DPRL Systems in the CLEF 2022 ARQMath Lab: Introducing MathAMR for Math-Aware Search

Behrooz Mansouri[1], Douglas W. Oard[2] and Richard Zanibbi[1]

[1]*Rochester Institute of Technology, NY, USA*
[2]*University of Maryland, College Park, USA*

Abstract
There are two main tasks defined for ARQMath: (1) Question Answering, and (2) Formula Retrieval, along with a pilot task (3) Open Domain Question Answering. For Task 1, five systems were submitted using raw text with formulas in LaTeX and/or linearized MathAMR trees. MathAMR provides a unified hierarchical representation for text and formulas in sentences, based on the Abstract Meaning Representation (AMR) developed for Natural Language Processing. For Task 2, five runs were submitted: three of them using isolated formula retrieval techniques applying embeddings, tree edit distance, and learning to rank, and two using MathAMRs to perform contextual formula search, with BERT embeddings used for ranking. Our model with tree-edit distance ranking achieved the highest automatic effectiveness. Finally, for Task 3, four runs were submitted, which included the Top-1 results for two Task 1 runs (one using MathAMR, the other SVM-Rank with raw text and metadata features), each with one of two extractive summarizers.

Keywords
Community Question Answering (CQA), Mathematical Information Retrieval (MIR), Math-aware search, Math formula search

1. Introduction

The ARQMath-3 lab [1] at CLEF has three tasks. Answer retrieval (Task 1) and formula search (Task 2) are the tasks performed in ARQMath-1 [2] and -2 [3]. The ARQMath test collection contains Math Stack Exchange (MathSE)[1] question and answer posts. In the answer retrieval task, the goal is to return a ranked list of relevant answers for new math questions. These questions are taken from posts made in 2021 on MathSE. The questions are not included in the collection (which has only posts from 2010 to 2018). In the formula search task, a formula is chosen from each question in Task 1 as the formula query. The goal in Task 2 is to find relevant formulas from both question and answer posts in the collection, with relevance defined by the likelihood of finding materials associated with a candidate formula that fully or partially answers the question that a formula query is taken from. The formula-specific context is used in making relevance determinations for candidate formulas (e.g., variable and constant types, and operation definitions), so that formula semantics are taken into account. This year a new pilot Open-Domain Question Answering task was also introduced, where for the same questions

CLEF 2022 – Conference and Labs of the Evaluation Forum, September 21–24, 2022, Bucharest, Romania
bm3302@rit.edu (B. Mansouri); oard@umd.edu (D. W. Oard); rxzvcs@rit.edu (R. Zanibbi)

[1]math.stackexchange.com/

as in the Answer Retrieval task (Task 1), the participants were asked to extract or generate answers using data from any source.

The Document and Pattern Recognition Lab (DPRL) from the Rochester Institute of Technology (RIT, USA) participated in all three tasks. For Task 1, we have two categories of approaches. In the first approach, we search for relevant answers using Sentence-BERT [4] embeddings of raw text that includes the LaTeX representation for formulas given in the MathSE posts. In the second approach, we use a unified tree representation for text and formulas for search. For this, we consider the Abstract Meaning Representation (AMR) [5] for text, representing formulas by identifiers as placeholders, and then integrating the Operator Tree (OPT) representation of formulas into our AMR trees, forming MathAMR. The MathAMR representations are then linearized as a sequence, and Sentence-BERT is used for retrieval. We trained Sentence-BERT on pairs of (query, candidate) formulas with known relevance, and ranked (query, candidate) pairs with unknown relevance to perform the search.

Our runs in Task 1 are motivated by a common user behavior on community question answering websites such as MathSE. When there is a new question posted, the moderators can mark the question as duplicate if similar question(s) exist. We would expect that good answers to a similar question are likely to be relevant to a newly posted question. Our goal is use this strategy and to make this process automatic. First, we aim to find similar questions for a given topic in Task 1, and then rank the answers given to those similar questions.

For Task 2, we submitted two types of approaches. For the first type, we consider only isolated formulas during search: the context in which formulas occur are ignored for both query and candidates, with similarity determined by comparing formulas directly. In the second type of approach, we use contextual formula search. In contextual formula search, not only is formula similarity important, but also the context in which formulas appear. As in Task 1, we make use of AMR for text and then integrate the OPT into the AMR.

Finally, for Task 3, we select the first answers retrieved by two of our Task 1 runs, and then apply two extractive summarization models to each. These two summarizers select at most 3 sentences from each answer post returned by a Task 1 system.

In this paper, we first introduce the MathAMR representation, as it is used in our runs for all the tasks. We then explain our approaches for formula search, answer retrieval, and open-domain question answering tasks.

2. MathAMR

Formula Representations: SLTs and OPTs. Previously, math-aware search systems primarily used two representation types for formulas: Symbol Layout Trees (SLTs) capture the appearance of the formula, while Operator Trees (OPTs) capture formula syntax [6]. In an SLT, nodes represent formula elements (including variable, operator, number, etc.), whereas the edge labels capture their spatial relationships. In an OPT, nodes are again the formula elements, but the the edge labels indicate the order of the operands. For commutative operators such as '$=$', for which the order is not important, the edge labels are identical. Figure 1 shows the SLT and OPT representations for formula $x^n + y^n + z^n$.

In both representations, formula symbol types are given in the nodes (e.g., V! indicates that

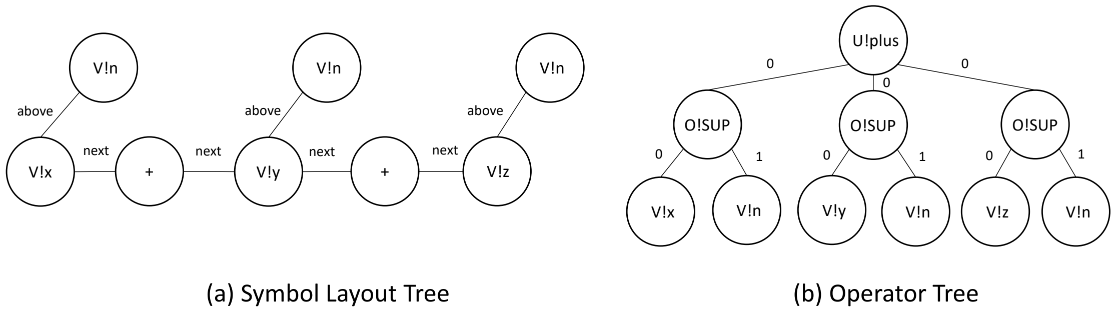

Figure 1: SLT (a) and OPT (b) representations for $x^n + y^n + z^n$. The nodes in SLT show the symbols and their types (with exception of operators). The edge labels *above* and *next* show the spatial relationship between symbols. Nodes in the OPT show symbols and their type (*U!* for unordered (commutative) operator, *O!* for ordered operator, and *V!* for variable identifiers). OPT edge labels indicate the ordering of operands.

the type is a variable). In our SLT represenation, there is no explicit node type for operators. SLT edge labels show the spatial relationship between symbols. For example, variable n is located *above* variable x, and operator $+$ is located *next* to variable x. As with a SLT, in an OPT the nodes represent the formula symbols. The difference is that in OPT representation, operators have an explicit node type. Unordered and ordered operators are shown with 'U!' and 'O!'. For further details refer to Davila et al. [7] and Mansouri et al. [8].

Operator trees capture the operation syntax in a formula. The edge labels provide the argument order for operands. By looking at the operator tree, one can see what operator is being applied on what operands. This is very similar to the representation of text with Abstract Meaning Representations (AMR), which can roughly be understood as representing "who is doing what to whom".[2]

Abstract Meaning Representation (AMR). AMRs are rooted Directed Acyclic Graphs (DAGs). AMR nodes represent two core concepts in a sentence: words (typically adjectives or stemmed nouns/adverbs), or frames extracted from Propbank [9].[3] For example in Figure 3, nodes such as 'you' and 'thing' are English words, while 'find-01' and 'solve-01' represent Propbank framesets. Labeled edges between a parent node and a child node indicate a semantic relationship between them. AMRs are commonly used in summarization [10, 11], question answering [12, 13], and information extraction [14, 15]. For example, Liu et al [10], generated AMRs for sentences in a document, and then merged them by collapsing named and date entities. Next, a summary sub-graph was generated using integer linear programming, and finally summary text was generated from that sub-graph using JARM [16].

Figure 2 shows an example summary of two sentences in their AMR representations [10]. There are two sentences:

(a) I saw Joe's dog, which was running in the garden.

[2]https://github.com/amrisi/amr-guidelines/blob/master/amr.md#part-i-introduction
[3]http://propbank.github.io/

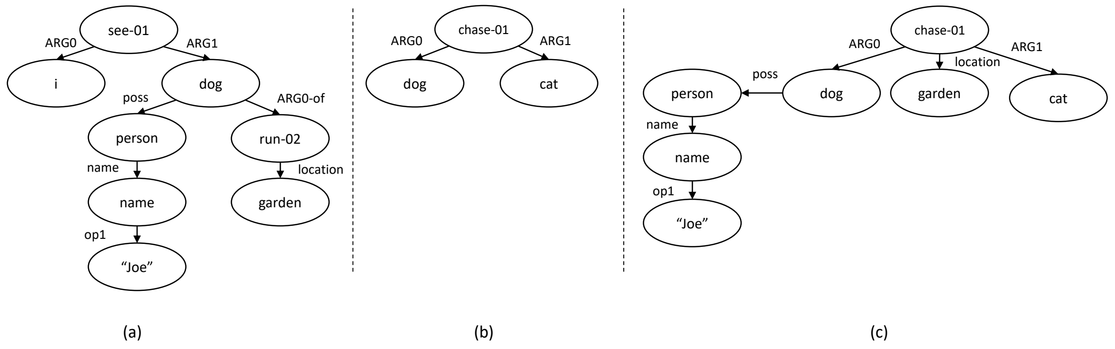

Figure 2: AMR summarization example adapted from Liu et al. [10]. (a) AMR for sentence 'I saw Joe's dog, which was running in the garden.' (b) AMR for a following sentence, 'The dog was chasing a cat.' (c) Summary AMR generated from the sentence AMRs shown in (a) and (b).

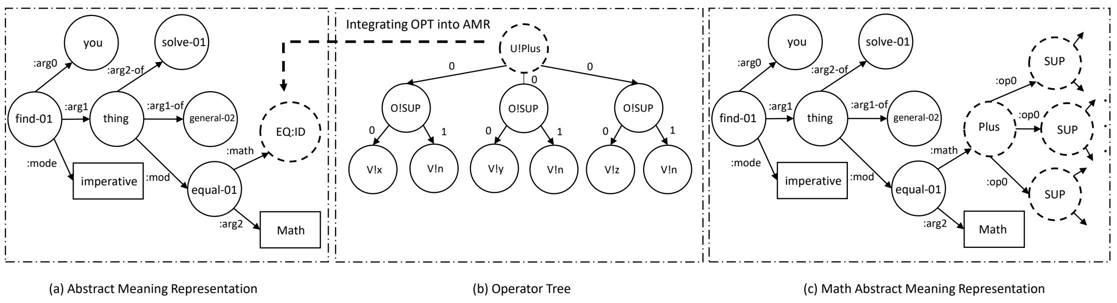

Figure 3: Generating MathAMR for the query "Find $x^n + y^n + z^n$ general solution" (ARQMath-2 topic A.289). (a) AMR tree is generated with formulas replaced by single tokens having ARQMath formula ids. (b) OPT formula representation is generated for formulas. (c) Operator tree root node replaces the formula place holder node. Note that in (c) the rest of OPT is not shown due to space limitations.

(b) The dog was chasing a cat.

Figure 2c shows the summary AMR generated for sentences (a) and (b).

To generate AMRs from text, different parsers have been proposed. There are Graph-based parsers that aim to build the AMR graph by treating AMR parsing as a procedure for searching for the Maximum Spanning Connected Subgraphs (MSCGs) from an edge-labeled, directed graph of all possible relations. JAMR [16] was the first AMR parser, developed in 2014, and it used that approach. Transition-based parsers such as CAMR [17], by contrast, first generate a dependency parse from a sentence and then transform it into an AMR graph using transition rules. Neural approaches instead view the problem as a sequence translation task, learning to directly convert raw text to linearized AMR representations. For example, the SPRING parser [18] uses depth-first linearization of AMR, and views the problem of AMR generation as translation problem, translation raw text to linearized AMRs with a BART transformer model [19] by modifying its tokenizer to handle AMR tokens.

While AMRs can capture the meaning (semantics) of text, current AMR parsers fail to correctly parse math formulas. Therefore, in this work, we introduced MathAMR. Considering the text "Find $x^n + y^n + z^n$ general solution" (the question title for topic A.289 in ARQMath-2 Task 1), Figure 3 shows the steps to generate the MathAMR for this query. First, each formula is replaced with a placeholder node that includes the identifier for that formula. In our example, we show this as "EQ:ID", where in practice ID would be the formula's individual identifier in the ARQMath collection.

The modified text is then presented to an AMR parser. For our work we used the python-based AMRLib,[4] using the model "model_parse_xfm_bart_large". Figure 3(a) shows the output AMR. Nodes are either words or concepts (such as 'find-01') from the PropBank framesets [9]. The edge labels show the relationship between the nodes. In this instance, 'arg0' indicates the subject and 'arg1' the object of the sentence. We introduce a new edge label 'math' to connect a formula's placeholder node to its parent. For further information on AMR notation, see [20].

Figure 3(b) is the OPT representation of the formula, which is integrated into the AMR by replacing the placeholder with the root of the OPT, thus generating what we call MathAMR. This is shown in Figure 3(c). To follow AMR conventions, we rename the edge labels from numbers to 'opX' where 'X' is the edge label originally used in the OPT. We use the edge label 'op' as in AMRs edge labels capture the relation and its ordering. In the next sections, we show how MathAMR is used for search.

This is an early attempt to introduce a unified representation of text and formula using AMRs. Therefore, we aim to keep our model simple and avoid other information related to the formula that could have been used. For example, our current model only uses the OPT formula representation, where as previous researches have shown using SLT representation can be helpful as well [8, 21, 22]. In our current model we only use OPTs. Also, we are using the AMR parser that is trained on general text not specific for math which is a limitation of our work. For other domains such as biomedical research, there exist pre-trained AMR parsers [23].

3. Task 2: Formula Retrieval

Because of our focus on formula representation in AMRs, we start by describing our Task 2 runs. In Section 4 we then draw on that background to describe our Task 1 runs.

In the formula retrieval task, participating teams were asked to return a set of relevant formulas for a given formula query. Starting in ARQMath-2 [24], the relevance criteria for Task 2 were defined in such a way that a formula's context has a role in defining its relevance. Therefore, in our ARQMath-3 Task 2 runs we use athAMR to create a unified representation of formulas and text. In addition to the MathAMR model, we also report results from some of our previous isolated formula search models for this task, as they yielded promising results in ARQMath-1 and -2.

3.1. Isolated Formula Search Runs

For isolated formula search, we created three runs. These runs are almost identical to what we had in ARQMath-2. Therefore, we provide a brief summary of the systems, along with

[4] https://github.com/bjascob/amrlib

differences compared to our previous year's systems. Please eefer to Mansouri et al. [25] for more information.

TangentCFT-2: Tangent-CFT [8] is an embedding model for mathematical formulas that considers SLT and OPT representations of the formulas. In addition to these representations, two unified representations are considered where only types are represented when present in SLT and OPT nodes, referred to as SLT TYPE and OPT TYPE.

Tangent-CFT uses Tangent-S [21] to linearize the tree representations of the formulas. Tangent-S represents formulas using tuples comprised of symbol pairs along with the labeled sequence of edges between them. These tuples are generated separately for SLT and OPT trees. In Tangent-CFT we linearize the path tuples using depth-first traversals, tokenize the tuples, and then embed each tuple using an n-gram embedding model, implemented using fastText [26]. The final embedding of a formula SLT or OPT is the average of its constituent tuple embeddings. Our fastText models were trained on formulas in the ARQMath collection. Using the same pipeline for training, each formula tree is linearized with Tangent-S, then tokenized with Tangent-CFT, and their vector representations are extracted using trained models.

In our run, we use the MathFIRE[5] (Math Formula Indexing and Retrieval with Elastic Search) system. In this system, formula vector representations are extracted with Tangent-CFT, then loaded in OpenSearch [6] where dense vector retrieval was performed by approximate k-NN search using nmslib and Faiss [27]. We used the default parameters. Note that in our previous Tangent-CFT implementations we had used exhaustive rather than approximate nearest neighbor search.

As there are four retrieval results from four different representations, we combined the results using modified Reciprocal Rank Fusion [28] as:

$$RRFscore(f \in F) = \sum_{m \in M} \frac{s_m(f)}{60 + r_m(f)} \tag{1}$$

where the s_m is the similarity score and r_m is the rank of the candidate. As all the scores from the retrieval with different representations are cosine similarity scores with values in the interval $[0, 1]$, we did not apply score normalization.

TangentCFT2TED (Primary Run): Previous experiments have shown that Tangent-CFT can find partial matches better than it can find full-tree matches [29]. As it is an n-gram embedding model, it focuses on matching of n-grams. Also, vectors aim to capture features from the n-gram appearing frequently next to each other, and this approach has less of a focus on structural matching of formulas. Therefore, in TangentCFT2TED, we rerank the top retrieval results from TangentCFT-2 using tree edit distance (TED). We considered three edit operations: deletion, insertion, and substitution. Note that in our work, we only consider the node values and ignore the edge labels. For each edit operation, we use weights learnt on the NTCIR-12 [22] collection.[7] We use an inverse tree-edit distance score as the similarity score:

$$sim(T_1, T_2) = \frac{1}{TED(T_1, T_2) + 1}. \tag{2}$$

[5]https://gitlab.com/dprl/mathfire

[6]https://opensearch.org/

[7]We have also trained weights on ARQMath-1, and those weights were similar to those trained on the NTCIR-12 collection.

The tree edit distance was used on both SLT and OPT representations, and the results were combined using modified Reciprocal Rank Fusion as in equation 1.

Because in ARQMath-2 this run had the highest nDCG$'$, this year we annotated it as our primary run, for which the hits are pooled to a greater depth.

Learning to Rank: Our third isolated formula search model is a learning to rank approach for formula search that we introduced in [29]. In this model, sub-tree, full-tree, and embedding similarity scores are used to train an SVM-rank model [30]. Our features are:

- Maximum Sub-tree Similarity (MSS) [7]
- Tuple and node matching scores [7]
- Weighted and Unweighted tree edit distance scores [29]
- Cosine similarity from the Tangent-CFT model

All features, with the exception of MSS, were calculated using both OPT and SLT representations, both with and without unification of node values to types. The MSS features were calculated only on the unified OPT and SLT representations. MSS is computed from the largest connected match between the formula query and a candidate formula obtained using a greedy algorithm, evaluating pairwise alignments between trees using unified node values. Tuple matching scores are calculated by considering the harmonic mean of the ratio of matching symbol pair tuples between a query and the candidates. The tuples are generated using the Tangent-S [21] system, which traverses the formula tree depth-first and generates tuples for pairs of symbols and their associated paths.

For training, we used all topics from ARQMath-1 and -2, a total of about 32K pairs. Following our original proposed approach in [29], we re-rank Tangent-S results using linearly weighted features, with weights defined using SVM-rank.

3.2. Contextual Formula Search Runs

Some previous approaches to combined formula and text search combined separate search results from isolated formula search and text retrieval models. For example, Zhong et al. [31] combined retrieval results from a isolated formula search engine, Approach0 [32], and for text used the Anserini toolkit. Similarly, Ng et al. [33] combined retrieval results from Tangent-L [34] and BM25+. In the work of Krstovski et al. [35], by contrast, equation embeddings generated unified representations by linearizing formulas as tuples and then treated them as tokens in the text. These equation embeddings utilized a context window around formulas and used a word embedding model [36] to construct vector representations for formulas.

While our first category of runs focused on isolated formula search, in our second category we made use of the formula's context. Our reasoning is based in part on the potential for complementary between different sources of evidence for meaning, and in part on the relevance definition for Task 2, where a candidate formula's interpretation in the context of its post matters. In particular, it is possible for a formula identical to the query be considered not relevant. As an example from ARQMath-2, for the formula query $x^n + y^n + z^n$ (B.289), some exact matches were considered irrelevant, as for that formula query, x, y, and z could (according to the question text) be any real numbers. The assessors thus considered all exact matches in the pooled posts in which x, y, and z referred not to real numbers but specifically to integers as not relevant.

Therefore, in our contextual formula search model we use MathAMR for search. For each candidate formula, we considered the sentence in which the formula appeared along with a sentence before and after that (if available) as the context. We then generated MathAMR for each candidate. To generate MathAMR for query formulas, we used the same procedure, using the same context window. For matching, we made use of Sentence-BERT Cross-Encoders [4]. To use Sentence-BERT, we traversed the MathAMR depth-first. For simplicity, we ignored the edge labels in the AMRs. For Figure 3(c) (included the full OPT from Figure 3(b)), the linearized MathAMR string is:

```
find-01 you thing general-02 solve-01 equal-01 plus SUP z n SUP y n
SUP x n imperative
```

To train a Sentence-BERT cross encoder, we used the pre-trained 'all-distilroberta-v1' model and trained our model with 10 epochs using a batch size of 16 and a maximum sequence size of 256. Note that our final model is what is trained after 10 epochs on the training set, and we did not use a separate validation set. For training, we made use of all the available pairs from ARQMath-1 and -2 topics. For labeling the data, high and medium relevance formula instances were labeled 1, low relevance instances were labeled 0.5, and non-relevant instances 0. For retrieval, we re-rank the candidates retrieved by the Tangent-CFTED system (top-1000 results). Note that candidates are ranked only by the similarity score from our Sentence-BERT model.

Our fifth run combined the search results from MathAMR and Tangent-CFT2TED systems. This was motivated by MathAMR embeddings for sentences with multiple formulas having the same representation, meaning that all formulas in that sentence receive the same matching score. Because Tangent-CFT2TED performs isolated formula matching, combining these results helps avoid this issue. For combining the results, we normalize the scores to the range 0 to 1 with Min-Max normalization and use modified Reciprocal Rank Fusion as given in Equation 1.

Additional Unofficial Post Hoc Run. For our MathAMR model in our official submission we used three sentences as the context window: the sentence in which the candidate formula appears in, a sentence before and as sentence after that. We made a change to the context window size and considered only the sentence in which the formula appears as the context.

Also, in our previous approach with context window of three sentences, we simply split the question post containing the query formula at periods (.) in the body text, and then choose the sentence with the candidate formula. However, a sentence can end with other punctuation such as '?'. Also, formulas are delimited within $\LaTeX$ by '$'; these formula regions commonly contain sentence punctuation. To address these two issues, in our additional run, we first move any punctuation (. , ! ?) from the end of formula regions to after final delimiter. Then, we use Spacy[8] to split paragraphs into sentences and choose the sentence that a formula appears in. After getting the results with using a context window size of one, we also consider the modified reciprocal rank fusion of this system with Tangent-CFT2ED as another additional post-hoc run.

[8]https://spacy.io/

Table 1
DPRL Runs for Formula Retrieval (Task 2) on ARQMath-1 (45 topics) and ARQMath-2 (58 topics) for topics used in training (i.e., test-on-train). The DATA column indicates whether isolated math formulas, or both math formulas and surrounding text are used in retrieval.

		EVALUATION MEASURES					
FORMULA RETRIEVAL		ARQMATH-1			ARQMATH-2		
RUN	DATA	nDCG′	MAP′	P′@10	nDCG′	MAP′	P′@10
LtR	Math	**0.733**	**0.532**	0.518	0.550	0.333	0.491
Tangent-CFT2	Math	0.607	0.438	0.482	0.552	0.350	0.510
Tangent-CFT2TED	Math	0.648	0.480	0.502	0.569	0.368	0.541
MathAMR	Both	0.651	0.512	0.567	0.623	0.482	0.660
Tangent-CFT2TED+MathAMR	Both	0.667	0.526	**0.569**	**0.630**	**0.483**	**0.662**

3.3. Experiment Results

This section describes the results of our runs on the ARQMath-1, -2 and -3 Task 2 topics.

ARQMath 1 and -2 Progress Test Results. Table 1 shows the results of our progress test runs on ARQMath-1 and -2 topics. Note that because some of our systems are trained using relevance judgments for ARQMath-1 and -2 topics, those results should be interpreted as training results rather than as a clean progress test since some models (and in particular MathAMR) may be over-fit to this data.[9]

To compare isolated vs contextual formula search, we look at results from Tangent-CFT2TED and MathAMR runs. Using MathAMR can be helpful specifically for formulas for which context is important. For example, in query $f(x) = \frac{1}{1+\ln^2 x}$ (B.300 from ARQMath-2), the formula is described as "is uniformly continuous on $I = (0, \infty)$". Similar formulas such as $g(x) = \frac{1}{x \ln^2 x}$, that in isolation are less similar to the query are not ranked in the top-10 results from Tangent-CFT2ED. However, with MathAMR, as this formula in its context has the text "is integrable on $[2, \infty)$"; it was ranked 4th by MathAMR. The P′@10 for this query is 0.1 for Tangent-CFT2TED and 0.7 for MathAMR.

In contrast to this, there were cases where P′@10 was lower for MathAMR compared to Tangent-CFT2TED. As an example, for formula query, B.206, appearing in the title of a question as: "I'm confused on the limit of $\left(1+\frac{1}{n}\right)^n$", a low relevant formula appearing in the same context, "Calculate limit of $(1+\frac{1}{n^2})^n$" gets higher rank in MathAMR than Tangent-CFT2ED. Both query and candidate formula appear in the questions' title and there is no additional useful information in the text other then the word 'limit' is not providing any new information. Therefore, we can consider this another limitation in our current model, that we are not distinguishing between formula queries that are or are not dependent on the surrounding text, and also there is no pruning applied to MathAMR to remove information that is not necessary helpful.

ARQMath-3 Results. Table 2 shows the Task 2 results on ARQMath-3. Tangent-CFT2TED achieved the highest nDCG′ among our models, significantly better than other representations, except Tangent-CFT2TED+MathAMR ($p < 0.05$, t-test with Bonferroni correction).

[9]This progress-test-on-train condition was the condition requested by the ARQMath organizers; all systems were to be run on ARQMath-1 and ARQMath-2 topics in the same configuration as they were run on ARQMath-3 topics.

Table 2
DPRL Runs for Formula Retrieval (Task 2) on ARQMath-3 (76) topics. Tangent-CFT2TED is our primary run.

Formula Retrieval Run	Data	nDCG′	MAP′	P′@10
Tangent-CFT2TED	Math	**0.694**	**0.480**	0.611
Tangent-CFT2	Math	0.641	0.419	0.534
Tangent-CFT2TED+MathAMR	Both	0.640	0.388	0.478
LtR	Math	0.575	0.377	0.566
MathAMR	Both	0.316	0.160	0.253
Additional Unofficial Post Hoc				
Tangent-CFT2TED+MathAMR	Both	0.681	0.471	**0.617**
MathAMR	Both	0.579	0.367	0.549

We compare our Tangent-CFT2TED model with MathAMR, looking at the effect of using context. One obvious pattern is that using MathAMR can help with topics for which variables are important. For example, for topic $F = P \oplus T$. (Topic B.326), P is a projective module and F is a free module. There are instances retrieved in the top-10 results by TangentCFT2ED, such as $V = A \oplus B$, where variables are referring to different concepts; in this a formula k-dimensional subspace. With MathAMR, formulas such as $P \oplus Q = F$ appearing in a post that specifically says: "If P is projective, then $P \oplus Q = F$ for some module P and some free module F." (similar text to the topic) are ranked in the top-10 results.

For cases that Tangent-CFT2ED has better effectiveness, two patterns are observed. In the first pattern, the formula is specific and variables do not have specifications. In the second pattern, the context is not helpful (not providing any useful information) for retrieval. For instance, topic B.334, "logarithm proof for $a^{log_a(b)} = b$" the formula on its own is informative enough. Low relevant formulas appearing in a context such as "When I tried out the proof, the final answer I ended up with was $a^{log_b n}$" are ranked in the top-10 results because of having proof and part of formula.

Combining the results on Tangent-CFT2ED and MathAMR with our modified RRF provided better P′@10 than one of the individual system results for only 10% of the topics. For the topic (B.338) appearing in the a title of a question as "Find all integer solutions of equation $y = \frac{a+bx}{b-x}$", both Tangent-CFT2ED and MathAMR had P′@10 of 0.6. However combining the results with modified RRF increases the P′ value to 0.9. Table 3 shows the top-10 results for Tangent-CFT2ED+MathAMR, along with the original ranked lists for the Tangent-CFT2ED and MathAMR systems. As can be seen, there are relevant formula that Tangent-CFT2ED or MathAMR model gave lower rank to, but the other system provided a better ranking and combining the systems with our modified RRF improved the results.

4. Task 1: Answer Retrieval

The goal of the ARQMath answer retrieval task is to find relevant answers to the mathematical questions in a collection of MathSE answer posts. These are new questions that were asked after

Table 3
Top-10 Formulas Retrieved by Tangent-CFT2ED+MathAMR along with their ranks in original Tangent-CFT2ED and MathAMR runs for topic (B.338), appearing in a question post title as "Find all integer solutions of equation $y = \frac{a+bx}{b-x}$". For space, sentences for formula hits (used by MathAMR) are omitted.

TangentCFT+MathAMR Top-10 Formula Hits	Relevance Score	Tangent-CFT2TED Rank	MathAMR Rank
1. $y = \frac{a+bx}{c+x}$	2	1	10
2. $y = \frac{a+bx}{x+c}$	2	3	88
3. $y = \frac{a+x}{b+cx}$	2	8	8
4. $y = \frac{a+bx}{c+dx}$	2	2	30
5. $y = \dfrac{bx}{x-a}$	1	29	5
6. $y = \frac{ax+b}{cx+d}$	3	53	2
7. $y = \frac{b+dx}{1-b-dx}$	2	4	42
8. $g(x) = \frac{a+bx}{b+ax},$	2	7	31
9. $y = \lvert\frac{b+cx}{a+x}\rvert$	2	27	9
10. $y = \frac{b-x}{1-bx}$	2	19	14

the posts in the ARQMath collection (i.e., not during 2010-2018). Our team provided 5 runs[10] for this task. Two of our runs considered only text and LaTeX representation of the formulas. Two other runs used strings created by depth-first MathAMR tree traversals. Our fifth run combined the retrieval results from the two runs, one from each of the approaches.

4.1. Raw Text Approaches

We submitted two runs that use the raw text, with formulas being represented with LaTeX representations. In both runs, we first find similar questions for the given question in Task 1 and then compile all the answers given to those questions and re-rank them based on the similarity to the question.

4.1.1. Candidate Selection by Question-Question Similarity

To identify questions similar to a topic, we started with a model pre-trained on the Quora question pairs dataset,[11] and then fine-tuned that model to recognize question-question similarity in actual ARQMath questions. To obtain similar training questions we used the links in the ARQMath collection (i.e., the data from 2010-2018 that predates the topics that we are actually searching) to related and duplicate questions. Duplicate question(s) are marked by MathSE moderators as having been asked before, whereas related questions are marked by MathSE moderators as similar to, but not exactly the same as, the given question. We applied two-step fine-tuning: first using both related and duplicate questions, and then fine-tuning more strictly using only duplicate questions. We used 358,306 pairs in the first round, and 57,670 pairs in the second round.

[10]Note that ARQMath teams are limited to 5 submissions.
[11]https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

For training, we utilized a multi-task learning framework provided by Sentence-BERT, used previously for detecting duplicate Quora questions in the ‘distilbert-base-nli-stsb-quora-ranking‘ model. This framework combines two loss functions. First, the contrastive loss [37] function minimizes the distance between positive pairs and maximizes the distance between for negative pairs, making it suitable for classification tasks. The second loss function is the multiple negatives ranking loss [38], which considers only positive pairs, minimizing the distance between positive pairs out of a large set of possible candidates, making it suitable for ranking tasks. We expected that with these two loss functions we could distinguish well between relevant and not-relevant candidates, and also rank the relevant candidates well by the order of their relevance degrees.

We set the batch size to 64 and the number of training epochs to 20. The maximum sequence size was set to 128. In our training, half of the samples were positive and the other half were negative, randomly chosen from the collection. In the first fine-tuning, a question title and body are concatenated. In the second fine-tuning, however, we considered the same process for training, with three different inputs:

- Using the question title, with a maximum sequence length of 128 tokens.
- Using the first 128 tokens of the question body.
- Using the last 128 tokens of the question body.

To find a similar question, we used the three models to separately retrieve the top-1000 most similar questions. The results were combined by choosing the maximum similarity score for each candidate question across the three models.

4.1.2. Candidate Ranking by Question-Answer Similarity

Having a set of candidate answers given to similar questions, we re-rank them differently in our two runs, as explained below.

QQ-QA-RawText. In our first run, we used QASim (Question-Answer Similarity) [25] which achieved our highest nDCG$'$ value in ARQMath-2. Our training procedure is the same as for our ARQMath-2 system, but this time we added ARQMath-2 training pairs to those from ARQMath-1. For questions, we used the concatenation of title and body, and for the answer we choose only the answer body. For both questions and answers, the first 256 tokens are chosen. For ranking, we compute the similarity score between the topic question and the answer, and the similarity score between the topic question and the answer’s parent question. We multiplied those two similarity scores to get the final similarity score. Our pre-trained model is Tiny-BERT, with 6 layers trained on the “MS Marco Passage Reranking” [39] task. The inputs are triplets of (Question, Answer, Relevance), where the relevance is a number between 0 and 1. In ARQMath, high and medium relevance degrees were considered as relevant for precision-based measures. Based on this, for training, answers from ARQMath-1 and -2 assessed as high or medium got a relevance label of 1, a label of 0.5 was given to those with low relevance, and 0 was given for non-relevant answers. For the system details refer to [25].

SVM-Rank (Primary Run). Previous approaches for the answer retrieval task have shown that information such as question tags and votes can be useful in finding relevant answers [33, 31]. We aimed to make use of these features and study their effect for retrieval. In this second run (which we designated as our primary Task 1 run), we considered 6 features: Question-Question similarity (QQSim) score, Question-Answer similarity (QASim) score, number of comments on the answer, the answer's MathSE score (i.e, upvotes−downvotes), a binary field showing if the answer is marked as selected by the asker (as the best answerto their question), and the percentage of topic question post tags that the question associated with an answer post also contains (which we refer to as question tag *overlap*). Note that we did not apply normalization to feature value ranges. We trained a ranking SVM model [30] using all the assessed pairs from ARQMath-1 and -2, calling the result SVM-Rank. After training, we found that QQSim, QASim, and overlap between the tags were the most important features, with weights 0.52, 2.42 and 0.05, respectively, while the weights for other features were less than 0.01.

Both out QQ-QA-RawText and SVM-Rank models have the same first stage retrieval, using Sentence-BERT to find similar questions. Then the candidates are ranked differently. While both approaches make use of Question-Question and Question-Answer similarity scores (using Sentence-BERT), the second approach considers additional features and learns weights for the features using ARQMath-1 and -2 topics.

4.2. MathAMR Approaches

In our second category of approaches, we made use of our MathAMR representation, providing two runs. As in our raw text-based approach, retrieval is comprised of two stages: identifying candidates from answers to questions similar to a topic question, and then ranking candidate answers by comparing them with the topic question.

4.2.1. Candidate Selection by Question-Question Similarity

In our first step, we find similar questions for a given question in Task 1. For this, we only focus on the question title. Our intuition is that AMR was designed to capture meaning from a sentence. As the question titles are usually just a sentence, we assume that similar questions can be found by comparing AMR representations of their titles.

Following our approach in Task 2, we generated MathAMR for each question's title. Then the MathAMRs are linearized using a depth-first traversal. We used a model that we trained on RawText for question-question similarity as our pretrained model, although in this case we trained on the question titles. We used the known duplicates from the ARQMath collection (2010-2018) to fine tune our model on the linearized AMR of questions, using a similar process as for raw text.

4.2.2. Candidate Ranking by Question-Answer Similarity

Answers to similar questions are ranked in two ways for our two Task 1 AMR runs.

QQ-MathSE-AMR. Using a question title's MathAMR, we find the top-1000 similar questions for each topic. Starting from the most similar question and moving down the list, we compile the answers given to the similar questions. The answers for each similar question are ranked based on their MathSE score (i.e., upvotes−downvotes). To determine the similarity score of topic and an answer, we used the reciprocal of the rank after getting the top-1000 answers. Note that this approach does not use any topics from ARQMath-1 or -2 for training.

QQ-QA-AMR. This run is similar to our QQ-QA-RawText run, but instead of raw text representations, we use MathAMR representations. For similarity of questions, we only use the question titles, while for similarity of a question and an answer we use the first 128 tokens of the linearized MathAMR from the post bodies of the question and the answer. We trained a Sentence-BERT model, and did retrieval, similarly to our QAsim model with two differences: (1) we used 'all-distilroberta-v1' as the pre-trained model (2) instead of raw text we use linearized MathAMR. The parameters for Sentence-BERT such as number of epochs, batch size and loss function are the same. Our Sentence-BERT design is similar to the QAsim we had used for raw text in ARQMath-2 [25]. We used both ARQMath-1 and -2 topics from Task 1 for training.

For our fifth run, we combined the results from our SVM-Rank model (from raw text approaches) and QQ-QA-AMR (from MathAMR approaches) using modified reciprocal rank fusion, naming that run RRF-AMR-SVM.

Additional Unofficial Post Hoc Runs. In ARQMath-2021, we had two other runs using raw text representations that we also include here for ARQMath-3 topics, using post hoc scoring (i.e., without these runs having contributed to the judgement pools). One is our 'QQ-MathSE-RawText' run, which uses question-question (QQ) similarity to identify similar questions and then ranks answers associated with similar question using MathSE scores (upvotes−downvotes). The similarity score was defined as:

$$Relevance(Q_T, A) = QQSim(Q_T, Q_A) \cdot MathSE_{score}(A) \tag{3}$$

where the Q_T is the topic question, A is a candidate answer and Q_A is the question to which answer A was given. The other is our 'RRF-QQ-MathSE-QA-RawText' run, which combines retrieval results from two systems, 'QQ-MathSE-RawText' and 'QQ-QA-RawText', using our modified reciprocal rank fusion.

A third additional unofficial post hoc run that we scored locally is 'QQ-MathSE(2)-AMR'. To find similar questions, this model uses the exact same model as 'QQ-MathSE-AMR'. However, for ranking the answers, instead of the ranking function used for 'QQ-MathSSE-AMR', we use the ranking function in equation (3).

For corrected runs, we fixed an error for 'QQ-QA-RawText' model and report the results. This model affects two other models, "SVM-rank model" and "RRF-AMR-SVM". Therefore, we report the results on these systems as well.

4.3. Experiment Results

ARQMath 1 and -2 Results. Table 4 shows the results of our progress test runs for Task 1 on ARQMath-1 and -2 topics. As with our Task 2 progress test results, those results should be interpreted as training results rather than as a clean progress test since some models may be over-fit to this data. Note that the runs in each category of Raw Text and MathAMR have the same set of candidates to rank, which may lead to similar effectiveness measures.

ARQMath-3 Results. Table 5 shows the RPDL Task 1 results on ARQMath-3 topics along with baseline Linked MSE post that our models aim to automate. Our highest nDCG′ and mAP′ are achieved by our additional unofficial 'QQ-MathSE-RawText' run, while our highest P′@10

Table 4
DPRL progress test Runs for Answer Retrieval (Task 1) progress test on ARQMath-1 (71 topics) and ARQMath-2 (77 topics) for topics used in training (test-on-train). All runs use both text and math information. Stage-1 selects answers candidates that are then ranked in Stage-2. SVM-Rank is the primary run.

Answer Retrieval	Stage-1	Stage-2	ARQMath-1			ARQMath-2		
Run	Selection	Ranking	nDCG′	MAP′	P′@10	nDCG′	MAP′	P′@10
QQ-QA-AMR	QQ-MathAMR	QQSIM x QASIM (MathAMR)	0.276	0.180	0.295	0.186	0.103	0.237
QQ-MathSE-AMR	QQ-MathAMR	MathSE	0.231	0.114	0.218	0.187	0.069	0.138
QQ-QA-RawText	QQ-RawText	QQSIM x QASIM (RawText)	0.511	**0.467**	0.604	0.532	**0.460**	**0.597**
SVM-Rank	QQ-RawText	SMV-Rank	0.508	0.467	0.604	**0.533**	**0.460**	0.596
RRF-AMR-SVM	—	—	**0.587**	**0.519**	**0.625**	**0.582**	**0.490**	0.618

Table 5
DPRL Runs for Answer Retrieval (Task 1) on ARQMath-3 (78) topics along with the Linked MSE posts baseline. SVM-Rank is the primary run. For Post Hoc runs, (C) indicates corrected run, and (A) indicates additional run. Linked MSE posts is a baseline system provided by ARQMath organizers.

Answer Retrieval	Stage-1	Stage-2	Evaluation Measures		
Run	Selection	Ranking	nDCG′	MAP′	P′@10
Linked MSE posts	-	-	0.106	0.051	0.168
SVM-Rank	QQ-RawText	SVM-Rank	0.283	0.067	0.101
QQ-QA-RawText	QQ-RawText	QQSIM x QASIM (RawText)	0.245	0.054	0.099
QQ-MathSE-AMR	QQ-MathAMR	MathSE	0.178	0.039	0.081
QQ-QA-AMR	QQ-MathAMR	QQSIM x QASIM (MathAMR)	0.185	0.040	0.091
RRF-AMR-SVM	-	-	0.274	0.054	0.022
Post Hoc Runs					
QQ-QA-RawText (C)	QQ-RawText	QQSIM x QASIM (RawText)	0.241	0.030	**0.151**
SVM-Rank (C)	QQ-RawText	SVM-Rank	0.296	0.070	0.101
RRF-AMR-SVM (C)	-	-	0.269	0.059	0.106
QQ-MathSE-RawText (A)	QQ-RawText	MathSE	**0.313**	**0.147**	0.087
RRF-QQ-MathSE-QA-RawText (A)	-	-	0.250	0.067	0.110
QQ-MathSE(2)-AMR (A)	QQ-MathAMR	MathSE(2)	0.200	0.044	0.100

is for the our unofficial corrected "QQ-QA-RawText" run. Comparing the QQ-QA models using MathAMR or raw text, in 41% of topics raw text provided better P′@10, while with MathAMR a higher P′@10 was achieved for 21% of topics. In all categories of dependencies (text, formula, or both), using raw text was on average more effective than MathAMR. The best effectiveness for MathAMR was when questions were text dependent, with an average P′@10 of 0.12, over the 10 assessed topics dependent on text. Considering topic types, for both computation and proof topics, P′@10 was 0.10 and 0.06 higher, respectively, using raw text than MathAMR. For concept topics, P′@10 was almost the same for the two techniques. Considering topic difficulty, only for hard questions did MathAMR do even slightly better numerically than raw text by P′@10, with just a 0.01 difference. Among those topics that did better at P′@10 using MathAMR, 94% were hard or medium difficulty topics.

To further analyze our approaches, we look at the effect of different representations on individual topics. With both raw text and MathAMR, selecting candidates is done by first finding similar questions. Considering the titles of questions to find similar questions, there are cases where MathAMR can be more effective due to considering OPT representations. For

Table 6
Titles of the top-5 most similar questions found with MathAMR and raw text, for the topic question with title *"Proving* $\sum_{k=1}^{n} \cos\frac{2\pi k}{n} = 0$*".*

Rank	MathAMR	RawText
1	Prove that $\sum\limits_{n=1}^{N} \cos(2\pi n/N) = 0$	How to prove $\sum_{k=1}^{n} \cos(\frac{2\pi k}{n}) = 0$ for any $n > 1$?
2	How to prove $\sum_{k=1}^{n} \cos(\frac{2\pi k}{n}) = 0$ for any $n > 1$?	How to prove that $\sum_{k=0}^{n-1} \cos\left(\frac{2\pi k}{n} + \phi\right) = 0$
3	Proving that $\sum_{x=0}^{n-1} \cos\left(k + x\frac{2\pi}{n}\right) = \sum_{x=0}^{n-1} \sin\left(k + x\frac{2\pi}{n}\right) = 0.$	Prove that $\sum\limits_{n=1}^{N} \cos(2\pi n/N) = 0$
4	$\sum_{k=0}^{n-1} \cos\left(\frac{2\pi k}{n}\right) = 0 = \sum_{k=0}^{n-1} \sin\left(\frac{2\pi k}{n}\right)$	Understanding a step in applying deMoivre's Theorem to $\sum_{k=0}^{n} \cos(k\theta)$
5	$\sum \cos$ when angles are in arithmetic progression	$\sum \cos$ when angles are in arithmetic progression

example, this happens for topic A.328 with the title:

$$\text{“Proving } \sum_{k=1}^{n} \cos\frac{2\pi k}{n} = 0\text{”}$$

Table 6 shows the titles of the top-5 similar questions for that topic. As seen in this table, MathAMR representations retrieved two similar questions (at ranks 3 and 4) that have similar formulas, whereas raw text failed to retrieve those formulas in its top-5 results. The P′@10 on that topic for the QASim model using MathAMR was 0.5, whereas with raw text it was 0.1.

5. Task 3: Open Domain Question Answering

Open domain question answering is a new pilot task introduced in ARQMath-3. The goal of this task is to provide answers to the math questions in any way, based on any sources. The Task 3 topics are the same as those used for Task 1. Our team created four runs for this task, each having the same architecture. All our four runs use extractive summarization, where a subset of sentences are chosen from the answer to form a summary of the answer. This subset hopefully contains the important section of the answer. The organizers provided one run using GPT-3 [40] from OpenAI as the baseline system.

We made two runs, "SVM-Rank" and "QQ-QA-AMR" from those two Task 1 runs by simply truncating the result set for each topic after the first post, then applying one of two BERT-based summarizers to the top-1 answer for each question for each run. For summarizers, we used one that we call BERT that uses 'bert-large-uncased' [41] and a second called Sentence-BERT (SBERT) [4] that is implemented using an available python library,[12] with its 'paraphrase-MiniLM-L6-v2' model.

Both summarizers split answers into sentences, and then embed each sentence using BERT or Sentence-BERT. Sentence vectors are then clustered into k groups using k-means clustering, after which the k sentences closest to each cluster centroid are returned unaltered, in-order, in the generated response. We set k to 3, meaning that all sentences for posts with up to three sentences are returned, and exactly three sentences are returned for posts with four or more sentences.

Results. The results of our Task 3 are reported in Table 7. As seen in this table, our results are not comparable to the baseline system. The highest Average Relevance (AR) and P@1 are achieved using the Sentence-BERT summarizer to summarize the top-1 answered retrieved with the SVM-Rank model for Task 1. Answers extracted by BERT and Sentence-BERT from top-1

[12]https://pypi.org/project/bert-extractive-summarizer/

Table 7
DPRL Runs for Open Domain Question Answering (Task 3) on ARQMath-3 (78) topics. For this task, we submitted the top hit from each run (i.e., a MathSE answer post) that was then passed through a BERT-based summarizer. All runs use both math and text to retrieve answers. GPT-3 is provided by the organizers as the baseline system.

Open Domain QA Run	Avg. Rel.	P@1
GPT-3 (Baseline)	1.346	0.500
SBERT-SVMRank	0.462	0.154
BERT-SVMRank	0.449	0.154
SBER-QQ-AMR	0.423	0.128
BERT-QQ-AMR	0.385	0.103

Table 8
Sentence-BERT vs. BERT extracted summaries on the first answer retrieved by QQ-QA-AMR model for topic A.325.

Topic Title	Find consecutive composite numbers (A.325)
BERT	"No, we can find consecutive composites that are not of this form. The point of $n!$ is just that it is a ""very divisible number""."
SBERT	No, we can find consecutive composites that are not of this form. For example the numbers $n!^2 + 2, n!^2 + 4 \cdots + n!^2 + n$ or $n!^3 + 2, n!^3 + 3 \ldots n!^3 + 2$. Also $kn! + 2, kn! + 3 \ldots k!n + n$ works for all $k > 0 \in \mathbb{Z}$ You can also get a smaller examples if instead of using $n!$ we use the least common multiple of the numbers between 1 and n.

SVM-Rank answers were only different for 13 of the 78 assessed topics. However, P@1 was identical for each topic.

For models using AMR, p@1 differed beteen BERT and Sentence-BERT for 3 topics, although 19 topics had different sentences extracted. In two of those three cases, Sentence-BERT included examples in the extracted answer, resulting in a higher P@1 in both cases compared to BERT. Table 8 shows the answers extracted for Topic A.325, which has the title "Find consecutive composite numbers", with the BERT and Sentence-BERT summarizers, where the answers are highly relevant and low relevant, respectively. The only case in hich P@1 for the Sentence-BERT summarizer was lower than that of the BERT summarizer with the "QQ-QA-AMR" model was a case in which the answer extracted by Sentence-BERT was not rendered correctly, and thus was not assessed, which in Task 3 was scored as non-relevant.

6. Conclusion

This paper has described the DPRL runs for the ARQMath lab at CLEF 2022. Five runs were submitted for the Formula Retrieval task. These runs used isolated or contextual formula search models. Our models with tree-edit distance ranking had the highest effectiveness among the automatic runs. For the Answer Retrieval task, five runs were submitted using raw text or new unified representation of text and math that we call MathAMR. While for model provided better effectiveness compared to the baseline model we were aiming to automate, there results were less effective compared to our participating teams. For the new Open Domain Question Answering task, four runs were submitted, each of which summarizes the first result from an Answer Retrieval run using extractive summarization on models with MathAMR and raw text. The models using raw text were more effective.

Acknowledgments

We thank Heng Ji and Kevin Knight for helpful discussions about AMR and multimodal text representations. This material is based upon work supported by the Alfred P. Sloan Foundation under Grant No. G-2017-9827 and the National Science Foundation (USA) under Grant No. IIS-1717997.

References

[1] B. Mansouri, A. Agarwal, D. W. Oard, R. Zanibbi, Advancing Math-Aware Search: The ARQMath-3 Lab at CLEF 2022, in: European Conference on Information Retrieval, Springer, 2022.

[2] R. Zanibbi, D. W. Oard, A. Agarwal, B. Mansouri, Overview of ARQMath 2020: CLEF Lab on Answer Retrieval for Questions on Math, in: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer, 2020.

[3] B. Mansouri, A. Agarwal, D. Oard, R. Zanibbi, Advancing Math-Aware Search: The ARQMath-2 lab at CLEF 2021, in: European Conference on Information Retrieval, Springer, 2021.

[4] N. Reimers, I. Gurevych, Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, 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.

[5] L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, N. Schneider, Abstract Meaning Representation for Sembanking, in: Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, 2013.

[6] R. Zanibbi, D. Blostein, Recognition and retrieval of mathematical expressions, Int. J. Document Anal. Recognit. 15 (2012) 331–357. URL: https://doi.org/10.1007/s10032-011-0174-4. doi:10.1007/s10032-011-0174-4.

[7] K. Davila, R. Zanibbi, A. Kane, F. W. Tompa, Tangent-3 at the NTCIR-12 MathIR task, in: N. Kando, T. Sakai, M. Sanderson (Eds.), Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies, National Center of Sciences, Tokyo, Japan, June 7-10, 2016, National Institute of Informatics (NII), 2016. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings12/pdf/ntcir/MathIR/06-NTCIR12-MathIR-DavilaK.pdf.

[8] B. Mansouri, S. Rohatgi, D. W. Oard, J. Wu, C. L. Giles, R. Zanibbi, Tangent-CFT: An

Embedding Model for Mathematical Formulas, in: Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval, 2019.

[9] P. R. Kingsbury, M. Palmer, From TreeBank to PropBank., in: LREC, 2002.

[10] F. Liu, J. Flanigan, S. Thomson, N. Sadeh, N. A. Smith, Toward Abstractive Summarization Using Semantic Representations, in: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2015.

[11] K. Liao, L. Lebanoff, F. Liu, Abstract Meaning Representation for Multi-Document Summarization, in: Proceedings of the 27th International Conference on Computational Linguistics, 2018.

[12] P. Kapanipathi, I. Abdelaziz, S. Ravishankar, S. Roukos, A. Gray, R. F. Astudillo, M. Chang, C. Cornelio, S. Dana, A. Fokoue-Nkoutche, et al., Leveraging Abstract Meaning Representation for Knowledge Base Question Answering, in: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, 2021.

[13] W. Xu, H. Zhang, D. Cai, W. Lam, Dynamic Semantic Graph Construction and Reasoning for Explainable Multi-hop Science Question Answering, in: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, 2021.

[14] S. Garg, A. Galstyan, U. Hermjakob, D. Marcu, Extracting Biomolecular Interactions Using Semantic Parsing of Biomedical Text, in: Thirtieth AAAI Conference on Artificial Intelligence, 2016.

[15] Z. Zhang, H. Ji, Abstract Meaning Representation Guided Graph Encoding and Decoding for Joint Information Extraction, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2021.

[16] J. Flanigan, S. Thomson, J. G. Carbonell, C. Dyer, N. A. Smith, A Discriminative Graph-based Parser for the Abstract Meaning Representation, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2014.

[17] C. Wang, N. Xue, S. Pradhan, A transition-based algorithm for AMR parsing, in: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2015.

[18] M. Bevilacqua, R. Blloshmi, R. Navigli, One SPRING to Rule them Both: Symmetric AMR Semantic Parsing and Generation without a Complex Pipeline, in: Proceedings of AAAI, 2021.

[19] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, L. Zettlemoyer, BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020.

[20] K. Knight, B. Badarau, L. Baranescu, C. Bonial, M. Bardocz, K. Griffitt, U. Hermjakob, D. Marcu, M. Palmer, T. O'Gorman, et al., Abstract Meaning Representation (AMR) Annotation Release 3.0 (2021). URL: https://catalog.ldc.upenn.edu/LDC2020T02.

[21] K. Davila, R. Zanibbi, Layout and Semantics: Combining Representations for Mathematical Formula Search, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2017.

[22] R. Zanibbi, A. Aizawa, M. Kohlhase, I. Ounis, G. Topic, K. Davila, NTCIR-12 MathIR Task

Overview., in: NTCIR, 2016.
[23] J. May, J. Priyadarshi, Semeval-2017 Task 9: Abstract Meaning Representation Parsing and Generation, in: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), 2017.
[24] B. Mansouri, R. Zanibbi, D. W. Oard, A. Agarwal, Overview of ARQMath-2 (2021): Second CLEF Lab on Answer Retrieval for Questions on Math, in: International Conference of the Cross-Language Evaluation Forum for European Languages, LNCS, Springer, 2021.
[25] B. Mansouri, D. W. Oard, R. Zanibbi, DPRL Systems in the CLEF 2021 ARQMath Lab: Sentence-BERT for Answer Retrieval, Learning-to-Rank for Formula Retrieval (2021).
[26] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching Word Vectors with Subword Information, Transactions of the Association for Computational Linguistics 5 (2017).
[27] J. Johnson, M. Douze, H. Jégou, Billion-Scale Similarity Search with GPUs, IEEE Transactions on Big Data (2019).
[28] G. V. Cormack, C. L. Clarke, S. Buettcher, Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods, in: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2009.
[29] B. Mansouri, R. Zanibbi, D. W. Oard, Learning to Rank for Mathematical Formula, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021.
[30] T. Joachims, Training Linear SVMs in Linear Time, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006.
[31] W. Zhong, X. Zhang, J. Xin, J. Lin, R. Zanibbi, Approach Zero and Anserini at the CLEF-2021 ARQMath Track: Applying Substructure Search and BM25 on Operator Tree Path Tokens, CLEF, 2021.
[32] W. Zhong, J. Lin, PYA0: A Python Toolkit for Accessible Math-Aware Search, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021.
[33] Y. K. Ng, D. J. Fraser, B. Kassaie, F. W. Tompa, Dowsing for Math Answers, in: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer, 2021.
[34] D. Fraser, A. Kane, F. W. Tompa, Choosing Math Features for BM25 Ranking with Tangent-L, in: Proceedings of the ACM Symposium on Document Engineering 2018, 2018.
[35] K. Krstovski, D. M. Blei, Equation embeddings, arXiv preprint arXiv:1803.09123 (2018).
[36] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed Representations of Words and Phrases and their Compositionality, Advances in Neural Information Processing Systems (2013).
[37] S. Chopra, R. Hadsell, Y. LeCun, Learning a Similarity Metric Discriminatively, with Application to Face Verification, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), IEEE, 2005.
[38] M. Henderson, R. Al-Rfou, B. Strope, Y.-H. Sung, L. Lukács, R. Guo, S. Kumar, B. Miklos, R. Kurzweil, Efficient Natural Language Response Suggestion for Smart Reply, arXiv preprint arXiv:1705.00652 (2017).
[39] 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.

[40] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language Models are Few-Shot Learners, 2020.

[41] D. Miller, Leveraging BERT for Extractive Text Summarization on Lectures, arXiv preprint arXiv:1906.04165 (2019).