on the shortcut of only considering overlapping variables between question and answer when judging an answer's relevance, we construct two more training sets based on the initial training set.

Table 2: Example data pairs for the three fine-tuning data sets.

	Baseline	Shortcut	Symbol Agnostic
Question	I have a question regarding this calculation: How can I expand $(x+1)\cdot y$?		
Correct	The solution is $x \cdot y + y$	The solution is $x \cdot y + y$	The solution is $c \cdot r + r$
Incorrect	Let the probability $P(x)$	Let the probability $P(a)$	Let the probability $P(a)$

In the first modified training set, all examples adhere to the shortcut, which means that all correct answers contain the same variables as the question, while all incorrect answers contain other variables. During training, the model simply needs to learn to look at the overlap between variables in questions and answers to classify the relevance of an answer. We therefore call this fine-tuning set "shortcut data set". By evaluating models trained on this data set, we can compare it against models trained on the baseline data set and analyze if the renaming leads to significant drops in performance. Furthermore, when rerunning the evaluation of the variable overlap prediction we can see if the models trained on the shortcut data set receive even higher scores. In addition to this first set, we construct a training set where the models cannot learn the shortcut to look only at the variable overlap, but instead - hopefully - capture more structural information of the formulae. In this data set no overlapping variables exist between questions and answers and the model needs to focus on other features. We compare the performance of models trained on this second data set, which we call "symbol agnostic data set", to models trained on the baseline data set and the shortcut data set. If no significant differences can be found, the models do not rely on the shortcut.
In the following, we will explain in detail how the data sets are constructed. The models are trained in the same way as the baseline model in Sec. 2.2, but with the modified fine-tuning data set. For each data set we train five models with different random seeds. The evaluation of each sub-task is carried out as outlined in the previous two sections.

5.1 Renamed Fine-Tuning Data Sets

The base for the two additional data sets is the baseline data set, which consists of pairs of questions and answers, and their labels ("relevant" or "non-relevant" answer). For each question, we tokenized[3] the content of the math containers in the underlying HTML and stored its variables. Each answer first remained split into text and math container parts. We left the text parts as they were, but tokenized the math container content. For the shortcut data set, we only further processed the "non-relevant" answers. Here, we determined the overlapping variable names with the variables from the question. The variables in both

[3] We use a custom LaTeX tokenizer, e.g., `\sum_ix_i` is tokenized as `\sum _ i x _ i`

posts were renamed by other variables from the set of single letter variables from a to z and Greek letters, both upper and lower case. From this set of variables, we sampled randomly a variable name that was not contained in the question and did not already exist in the post. All instances of overlapping variables were renamed using this procedure. We kept the renaming consistent among all formulae of an answer post. After the renaming was conducted, text and formula parts were concatenated as in the baseline data set. The text and formula parts of the correct answers were concatenated in the same way, but without renaming. For the symbol-agnostic fine-tuning set, we follow the same procedure, but rename all overlapping variables in "relevant" and "non-relevant" answers. It should be noted, that the three sets, the baseline set, the shortcut set, and the symbol agnostic set, contain the same question-answer pairs, only the variable names differ. This way we minimize the influence of the concrete pairing of questions and answers and the influence of how the negative examples were selected. An example of the fine-tuning sets is displayed in Tab. 2.

5.2 Results

Results of the models on Math AR on each of the test sets of Sec. 3 can be seen in Tab. 3. Overall, the scores of the models trained on the three data sets are relatively close in all three metrics and on all test sets. All models show the same losses as the baseline model when removing, replacing, or sorting mathematical formulae in the answers. However, on each data set and each metric the symbol agnostic model demonstrates the highest relative and absolute losses in comparison to the default test data, except for p@10 on the sorted formula test set. For example, the relative loss on nDCG@1,000 when comparing the default data set to the sorted data set is 15.35% for the baseline model, 15.55% for the shortcut model, and 16.39% for the symbol agnostic model. We can therefore conclude that the symbol agnostic model relied the most on formulae in the answers when comparing it to the other models. Another possible explanation for this behavior could have been that the symbol agnostic model learns to not focus on formulae at all. But this would have led to a lower loss on the test sets without formula information (*No Math*). Since this is not the case, we conclude that the symbol agnostic models actually focus more on formulae. In addition, eliminating the shortcut also led to the (significantly) highest results of this model in both nDCG and p@10.

The F_1 scores on the Variable Overlap task follow the same trend for the models on all three data sets (dashed lines in Fig. 3). Nevertheless, it is noteworthy that significant differences between the models trained on the symbol agnostic data and the baseline and shortcut models exist in the layers 5 and 12. This could indicate that in the later layers, where the information is needed by the shortcut model, the symbol agnostic model does not rely on this information and therefore use its capacities to encode other information in its embeddings. Overall, our evaluation also showed that the differences between models trained on the baseline set and the shortcut set are small and mostly not significant. Especially on the Variable Overlap Prediction, there was not a single layer in

Table 3: Results of the additional models on all test sets for Math AR, rows indicate different test sets, columns indicate different fine-tuning data, *All Answers* denotes the default test set including judged and non-judged answers.

Data Set	nDCG@1,000			p@10		
	Baseline	Shortcut	Symbol Agn.	Baseline	Shortcut	Symbol Agn.
All Answers	0.4487	0.4437	0.4559	0.3733	0.3656	0.3794
Default	0.6990	0.6943	0.7019	0.3733	0.3656	0.3795
No Math	0.5438	0.5440	0.5422	0.1349	0.1410	0.1360
Dummy Math	0.5309	0.5334	0.5302	0.1434	0.1374	0.1397
Sorted Formulae	0.5917	0.5864	0.5868	0.2221	0.2183	0.2259

which significant differences between the models were visible. We can therefore conclude that the shortcut of only looking at the overlap of the variables in question and answer, which we artificially introduced with the shortcut data set, was already there and the baseline models learned to rely on it. That means that training sets for Math AR that only consider correct answers from their corresponding question will always contain the shortcut as long as the variables are not renamed.

6 Summary

The goal of this paper was to investigate which factors Transformer-Encoder-based Language Models rely on when they are employed for Mathematical Answer Retrieval. We developed an evaluation set up which includes a probing classifier and three evaluation data sets for the retrieval task to study the usage of formulae and their structural features as well as the overlap of variables between question and answer. In this study, we applied this evaluation set up to the model `AnReu/math_pretrained_bert`.
Our analysis demonstrated that the model considers formula information when assessing the relevance of an answer given a mathematical question. We further showed that the model loses a significant part of its modeling capacities of structural relationships of formulae after being fine-tuned on Mathematical Answer Retrieval. We attribute this to the fact that the fine-tuning data forces the model to consider mostly the overlap of variables between question and answer formulae, and not to use structural features. We verified this claim by demonstrating that there are no significant differences between a model that is fine-tuned on a data set that explicitly contains this malicious shortcut and the baseline data set. Furthermore, by removing the shortcut from the data, we improved the retrieval performance of the model. Our evaluation set up can easily be applied to further models for Mathematical Answer Retrieval and other fine-tuning tasks. Even though we only studied a model trained in a cross-encoder set up, the presented set up can also be used with ColBERT-based models [10] such as the one presented by Zhong et al. [33].

However, certain limitations still hold for our set up. All presented data sets on ARQMath data only considered formulae which were originally enclosed in math containers in the original HTML source, but users can also write formulae without LaTeX environments in their posts. These formulae would be unnoticed, since we considered everything outside of math containers as text. However, the proportion of unnoticed formulae is rather small, because the MathStackExchange community moderates and corrects posts, which leads to a high data quality in this regard. Nevertheless, it could still influence the results, since formulae whose variables were not renamed could still be present in the symbol agnostic fine-tuning set.
Future work should include further possible features that the models could use, for example, the matching of topic categories (such as matrix calculus, probability theory, etc.) of question and answers or the interplay between formulae and text. In this work, we only looked at overlapping variables between question and answer. In addition, one could also investigate the degree to which the tree distance between formulae in questions and answers is captured. The identified shortcut of only looking at the overlapping strings between questions and answers is most likely not limited to the formulas. The model might also learn to look at reoccurring natural language tokens, because answers will probably pick up terms and phrases from the question text. Because this hinders the model from learning to actually capturing the relevance between question and answer, a similar counteraction as renaming the variables during training should be conducted for the natural language text of the answers. Furthermore, future research should identify the components in the model that correspond to structural modeling, variable overlap or other features in order to gain more insights about where in the model knowledge is stored. Overall, these insights should eventually be used optimize to the fine-tuning procedure to train models for Math AR which make use of formulas in a more meaningful way.

7 Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful feedback and comments. 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.

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2019)

2. Belinkov, Y.: Probing classifiers: Promises, shortcomings, and advances. Computational Linguistics **48**(1), 207–219 (2022)
3. Del Barrio, E., Cuesta-Albertos, J.A., Matrán, C.: An optimal transportation approach for assessing almost stochastic order. In: The Mathematics of the Uncertain, pp. 33–44. Springer (2018)
4. Dror, R., Shlomov, S., Reichart, R.: Deep dominance - how to properly compare deep neural models. In: Korhonen, A., Traum, D.R., Màrquez, L. (eds.) Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers. pp. 2773–2785. Association for Computational Linguistics (2019). https://doi.org/10.18653/v1/p19-1266, https://doi.org/10.18653/v1/p19-1266
5. Dua, D., Wang, Y., Dasigi, P., Stanovsky, G., Singh, S., Gardner, M.: Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In: Proceedings of NAACL-HLT. pp. 2368–2378 (2019)
6. Fan, Y., Guo, J., Ma, X., Zhang, R., Lan, Y., Cheng, X.: A linguistic study on relevance modeling in information retrieval. In: Proceedings of the Web Conference 2021. pp. 1053–1064 (2021)
7. Geletka, M., Kalivoda, V., Štefánik, M., Toma, M., Sojka, P.: Diverse semantics representation is king. Proceedings of the Working Notes of CLEF 2022 (2022)
8. Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., Steinhardt, J.: Measuring mathematical problem solving with the math dataset. NeurIPS (2021)
9. Humeau, S., Shuster, K., Lachaux, M.A., Weston, J.: Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In: International Conference on Learning Representations (2019)
10. Khattab, O., Zaharia, M.: 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. pp. 39–48 (2020)
11. Mansouri, B., Agarwal, A., Oard, D., Zanibbi, R.: Finding old answers to new math questions: the arqmath lab at clef 2020. In: European Conference on Information Retrieval. pp. 564–571. Springer (2020)
12. Mansouri, B., Agarwal, A., Oard, D., Zanibbi, R.: Advancing math-aware search: The arqmath-2 lab at clef 2021 pp. 631–638 (2021)
13. Mansouri, B., Novotnỳ, V., Agarwal, A., Oard, D.W., Zanibbi, R.: Overview of arqmath-3 (2022): Third clef lab on answer retrieval for questions on math (working notes version). Proceedings of the Working Notes of CLEF 2022 (2022)
14. Mansouri, B., Oard, D.W., Zanibbi, R.: Dprl systems in the clef 2021 arqmath lab: Sentence-bert for answer retrieval, learning-to-rank for formula retrieval (2021)
15. Novotnỳ, V., Štefánik, M.: Combining sparse and dense information retrieval. Proceedings of the Working Notes of CLEF (2022)
16. O'Connor, J., Andreas, J.: What context features can transformer language models use? In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). pp. 851–864 (2021)
17. Pham, T., Bui, T., Mai, L., Nguyen, A.: Out of order: How important is the sequential order of words in a sentence in natural language understanding tasks? In: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021. pp. 1145–1160 (2021)
18. Polu, S., Sutskever, I.: Generative language modeling for automated theorem proving. arXiv preprint arXiv:2009.03393 (2020)

19. Qiao, Y., Xiong, C., Liu, Z., Liu, Z.: Understanding the behaviors of bert in ranking. arXiv preprint arXiv:1904.07531 (2019)
20. Reusch, A., Lehner, W.: Extracting operator trees from model embeddings. In: Proceedings of the 1st MathNLP Workshop (12 2022)
21. Reusch, A., Thiele, M., Lehner, W.: Transformer-encoder and decoder models for questions on math. Proceedings of the Working Notes of CLEF 2022 pp. 5–8 (2022)
22. Reusch, A., Thiele, M., Lehner, W.: Transformer-encoder-based mathematical information retrieval. In: International Conference of the Cross-Language Evaluation Forum for European Languages. pp. 175–189. Springer (2022)
23. Rohatgi, S., Wu, J., Giles, C.L.: Psu at clef-2020 arqmath track: Unsupervised reranking using pretraining. In: CEUR Workshop Proceedings. Thessaloniki, Greece (2020)
24. Saxton, D., Grefenstette, E., Hill, F., Kohli, P.: Analysing mathematical reasoning abilities of neural models. In: International Conference on Learning Representations (2019)
25. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: International conference on machine learning. pp. 3319–3328. PMLR (2017)
26. Ulmer, D., Hardmeier, C., Frellsen, J.: deep-significance: Easy and meaningful signifcance testing in the age of neural networks. In: ML Evaluation Standards Workshop at the Tenth International Conference on Learning Representations (2022)
27. Van Aken, B., Winter, B., Löser, A., Gers, F.A.: How does bert answer questions? a layer-wise analysis of transformer representations. In: Proceedings of the 28th ACM international conference on information and knowledge management. pp. 1823–1832 (2019)
28. Vashishth, S., Upadhyay, S., Tomar, G.S., Faruqui, M.: Attention interpretability across nlp tasks. arXiv preprint arXiv:1909.11218 (2019)
29. Wallat, J., Singh, J., Anand, A.: Bertnesia: Investigating the capture and forgetting of knowledge in bert. CoRR **abs/2106.02902** (2021), https://arxiv.org/abs/2106.02902
30. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Transformers: State-of-the-Art Natural Language Processing. pp. 38–45. Association for Computational Linguistics (10 2020), https://www.aclweb.org/anthology/2020.emnlp-demos.6
31. Zhan, J., Mao, J., Liu, Y., Zhang, M., Ma, S.: An analysis of bert in document ranking. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 1941–1944 (2020)
32. Zhong, W., Lin, S.C., Yang, J.H., Lin, J.: One blade for one purpose: Advancing math information retrieval using hybrid search. In: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 141–151 (2023)
33. Zhong, W., Yang, J.H., Lin, J.: Evaluating token-level and passage-level dense retrieval models for math information retrieval. arXiv preprint arXiv:2203.11163 (2022)