DPRL Systems in the CLEF 2020 ARQMath Lab

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

[1] Rochester Institute of Technology (USA)
{bm3302,rxzvcs}@rit.edu
[2] University of Maryland, College Park (USA)
oard@umd.edu

Abstract. This paper describes the participation of the Document and Pattern Recognition Lab from the Rochester Institute of Technology in the CLEF 2020 ARQMath lab. There are two tasks defined for ARQMath: (1) Question Answering, and (2) Formula Retrieval. Four runs were submitted for Task 1 using systems that take advantage of text and formula embeddings. For Task 2, three runs were submitted: one uses only formula embedding, another uses formula and text embeddings, and the final one uses formula embedding followed by re-ranking results by tree-edit distance. The Task 2 runs yielded strong results, the Task 1 results were less competitive.

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

1 Introduction

The ARQMath lab at CLEF 2020 has two tasks [3]. In the first task, called Answer Retrieval, given the mathematical questions, the participants were asked to return a set of relevant answers to each question. Formula Retrieval is the second task, where given a mathematical formula as the query, the participants were asked to retrieve a set of relevant formulas.

The Document and Pattern Recognition Lab (DPRL) from the Rochester Institute of Technology (RIT, USA) participated in both tasks. For Task 1, we used a text and formula embedding approach, where each question and answer are represented by a vector of float numbers. We submitted 4 runs for this task.

For the second task, we submitted 3 runs, all based on a mathematical formula embedding model, Tangent-CFT [4]. This embedding model uses both Symbol Layout Tree (SLT) and Operator Tree (OPT) formula representations. The SLT representation encodes the appearance of the formula, while the OPT representation encodes its operator syntax (i.e., the operator tree). Figure 1

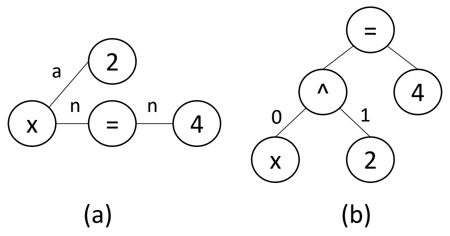


Fig. 1. Formula $x^2 = 4$ represented as (a) Symbol Layout Tree and (b) Operator Tree.

shows the SLT and OPT representations for the formula $x^2 = 4$. In the SLT representation, the edge labels show the spatial relationship between the formula elements. For instance, the edge label 'a' shows that the number '2' is located above the variable 'x', while the edge label 'n' shows operator '=' is located next after 'x'. In the OPT representation, the edge labels for non-commutative operators indicate argument position. For more details, refer to Mansouri et al. [4].

While our runs results for Task 1 were not as good as for some other participating teams, our Task 2 results were noteworthy for including a system that was statistically indistinguishable from the baseline (which was a strong baseline relative to submitted systems). In this paper, we first introduce our runs for Task 2 because some techniques from Task 2 were used in our Task 1 systems.

2 Formula Retrieval (Task 2)

For Task 2, we built three systems and submitted one run from each. All three systems use formula embeddings produced from strings generated after parsing formulas with Tangent-S [2], which produces both SLT and OPT representations. Tangent-S and Tangent-CFT are publicly available.[3]

In Tangent-S, candidate formula SLTs and OPTs are first retrieved independently, using tuples representing the relative paths between pairs of symbols. The top-2000 results from the separate OPT and SLT result lists are reranked after aligning candidates to the query formula, using an ordered list based on three features (the second and third features are used to break ties when ranking): 1) a unified structural matching score (Maximum Subtree Similarity (MSS)), 2) precision for candidate symbols matched *only* by unification (treated as a penalty), and 3) symbol recall based on exact matching of candidate and query symbols. The reranked candidates are then rescored again using a linear combination of the three-element score vectors associated with each retrieved formula's SLT and OPT. This combines the SLT and OPT score vectors for each candidate formula into a single rank score. The baseline uses a weight vector learned on the NTCIR-12 Wikipedia Formula Browsing task test collection [2].

[3] `https://www.cs.rit.edu/~dprl/software.html`

In the following, we describe our formula retrieval models, all of which use the Tangent-S SLT and OPT tree encodings. Note that our systems do not use any of the retrieval models of Tangent-S: only the SLT and OPT string encodings, which represent depth-first traversals on OPTs and SLTs.

2.1 Tangent-CFT

Tangent-CFT [4] is an embedding model for mathematical formulas. This model uses both SLT and OPT representations of formulas to consider both the appearance and the syntax of formulas. Summarizing the indexing steps of Tangent-CFT (for the complete description, please see the paper):

– **Tuple Extraction**: Using Tangent-S, LaTeX formulas are converted to Presentation and Content MathML, from which the internal SLT and OPT formula representations are produced. Depth-first traversals of these OPT and SLT trees are used to produce strings consisting of a sequence of tuples. Each tuple has three elements (N_i, N_j, E). The N_i shows the node value which is in the form of *Type!Value*, where type can take values such as Variable (V) or Number(N) and value shows the variable name or the numeric value. E shows a sequence of edge labels connecting the two nodes.
– **Tuple Encoding**: Each tuple is then tokenized and enumerated. The tokenization is based on type, value. For example the tuple (V!x, N!2, a), which represents x^2, will be tokenized to {'V', 'x', 'N', '2', 'a' }.
– **Training Embedding Models with fastText**: After tokenizing each tuple, each token is considered as a character and the whole tuple as a word. The resulting sequence of tuples is considered as a sentence in which the words (tuples) appear. As shown in Mansouri et al. [5], mathematical formulas can be rather infrequent, so the fastText [1] n-gram embedding model is used to get vector representations for each tuple. The formula vector representation is then obtained by averaging the vector representations for the individual tuples.
– **Combining the Vector Representations**: Tangent-CFT uses three representations of the formulas: SLT, OPT, and SLT-Type. In SLT-Type, each node represents only the type, and not the value. This can help with the partial matching of formulas especially for the cases where the variable names are not critical. For each representation, a separate embedding model is trained and a vector representation is obtained. The final formula vector is obtained by adding the 3 vectors together.

We trained Tangent-CFT on Math Stack Exchange formulas, using the default parameters which were obtained by training the model on the NTCIR-12 Wikipedia Formula Browsing task. SLT and OPT representations are sometimes not available for specific formulas, and Tangent-CFT will only retrieve formulas for which at least one of those representations is available. For this task, if one of the formula representations is not available, a vector zeros is used for that embedding.

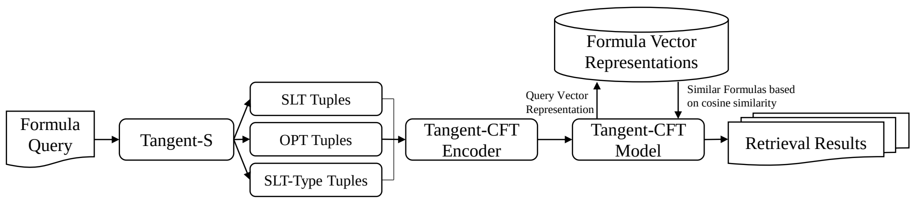


Fig. 2. Overview of Tangent-CFT. The set of SLT, OPT and SLT-type tuples are extracted from the formula query using Tangent-S. These tuples are then encoded using the Tangent-CFT encoder, and compared against vector representations for formulas in the collection using cosine similarity.

2.2 Tangent-CFT-Plus

As described in [3], to decide the relevance, the annotators might consider the question (context) in which the formula appeared in. Tangent-CFT only considers the structural similarity of formulas when embedding them. Tangent-CFT-Plus is an extension to Tangent-CFT embedding model in which each formula has two vector representations:

– *Formula Vector*: Vector representation obtained by Tangent-CFT. The vector size is 300.
– *Text Vector*: Vector representation obtained by considering formula as a word and training the fastText model on the surrounding words of the formula. We used a vector size of 100 which is the fastText default value.

Each formula is represented as the concatenation of Formula and Text vectors.

2.3 Tangent-CFT with Tree Edit Distance

Experiment results on the NTCIR-12 Wikipedia Formula Browsing task test collection [7] showed Tangent-CFT to do better on partial matching compared to full matching. To address that issue, we extended Tangent-CFT by re-ranking the retrieval results based on tree-edit distance.

Tree-edit distance (TED) is the minimum cost of converting one tree to another. Three edit operations were considered: insertion, deletion and substitution. Each of these operations can have different weights when calculating the tree-edit distance cost. The ranking score for each retrieved formula is calculated as:

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

We tuned the weights for each edit operations, over the range [0, 1] with step size 0.05, on the NTCIR-12 test collection using leave-one-out cross-validation. In the NTCIR-12 Formula Browsing Task test collection [7] there are 20 formula queries without wildcards. We learn operation weights for each query independently, and

Table 1. DPRL runs for Task 2, results averaged over 45 topics and computed over deduplicated ranked lists of visually distinct formulas. For MAP′ and P@10, relevance was thresholded H+M (High or Medium) binarization.

			Evaluation Measures		
Run	Data	Primary	nDCG′	MAP′	P@10
Baseline					
Tangent-S	Math	(✓)	**(0.506)**	**(0.288)**	**(0.478)**
DPRL					
Tangent-CFTED	Math	✓	**0.420**	**0.258**	**0.502**
Tangent-CFT	Math		0.392	0.219	0.396
Tangent-CFT-Plus	Both		0.135	0.047	0.207

report results as averages over the 20 queries using weights learned for each query. Our goal was to maximize the harmonic mean bpref score, which balances full and partial bpref scores. As there are two tree representations for formulas (OPT and SLT), the re-ranking is done on each representation separately, and then the results were linearly combined. The linear combination weight is calculated in a similar way as the edit operation weights using leave-one-out cross-validation.

To use the tree edit distance for formula retrieval, the following steps are taken:

1. Retrieve the top-1000 results with Tangent-CFT model.
2. Re-rank the results with tree-edit distance using SLT and OPT representations. For SLTs, the edit operations cost were set 0.85, 0.15, and 0.54 for deletion, substitution and insertion operations respectively and for OPTs they were set to 0.28, 0.185, 0.225.
3. The re-ranked results with OPT and SLT representations are linearly combined as:

$$Score_q(f) = \alpha \cdot SLT\text{-}Score_q(f) + (1-\alpha) \cdot OPT\text{-}Score_q(f) \qquad (2)$$

   with $\alpha = 0.95$. The tree-edit distance values are calculated using a publicly-available Python package for the APTED algorithm [6].

2.4 Results

Table 1 shows the results of our runs and for the baseline (Tangent-S) (Taken from the lab overview paper [8]). For all three evaluation measures, the apparent differences between Tangent-CFTED and Tangent-S (which, for P@10, are 5% relatively higher for Tangent-CFTED) are not statistically significant by a two-tailed paired t-test (at $p < 0.05$).

We analyzed our runs against the baseline and against each other. Our first analysis was comparing the Tangent-S system against Tangent-CFTED to investigate how many relevant formulas are retrieved by both systems, and how many are retrieved by one system and not the other. We considered different relevance degrees, first treating only high as relevant, then high and medium as relevant, and finally all the three levels as relevant. Figure 3 shows the Venn diagram of retrieval results between Tangent-S (the right circles with dashed blue outline) and Tangent-CFTED (the left circles with grey background).

The largest difference in P@10 values between Tangent-CFTED and Tangent-S was 0.5 for the formula

$$(x+y)^k \geq x^k + y^k \tag{3}$$

(Topic B.48). Half of the top-10 formulas retrieved by Tangent-CFTED are annotated as highly relevant and the other half are annotated as medium. Tangent-S retrieved 4 formulas annotated as high/medium, and 5 as low. Formulas such as $(x+y)^p \equiv x^p + y^p \pmod{p}$ with a low relevance rating get a higher rank in Tangent-S, while formulas such as $(x+y)^a \geq x^a + y^a$ and $(x+y)^s \geq x^s + y^s$ that are highly relevant get lower ranks, and are not in the top-10 unique formulas. However, these last two formulas appear at ranks 6 and 3 in the Tangent-CFTED search results, respectively.

Another topic where Tangent-CFTED did better than the baseline is the formula

$$\sum_{k=0}^{n} \binom{n}{k} k \tag{4}$$

(Topic B.4) where P@10 for the baseline was 0, but 0.4 for Tangent-CFTED. The top-5 unique formulas retrieved by Tangent-S and Tangent-CFTED for this query are shown in Table 2.

For 8 Task 2 topics, the P@10 for Tangent-CFTED was 0 (i.e., no formulas with medium or high relevance ratings were found), and for 4 of these topics, no relevant formulas (at any relevance level) were retrieved in top-10 results. The highest P@10 achieved by Tangent-S for these 8 topics was 0.3. Some of these topics were more complex than other queries, and had fewer relevant hits. Providing an example, for formula

$$\exists p \ \big(p \text{ is prime } \rightarrow \forall x \ (x \text{ is prime})\big) \tag{5}$$

(Topic B.56) there are only 5 formulas with a relevance rating of medium or high in the collection. For this formula, the Tangent-CFT model fails to retrieve

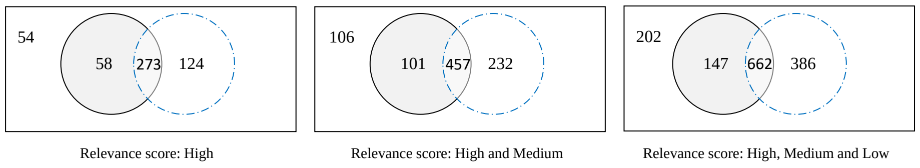


Fig. 3. Venn Diagrams for all Visually Distinct Relevant formulas in Task 2 (45 Topics). Each diagram indicates how many relevant formulas were retrieved by Tangent-S (blue circles) and Tangent-CFTED (grey circles) within their top-1000 results. Relevant formulas missed by both systems are at top-left in each diagram. In total, there are 1,397 relevant formulas in the collection (509 high, 387 medium, and 501 low).

Table 2. Top-5 unique formulas for query $\sum_{k=0}^{n}\binom{n}{k}k$

	Tangent-S		Tangent-CFTED	
Rank	Formula	Relevance	Formula	Relevance
1.	$\sum\limits_{k=0}^{n}\binom{n}{k}k2^{n-k}$	Non-Relevant	$\sum_{k=0}^{n}\binom{n}{k}k$	High
2.	$\sum_{k=0}^{n}\binom{n}{k}k2^{n-k}$	Non-Relevant	$\sum_{k=0}^{n}\binom{n}{k}M_k$	Non-Relevant
3.	$\sum_{k=0}^{n}\binom{n}{k}k!\,D_{n-k}$	Non-Relevant	$\sum_{k=0}^{n}\binom{n}{k}x^k$	Medium
4.	$\sum_{k=0}^{n}\binom{n}{k}k^p$	Low	$\sum_{k=0}^{n}\binom{n}{k}x^k$	Medium
5.	$\sum_{k=0}^{n}\binom{n}{k}x^k(1-x)^{n-k}k$	Non-Relevant	$\sum_{k=0}^{n}\binom{n}{k}k^p$	Low

any relevant formulas in its top-10 results, and Tangent-CFTED has only one relevant formula, $\exists p(p \text{ is prime and } p^2 \mid n)$, which has a low relevance rating at rank 5. It is possible that the larger depth used for re-ranking by Tangent-S (i.e., top-2000, vs. top-1000 for Tangent-CFTED) may be contributing to the difference in performance for these topics.

Comparing Tangent-CFT-Plus with our other two runs, there were four queries for which Tangent-CFT-Plus had higher P@10. One of these cases was the query

$$\frac{df}{dx} = f(x+1) \tag{6}$$

where the P@10 for our other two runs was 0, but for Tangent-CFT-Plus was 0.4. Tangent-CFT only retrieves 6 relevant formulas for this query, and Tangent-CFTED fails to re-rank them better than Tangent-CFT. The Tangent-CFT-Plus model takes advantage of surrounding text. Terms such "differential equations" appear frequently around the related formulas. Therefore, Tangent-CFT-Plus gives higher ranks to formulas such as $\frac{df(x)}{dx} = f(x)$, which was the first formula retrieved by this system for the query $\frac{df}{dx} = f(x+1)$.

Performance. Table 3 shows the minimum, maximum, mean and standard deviation of retrieval times for our three runs in task 2. Our runs were done on a machine with 4 NVIDIA GeForce GPU cards with 11GB memory, a 528 GB Ram, with Intel(R) Xeon(R) CPU E5-2667 v4 @ 3.20GHz (32 cores). For calculating the cosine similarity of vectors we used Pytorch framework. As can be seen, the retrieval time for Tangent-CFT is lower than Tangent-CFT-PLUS due to smaller size of vectors. As Tangent-CFTED has an additional re-ranking step, its retrieval time is longer than Tangent-CFT.

Post-Task Experiments. After the submission of our runs, we found an error in our run, Tangent-CFT-Plus which used partial data for retrieval: results were only returned from answers written between 2010 and 2015. Therefore, we ran our model again, after assessment was complete (i.e., after pooling). With

Table 3. Retrieval Times in Seconds for Task 2.

	Run Time (Seconds)		
System	Min (Topic)	Max (Topic)	(Avg, StDev)
Tangent-CFT	0.653 (B.4)	0.669 (B.41)	(0.658, 0.004)
Tangent-CFT-PLUS	0.848 (B.69)	0.857 (B.54)	(0.851, 0.002)
Tangent-CFTED	0.672 (B.3)	7.052 (B.26)	(1.752, 1.622)

correct use of the dataset, Tangent-CFT-Plus could achieved the nDCG′ value of 0.305, mAP′ of 0.155 and P@10 of 0.342.

In another experiment, we looked at the Tangent-CFTED model, to check the effect of α value used to combine SLT and OPT results. This value α was set to 0.95, which was learnt on NTCIR-12 dataset and leaned highly toward SLT representation. We conducted another experiment by setting the α value to 0.5, weighting SLTs and OPTs equally. With this setting, the effectiveness measures were nDCG′ of 0.424, mAP′ of value of 0.253 and P@10 of 0.488. This produces small changes in nDCG′ and mAP′ scores ($\leq 0.5\%$), and a decrease in P@10 of about two percent compared to using Tangent-CFTED using $\alpha = 0.95$, and so giving greater weight to OPTs did not improve the performance of the model.

Note that the formula relevance definition in NTCIR-12 was different than ARQMath, and therefore in future work, we will do parameter tuning using the ARQMath dataset.

3 Answer Retrieval (Task 1)

In this section we present our system design, runs, and results for Task 1.

3.1 System Design

For Task 1, our approach was to use text and formula embeddings to find relevant answers for each topic. Our runs differ in how the embedding models are used for retrieval. Each post (question or answer) in our models is represented as a vector with dimension 400: the first 100 elements being the text embedding of the post and the next 300 being the formula embedding.

We trained fastText [1] to get text embeddings for each post. To tune the fastText parameters, we used 1,003 questions published in the year 2018 that had duplicate or related questions as our training questions. The duplicate and related questions are determined by the Math Stack Exchange moderators are data that was provided by the ARQMath organizers. The answers given to the duplicate or related questions, were considered as the relevant answers for the training questions. For each training question, all the answers from duplicate and related questions were collected and sorted based on scores. Then, the first quarter of the answers was considered as high, the second quarter medium, and the third quarter as an answer with low relevance level.

Using the training questions, we tuned the fastText parameters by considering our first proposed retrieval system. Our goal is to train a model that maximizes the precision at 10 for the training questions. After hyper-parameter tuning, our final setting for the fastText model was to use a skip-gram model with negative sampling (15 negative samples), with a context window size of 5, n-grams of lengths between 2 and 5 characters, and a vector size of 100. The fastText model was trained on all the questions, answers, and comments in the collection with 10 epochs. As fastText provides embeddings for the words, we averaged all the vector representations of the words inside a post to get a post embeddings. For the questions, the vector representation is the average of vector representations of the title and the body of the questions.

The second part of a post vector represents the formula embedding. We trained the formula embedding model, Tangent-CFT [4] with the same parameters that this model used for the NTCIR-12 [7] formula browsing task. The model is described in previous section. As each post can have multiple formulas, the formula vector is the unweighted average of all the formulas vectors in three of our runs. In one run (DPRL3), we used the weighted average of formulas, using the length of formula as the weight.

3.2 Submitted Runs

Here we describe our four runs for Task 1, which are summarized in Table 4.

DPRL1. This was our primary run, in which the answers in the collection are represented as the summation of two vector representations: one for the answer and one for the question to which the answer was given. Both the answer and question vector representations are obtained as described before, by concatenating the text and formula vectors. For each topic, the cosine similarities of the topic's vector and all the answers' vectors in the collection are computed and used to rank the answers.

DPRL2. This alternate run is similar to our primary run, with the difference being that the answer vector is just the vector representation of the answer. Compared to our primary run, the question to which the answer was given is ignored. For each topic, the cosine similarity of the topic vector and answer vector is the similarity score.

DPRL3. This alternate run is again similar to our primary run, with the difference being that the formula part of the vector is calculated differently. As mentioned earlier, there could be multiple formulas in a post. In this run, instead of using the unweighted average of vector representations of formulas, we used the weighted average of vector representations by considering the length of each formula as the weight. Our assumption was that the bigger formulas are more important. We used the length of the LaTeX string representation of formulas for this. As with the primary run, the similarity of topic and answer is the cosine similarity of their vector representations.

DPRL4. This alternate run first finds questions that are similar to the topic and then ranks the answers to those similar questions. To perform this ranking, after finding similar questions to the topic, we compute the cosine similarity score of the topic vector and the vector for the similar question with which each answer is associated, and we then multiply that by the score assigned to that

Table 4. Summary of DPRL Submissions for ARQMath Answer Retrieval Task. Each Answer has a vector representation of size 400, where the first 100 elements belong to the text embedding, and the final 300 elements are the formula embedding.

	Embedding Vector (400 elements)	
Run	Text (100)	Formula (300)
DPRL1	Answer + Question	Tangent-CFT (unweighted average)
DPRL2	Answer	Tangent-CFT (unweighted average)
DPRL3	Answer + Question	Tangent-CFT (weighted average)
DPRL4	Question	Tangent-CFT (unweighted average)

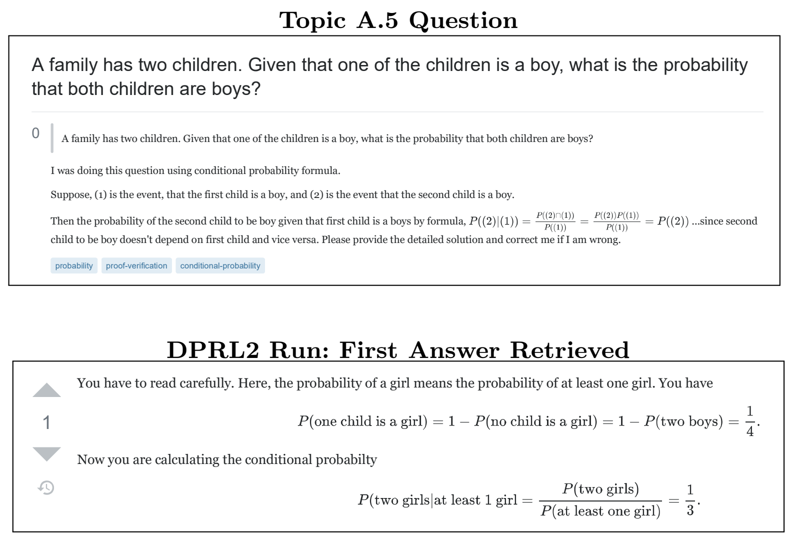
Table 5. Task 1 (CQA) results, averaged over 77 topics. **P** indicates a primary run, **M** indicates a manual run, and **(✓)** indicates a baseline pooled at the primary run depth. For Precision@10 and MAP, H+M (High or Medium) binarization was used. The best baseline results are in parentheses. * indicates that one baseline did not contribute to judgment pools.

		Run Type		Evaluation Measures		
Run	Data	P	M	nDCG′	MAP′	P@10
Baselines						
Linked MSE posts	n/a	(✓)		(**0.303**)	(**0.210**)	(**0.417**)
*Approach0**	Both		✓	0.250	0.100	0.062
TF-IDF + Tangent-S	Both	(✓)		0.248	0.047	0.073
TF-IDF	Text	(✓)		0.204	0.049	0.073
Tangent-S	Math	(✓)		0.158	0.033	0.051
DPRL4	Both			0.060	0.015	0.020
DPRL2	Both			0.054	0.015	0.029
DPRL1	Both	✓		0.051	0.015	0.026
DPRL3	Both			0.036	0.007	0.016

answer by Math Stack Exchange users. We then sort the answers in decreasing order of that product. We then perform a reranking step in which we make use of the Tags assigned to the Topic question and to the question posts that had been found, which had been provided by the people asking the questions. Answers associated with found questions that have at least one Tag that exactly matches a Tag in the query question are moved ahead of all other answer posts (with their relative order maintained).

3.3 Results

For the answer retrieval task, three evaluation measures were considered: nDCG′, MAP′ and P@10. Our results for this task were not promising. Table 5 shows the baselines and our runs results for this task (Taken from the lab overview paper [8]). Our runs mostly did better for topics in the category "Computation" than for topic in categories "Concept" or "Proof". For instance, all the topics for which our DPRL2 run (which had our highest P@10), was able to find relevant



Topic A.5 Question

A family has two children. Given that one of the children is a boy, what is the probability that both children are boys?

0

A family has two children. Given that one of the children is a boy, what is the probability that both children are boys?

I was doing this question using conditional probability formula.

Suppose, (1) is the event, that the first child is a boy, and (2) is the event that the second child is a boy.

Then the probability of the second child to be boy given that first child is a boys by formula, $P((2)|(1)) = \frac{P((2)\cap(1))}{P((1))} = \frac{P((2))P((1))}{P((1))} = P((2))$ ...since second child to be boy doesn't depend on first child and vice versa. Please provide the detailed solution and correct me if I am wrong.

DPRL2 Run: First Answer Retrieved

1

You have to read carefully. Here, the probability of a girl means the probability of at least one girl. You have

$$P(\text{one child is a girl}) = 1 - P(\text{no child is a girl}) = 1 - P(\text{two boys}) = \frac{1}{4}.$$

Now you are calculating the conditional probabilty

$$P(\text{two girls}|\text{at least 1 girl} = \frac{P(\text{two girls})}{P(\text{at least one girl})} = \frac{1}{3}.$$

Fig. 4. Topic A.5 and the top answer retrieved by DPRL2 run. The highest P@10 was achieved for this topic (0.9)

Table 6. Retrieval Times in Seconds for Task 1.

	Run Times (in seconds)		
System	Min (Topic)	Max (Topic)	(Avg, StDev)
DPRL 1	0.055 (A.84)	0.061 (A.41)	(0.056, 0.0001)
DPRL 2	0.056 (A.54)	0.057 (A.49)	(0.056, 0.0006)
DPRL 3	0.056 (A.1)	0.057 (A.18)	(0.056, 0.0001)
DPRL 4	0.901 (A.26)	1.143 (A.30)	(0.985, 0.0564)

answers were in the category "Computation". For two topics, the DPRL2 run had P@10 higher than 0.5 (note that P@10 was calculated by considering the relevance degrees of high and medium as relevant). Figure 4 shows topic A.5 for which DPRL2 achieved P@10 of 0.9, along with the first answer retrieved for this topic, which is highly relevant.

Another observation we had was that the DPRL4 run, which has the highest nDCG′ value among our runs, is able to find relevant answers for categories of Proof and Concept. For instance, for topic A.38 (in the Concept category, with difficulty Hard), this run achieved the P@10 of 0.4.

As mentioned earlier, we did the hyper-parameter tuning for our proposed models based on MSE scores. Comparing these scores, with the annotation scores, there is no correlation; that is to say, not all the relevant answers have high score and not all the not relevant ones have low score. Therefore, in our future work, we will do the hyper-parameter tuning using the annotated data.

Performance. Table 6 shows the minimum, maximum and average retrieval time for each of proposed models in task 1. We used the same machine as for task 2. As can be seen, all our three first runs have similar retrieval times as they just use cosine similarity of vector representations of documents. However, for our last run, we have an extra re-ranking step which results in longer retrieval times.

4 Conclusion

In this paper, we described the DPRL runs for ARQMath lab at CLEF 2020. For Formula Retrieval (Task 2), the DPRL submitted 3 runs. Our run that uses formula embedding and then re-ranks by tree-edit distance achieved the highest P@10 value and the second-best nDCG$'$ after the baseline system. For the answer retrieval task (Task 1), we used fastText to embed text, and Tangent-CFT to emebed formulas, concatenating text and formula vectors to represent each post (answer or question). Our runs differed from each other in terms of how the vector embeddings were used for answer retrieval. Our best run found similar questions to the topics and used the answers given to those similar questions. In future work, we plan to conduct further analysis of our results and to further enhance our systems for both tasks.

Acknowledgements. This material is based upon work supported by the National Science Foundation (USA) under Grant No. IIS- 1717997 and the Alfred P. Sloan Foundation under Grant No. G-2017-9827.

References

1. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017)
2. Davila, K., Zanibbi, R.: 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. pp. 1165–1168 (2017)
3. 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 (2020)
4. Mansouri, B., Rohatgi, S., Oard, D.W., Wu, J., Giles, C.L., Zanibbi, R.: Tangent-CFT: An embedding model for mathematical formulas. In: Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR). pp. 11–18 (2019)
5. Mansouri, B., Zanibbi, R., Oard, D.W.: Characterizing searches for mathematical concepts. In: Joint Conference on Digital Libraries (2019)
6. Pawlik, M., Augsten, N.: Tree edit distance: Robust and memory-efficient. Information Systems **56**, 157–173 (2016)
7. Zanibbi, R., Aizawa, A., Kohlhase, M., Ounis, I., Topic, G., Davila, K.: NTCIR-12 MathIR task overview. In: NTCIR (2016)
8. Zanibbi, R., Oard, D.W., Agarwal, A., Mansouri, B.: Overview of arqmath 2020: Clef lab on answer retrieval for questions on math (2020)