Approach Zero and Anserini at the CLEF-2021 ARQMath Track: Applying Substructure Search and BM25 on Operator Tree Path Tokens

Wei Zhong[1], Xinyu Zhang[1], Ji Xin[1], Richard Zanibbi[2] and Jimmy Lin[1]

[1]*David R. Cheriton School of Computer Science, University of Waterloo*
[2]*Department of Computer Science, Rochester Institute of Technology*

Abstract
This paper reports on substructure-aware math search system Approach Zero that is applied to our submission for ARQMath lab at CLEF 2021. We have participated in both Task 1 (math ARQ) and Task 2 (formula retrieval) this year. In addition to substructure retrieval, we have added a traditional full-text search pass based on the Anserini toolkit [1]. We use the same path features extracted from Operator Tree (OPT) to index and retrieve math formulas in Anserini, and we interpolate Anserini results with structural results from Approach Zero. Automatic and table-based keyword expansion methods for math formulas have also been explored. Additionally, we report preliminary results from using previous years' labels and applying learning to rank for our first-stage search results. In this lab, we obtain the most effective search results in Task 2 (formula retrieval) among submissions from 7 participants including the baseline system. Our experiments have also shown a great improvement over the baseline result we produced from previous year.

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

1. Introduction

The domain of Mathematics Information Retrieval (MIR) has been actively studied over recent years. It covers information retrieval in documents where math language presents. Traditional IR technologies have not addressed the special problems in MIR, including retrieval for structured math formulas or expressions where commutativity and symbol substitution may occur. MIR may also require understanding of math knowledge such as concept-to-formula association, higher level math semantics, and equivalent formula transformations, in order to create effective systems for math-aware text retrieval.

The ARQMath Lab [2, 3] addresses the problems of MIR under a Community Question Answering (CQA) setting. It utilizes user-generated data from the Math StackExchange (MSE) website as collection, and task topics are extracted from real-world math Q&A threads. The corpus contains over 1 million math-related questions and about 1.4 million answers, including

CLEF 2021 – Conference and Labs of the Evaluation Forum, September 21–24, 2021, Bucharest, Romania
w32zhong@uwaterloo.ca (W. Zhong); x978zhan@uwaterloo.ca (X. Zhang); ji.xin@uwaterloo.ca (J. Xin); rxzvcs@rit.edu (R. Zanibbi); jimmylin@uwaterloo.ca (J. Lin)

Table 1
Example formula retrieval result comparison between Tangent-CFT and Approach Zero for query $\boxed{O(mn \log m)}$ [7]

Rank	Tangent-CFT	Approach Zero
1	$O(mn \log m)$	$O(mn \log m)$
2	$O(m \log n)$	$O(nk \log k)$
3	$O(n \log m)$	$O(KN \log N)$

over 17 million math formulas or notations. The data collection covers MSE threads from 2010 to 2018, and task topics are selected from MSE questions of 2019 (for ARQMath-2020 [2]) and 2020 (for ARQMath-2021 [3]). A main task (CQA Task, or Task 1) and a secondary formula retrieval task (Task 2) are included in this lab. Participants are able to leverage math notations together with its (text) context to retrieve relevant post answers. For Task 1, complete answers are available for applying full-text retrieval, but participants are also allowed to utilize structured formulas in the documents. On the other hand, formula retrieval in Task 2 is about identifying similar formulas in the document that is related to the topic question. The formula retrieval task specifies query formula with its question post, and optionally, participant could use contextual information around the topic formula in the question post. Both tasks ask participants to return up to five runs (one primary run and four alternative runs) that contain relevant post answers to the given question topic. Relevance judgement will be received for primary runs and selected results of alternative runs from the submission pool. Official evaluation metrics include NDCG' [4], MAP', and P'@10, where MAP' and P'@10 use H+M binarization (hits with relevance score ≥ 2 are considered as relevant, and relevance levels are collapsed into binary). NDCG', MAP', and P'@K are identical to their corresponding standard measurements except that unjudged hits are removed before metric computation. Relevance is scored on a graded scale, from 0 (irrelevant) to 3 (highly relevant).

We submitted 5 runs for both tasks. Our system for this ARQMath lab is based on a structure-aware search system Approach Zero [5, 6] and a full-text retrieval system Anserini [1]. We adapted a two-pass search architecture for most of our submitted runs. In the Approach Zero pass, a substructure matching approach is taken to assess formula similarity where the largest common subexpression from formulas is obtained and we use the maximum matched subtree to compute their structure similarity. Symbol similarity is further calculated with the awareness of symbol substitutions in math formulas. The similarity metric used by Approach Zero is easily interpretable and it may serve better the needs of identifying highly structured mathematical formulas.

As illustrated by Mansouri et al. [7] in an example query result (see Table 1), substructure matching and variable name substitution are desired for identifying highly relevant math formulas. Usually, this can be more easily achieved using tree-based substructure search than using full-text search. However, searching math formulas also requires more "fuzzy" match or high-level semantics. In this case, embedding formulas or matching bag-of-word tokens using traditional text retrieval methods (but with careful feature selection) are shown to be effective as well [7, 8]. For example, Tangent-CFT system is able to find document formula

$\boxed{O(\lambda_\delta(n)\log n)}$ (not shown in the Table 1) for the example query by considering semantic information, but this formula is hard to be identified by substructure search engine because it shares little common structure feature with the query Operator Tree (OPT).

In our submission this year, we try to compensate strict substructure matching by introducing a separate pass that performs simple token matching on Operator Tree path tokens. Specifically, we include a full-text search engine Anserini to boost the results of Approach Zero. In the Anserini pass, we use feature tokens extracted from a formula as terms and directly apply full-text retrieval by treating those tokens as normal text terms. The difference between our Anserini pass and other existing bag-of-word math retrieval systems is that we use leaf-root path prefixes from formula Operator Tree representation (See Figure 1). This representation is the same representation we use to carry structural information for formulas in Approach Zero, but the latter additionally performs substructure matching and variable name substitution in math formulas.

We further try to improve our system recall by applying query expansion on both text and math query keywords. We investigate RM3 [9] (for both text and math keywords) and a method using lookup table to extract math keywords from formulas. In addition, we report the result from using previous years' label and applying learning-to-rank methods.

Our main objectives of experiments for this lab are as follows.

- Evaluate the effectiveness of treating OPT leaf-root path features as query/index terms.
- Try different ways to combine results from structure search and traditional bag-of-word matching paradigm. Evaluate the effectiveness of query expansion involving math formulas.
- Apply learning-to-rank methods to the ARQMath dataset from previous years' labels and post meta data, and determine its usefulness.

2. Formula Retrieval

2.1. Representation

In this lab, we adapt an enhanced version of Operator Tree (OPT) representation, trying to improve math formula retrieval recall. As illustrated by an example formula topic in Figure 1, this representation contains more nodes than typical OPT in the following ways:

- Always placing an `add` node on top of a term, this allows matching a path from a single term to another path from a multi-term expression, e.g., $\boxed{a}$ and $\boxed{a+b}$.
- Having an additional sign node (i.e., + and -) on top of each term. They are tokenized into the same token such that it can match another math term even with different signs. It changes the path fingerprint (see Section 2.2) so that we have the information to penalize those paths of different signs.
- For any variable, it will place a `subsup` node (optionally an additional `base` node) on top of the variable node, even if it comes without a subscript/supscript. This helps to increase recall for cases when subscripted and non-subscripted variables are both commonly used to denote the same math entity, e.g., $\boxed{kx}$ and $\boxed{c_1x}$. Notice that this rule is not applied to constants, as they are not usually subscripted in math notations.

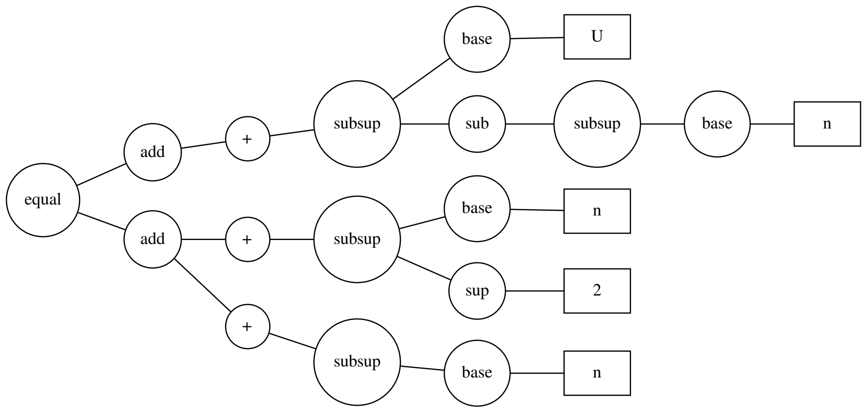


Figure 1: Operator Tree representation for topic formula keyword $\boxed{U_n = n^2 + n}$ (i.e., Topic B.285)

When being indexed, a formula OPT will be broken down into linear leaf-root paths, so that they can be treated as normal "terms" to be used in inverted index. Different leaf-root paths may end up being the same path tokens after applying tokenization, e.g., path `U/base/subsup/+/add/equal` and `n/base/subsup/+/add/equal` will result in the same token path VAR/BASE/SUBSUP/SIGN/ADD/EQUAL since both U and n are tokenized to variable token VAR (a capitalized name indicates it is tokenized). The purpose of tokenizing every node in the path is to improve recall, such that we can find identical equations with different symbol set, as it is frequently the case in math notations.

In addition to leaf-root path tokens, we also index the prefixes of those path tokens, this is necessary to identify a subexpression from a formula in the document. For example, if we need to find $\boxed{U_n = n^2 + n}$ by only querying its subexpression $\boxed{n^2 + n}$, then all the possible leaf-root path prefixes should also be indexed. To alleviate the cost, one may optionally prune prefix paths which always occur together. For example, the `*/base` path will always follow a `*/base/subsup` path (asterisk denotes any prefix), thus we can remove the former path to reduce index size.

2.2. Path Symbol Weight

Tokenization on path boosts recall for formulas, however, we still need original path information to break ties when tokenized paths have matched, e.g., $\boxed{a < 0}$ and $\boxed{a \leq 0}$. To address this issue, in this task, we apply a 3-level similarity weight for path-wise matching. More specifically, we use the original operator symbols along the token path to generate a hash value for each path by computing the Fowler-Noll-Vo hash [10] from the leaf node up to 4 nodes above, and we call this hash value the *fingerprint* of a path. The fingerprint captures a local symbolic appearance for operators on a path, it can be used to differentiate formulas of the same structure but with different math operator(s).

Upon scoring the match between a pair of paths, we compare against their leaf node symbol as well as fingerprint value, we will assign the highest path-match weight if both values agree between two paths, a medium weight if leaf symbols match but not the fingerprints, and a lower path-match score if otherwise. A weighted sum of matched paths represents the *symbol similarity* in our model.

2.3. Structure-Based Scoring

The Approach Zero formula search system takes a tree-based matching approach, and specialized query optimization is applied to match formula substructures during the very first stage of retrieval [6, 5]. The benefit of substructure matching is that the formula similarity score is well-defined and can be interpreted easily. In our case, the *structure similarity* is previously defined as the number of paths in the maximum matched common subtree [5]. In this task, we acknowledge the different contribution from paths and apply an IDF weight to a matched formula, defined by the sum of each individual path IDFs:

$$\mathrm{IDF}(\hat{T}_{q_f,d_f}) = \sum_{p \in \hat{T}_{q_f,d_f}} \log \frac{N_p}{df_p} \tag{1}$$

where p is the path in the largest common subtree $\hat{T}_{q_f,d_f}$ of query and document formulas, df_p is the document frequency of path p, and N_p is the total number of paths in the collection.

We also incorporate symbol similarity score $S_{sym}(q_f, d_f)$ to further differentiate formulas with identical structure but different symbols. This score is only computed in the second stage when structure similarity score is computed and possible to make the hit into top K results. Specifically, we penalize symbol similarity by the length of document formula L_{d_f}:

$$\mathrm{SF}(q_f, d_f) = \frac{1}{1 + (1 - S_{sym}(q_f, d_f))^2} \left((1 - \eta) + \eta \frac{1}{\log(1 + L_{d_f})} \right) \tag{2}$$

where the penalty is determined by parameter η.

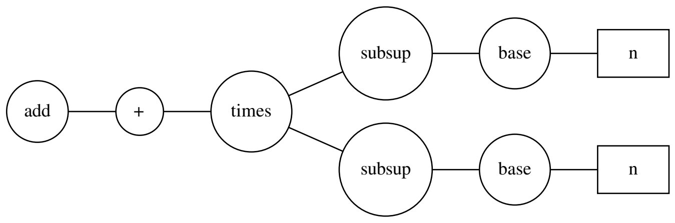
Given structure similarity and symbol similarity, we adapt the following formula to compute overall similarity for a math formula match:

$$\mathrm{Similarity}(q_f, d_f) = \mathrm{SF}(q_f, d_f) \cdot \mathrm{IDF}(\hat{T}_{q_f,d_f}) \tag{3}$$

whereas for normal text terms in query, we compute their scores using BM25+ scoring schema [11]. The final score for this pass is then accumulated from math and text keywords.

2.4. Text-Based Scoring

On the other hand, we also add a separate pass in parallel to score document formulas by “bag of paths” (without applying substructure matching). The path set for text-based scoring includes two copies of a path, one with original leaf symbol and another with tokenized leaf symbol, however, both types of paths apply tokenization to operator nodes (See Figure 2 for an example). Including original leaf symbol will award exact operand matches, and the fully tokenized leaf paths are included to boost recall and enable us to match expressions with different operand symbols.



```
_VAR_BASE_SUBSUP_TIMES_SIGN_ADD _VAR_BASE_SUBSUP_TIMES_SIGN _VAR_BASE_SUBSUP_TIMES
_VAR_BASE_SUBSUP _VAR_BASE _normal__n___BASE_SUBSUP_TIMES_SIGN_ADD
_normal__n___BASE_SUBSUP_TIMES_SIGN _normal__n___BASE_SUBSUP_TIMES
_normal__n___BASE_SUBSUP _normal__n___BASE _VAR_BASE_SUBSUP_TIMES_SIGN_ADD
_VAR_BASE_SUBSUP_TIMES_SIGN _VAR_BASE_SUBSUP_TIMES _VAR_BASE_SUBSUP _VAR_BASE
_normal__n___BASE_SUBSUP_TIMES_SIGN_ADD _normal__n___BASE_SUBSUP_TIMES_SIGN
_normal__n___BASE_SUBSUP_TIMES _normal__n___BASE_SUBSUP _normal__n___BASE
```

Figure 2: Up to bottom: Tree representation for formula keyword $\boxed{n \times n}$ (Topic B.201) in OPT and its decomposed paths to be used as query terms in text-based search system (symbols having a `_normal_` prefix indicates they have regular font).

Our text-based pass is based on Anserini [1], and we apply accurate Lucene BM25 [12] (with lossless document length) for scoring both text and formula paths, specifically

$$\sum_{t \in q} \log \left(1 + \frac{N - df_t + 0.5}{df_t + 0.5}\right) \cdot \frac{tf_{t,d}}{tf_{t,d} + k_1(1 - b + b(L_d/L_{avg}))} \tag{4}$$

where k_1 and b are parameters, and N, df_t, $tf_{t,d}$, L_d, L_{avg} refer to the total number of documents, the document frequency of the term t, the term frequency of term t in the document d, the length of document d, and the average document length respectively.

3. Math-Aware Retrieval

3.1. Query Expansion

In CQA Task, we need to return full-text answer posts as search results. In addition to simply merging results from formula and text retrieval independently, we have identified a few techniques to help map information from one type to another:

- To make use of information in formulas, we map tokens in LaTeX to text terms so that formula-centered document posts can also be found by querying text keywords.
- To utilize the context information in answer posts, we explore query expansion (covering both math and text) based on pseudo relevance feedback to add potentially relevant keywords based on both math and text context.

In the following sections, we explored two query expansion methods.

3.1.1. Math Keyword Expansion

For the purpose of mapping tokens in LaTeX to text, we designed some manual rules to convert a set of LaTeX math-mode commands to text terms. For example, we will expand text term "sine" to the query if a `\sin` command occurs in formula LaTeX markup. Furthermore, Greek-letter commands in LaTeX are also translated into plain text, e.g., `\alpha` will be mapped to term "alpha". A specialized LaTeX lexer from our PyA0 package [13] is used to scan and extract tokens from markup. The list of math keyword expansion mappings we have used in this task is enumerated in Appendix D.

In order to find more formulas by querying math tokens, we not only expand keywords in query, but also apply math keyword expansion to all document formulas for CQA Task.

3.1.2. RM3 Query Expansion for Mixed Types of Keywords

In addition to math keyword expansion, we apply RM3 [14] to expand larger range of possibly relevant terms or formulas from initially retrieved documents. Based on relevance model [9], RM3 optimizes the expansion by closing the difference between document language model $Q(w|D)$ and query relevance model $P(w|R)$, where random variable w represents generated word, D and R are document and relevant document respectively.

The objective is then reflected as negative KL divergence

$$- KL(P|Q) \propto \sum_{w \in V} P(w|R) \log Q(w|D) \tag{5}$$

and RM3 with pseudo relevance assumption utilizes top retrieved results $\mathbf{C}$ to approximate above objective. $P(w|R)$ is estimated by normalized expanded query probability, i.e., $P(w, q_1, ...q_n)/Z$ where q_i are existing query keywords and Z is normalizing constants. It can be further associated to query likelihood $\prod_i P(q_i|D)$ as shown below

$$\begin{aligned} P(w|R) &\approx \sum_{D \in \mathbf{C}} P(D)\ P(w, q_1, ...q_n|D)\ /\ Z \\ &= \sum_{D \in \mathbf{C}} P(D)\ P(w|D) \prod_i P(q_i|D)\ /\ Z \end{aligned} \tag{6}$$

the query likelihood can be approximately represented by other appropriate scoring: In this lab, we use BM25 for scoring in Anserini pass, and use the scoring functions stated in Section 2.3 in Approach Zero pass. To apply RM3 to math formulas, we treat math markup as the same as text keyword.

After estimating query relevance model from Eq. 6, we further perform an interpolation (using even ratio $\alpha = 0.5$) with maximum likelihood estimate of existing query keywords in order to improve model stability. We use top query keywords from our estimate of $P(w|R)$ to query them again in a cascade way. Our parameters for RM3 include: The number of top-K retrieved results used for estimation, and the number of top query keywords to be selected to query in the second round.

3.2. Learning to Rank ARQMath Answers

We make the assumption that most answer posts are relevant to its linked question post, thus we pair all answer posts with their question posts in the index. To eliminate the consequence from retrieving low-quality answers (i.e., answers irrelevant to its linked question), we apply learning to rank techniques using features such as the number of upvotes for an answer.

Two learning to rank methods have been explored, i.e., linear regression and LambdaMART [15]. LambdaMART works by minimizing the cross entropy between pair-wise odds ratio of perfect and actual results, and it is efficient and can be regarded as list-wise learning-to-rank method because it only requires to sample adjacent pairs. Furthermore, it can accumulate the "λ" for each document before updating parameters. λ serves as a nice symmetric connection for the cross entropy w.r.t model parameters. By default, LambdaMART is commonly set up to optimize NDCG measures by multiplying measurement gain directly to pair-wise $\lambda_{i,j}$ [16] where

$$\lambda_{ij} = -\frac{k}{1 + e^{k \cdot o_{ij}}} |\Delta_{NDCG}| \tag{7}$$

k is a parameter that determines the shape of distribution for probability p_{ij} that a document i is ranked higher than j, and o_{ij} is the likelihood ratio of p_{ij}.

The following factors are considered to rerank answer posts:

- **Votes** (the upvote number): As it is presumably a direct indicator to reflect answer post relevance to its question.
- **Similarity**: The first-phase score for each ranked result (which may be interpolated result from two separate passes, see discussion in Section 2.3 and 2.4).
- **Tags**: The number of tags matched between topic question and linked question of document. In ARQMath lab, each question has been potentially attached some "tags" to indicate the question scope in math terms. Tags are manually labeled by MSE users with a bar on reputation, and they can be a good abstraction for a Q&A thread.

These features are similar compared to the features proposed by Yin Ki NG et al. [17], however, they do not have assessed data available at that time, and they have to mock relevance assessments using indirect indicators (e.g., thread being marked as duplicate by users). Our experiments will be based on direct relevance judgement which is more reliable, accurate and less complicated. Furthermore, we also explore another learning-to-rank method using LambdaMART.

4. Experiment

The official results of our system compared to systems with best results are summarized in Table 9 and 10. System(s) noted "L" in the table are systems applying learning to rank using existing labels (for ARQMath-1, they are trained separately from 45 test topics in Task 2). Systems noted "M" use keywords manually extracted from topic post for querying, [1] while the

[1]The complete set of our manual queries can be found here: https://github.com/approach0/pya0/tree/arqmath2/topics-and-qrels (for 2021 topics, we use the "refined" version as indicated by our file name)

model execution is still automatic. The same subscripted M letter indicates the same set of topics. Notice that system TU_DBS team uses only text information (noted as “T”) for retrieval in Task 1. In addition, although Task 2 asks to submit at most 5 results for each visually unique formula, our index with limited number of visually unique formulas are not available in time, thus our official runs for Task 2 may contain extra results per unique formula (those are marked “F") and it may affect the comparison with other systems (although the official evaluation will remove those extra results, those are holding replaces in returned search results).

In our runs, a base letter (such as “P”, “A” etc.) indicates the set of parameters we have applied in Approach Zero system. Table 8 in Appendix shows the detailed parameters for different base settings. Math path weight is the weight associated to path in matched common subtree, it is used to adjust the contribution importance comparatively to text keyword match; formula η shown in Eq. 2 is the penalty applied to over-length formulas; and BM25+ is the scoring used for normal text search in in Approach Zero pass.

Additionally, we append a number to base letter in our run names to indicate the way it combines results with text-based system Anserini, details can be found in Section 4.5 and 4.6.

4.1. Task-1 Submission

In CQA Task, we adapt Lancaster stemmer, an aggressive stemmer that is able to canonicalize more math keywords, e.g. *summation* will be stemmed to *sum* whereas other stemmers such as Porter and Snowball will only convert it to *summat*.

Our Task-1 results are not quite competitive, however, we observe that text-only retrieval system from TU_DBS team can achieve better results than ours in Task 1. This implies that text retrieval alone in Task 1 plays a crucial role in effectiveness contribution, and a potential gain is anticipated when text retrieval and the way it combines with math search can be further improved in our case.

In our post experiments, we generate reranked runs for Task 1 by applying Linear Regression and LambdaMART (trees, depth = $7, 5$) directly on Approach Zero pass (see Section 4.7), which is trained on all Task-1 judgements from previous year. After applying learning to rank, our post-experiment result is on par with most effective systems in terms of P@10.

4.2. Task-2 Submission

In Task 2, two runs C30 and B30 using different base parameters achieve the same scores numerically, they are collapsed into one row in the table. We have also generated similar 5 runs for Task 2 again (having an asterisk on their run names) but with up to 5 results for each visually unique formula. We have further corrected an oversight in our Anserini pass which affects tree path token generation. It turns out our results can be further improved.

Without using any existing label for training in our official submission, we obtain the best effectiveness across all metrics in Formula Retrieval Task of this year data (ARQMath-2), and according to P’@10 metric, we are able to achieve the highest precision at the very top results in ARQMath-2 by returning results from Approach Zero alone (see run B*). We attribute this advantage to our structure-aware matching applied to the very first stage of retrieval process. Top-precision systems such as Tangent-S [18] introduced alignment phase to find matched substructures, and Tangent-CFTED performs tree edit distance calculation in reranking stage. These structure matching methods are too expensive to be applied in the first stage of retrieval.

Apart of the above results, we have conducted a variety of experiments trying to achieve the objectives listed in Section 1, although we select some best performed runs for submission, we still have made the following attempts in this paper to address those objectives:

- Explore traditional IR architecture and bag-of-words tokens using Anserini without applying substructure matching. And evaluate these two system using the same set of features extracted from OPT.
- Combine text-based approach with tree-based approach using score interpolation as well as search results concatenation, and try a more deeper integration to translate one type of token to another using RM3 and math keyword expansion.
- Apply learning-to-rank with ground truth labels from previous year using linear regression and LambdaMART, and evaluate their effectiveness.

All of our experiments in the following sections will be using previous-year topics for evaluation, since judgement data of this year is not available at the time we write this paper.

4.3. Employed Machine Configuration

Our experiments run on a server machine with the following hardware configuration: Intel(R) Xeon(R) CPU E5-2699 @ 2.20GHz (88 Cores), 1 TB DIMM Synchronous 2400 MHz memory and running on a HDD partitioned with ZFS.

4.4. Runtimes and Efficiency

For measuring runtime statistics, both systems are querying tokens extracted from the same representation in a similar way (i.e., by extracting prefix leaf-root paths from Operator Tree representation). Our index contains over 23 million formulas, over 17 millions of which are structured formulas (not single-letter math notations). The statistics of our path tokens per topic is (143.5, 102.5, 110, 400) in (avg, std, med, max).

Table 2 reports the query execution time separately from two passes. Compared to our previously published results [5], the system we have used in this task has on-disk math index also compressed, that technical improvement has improved system efficiency. However, our query execution times are unable to match Anserini which only performs matching at the token level without aligning substructures.

4.5. Text and Math Interpolation

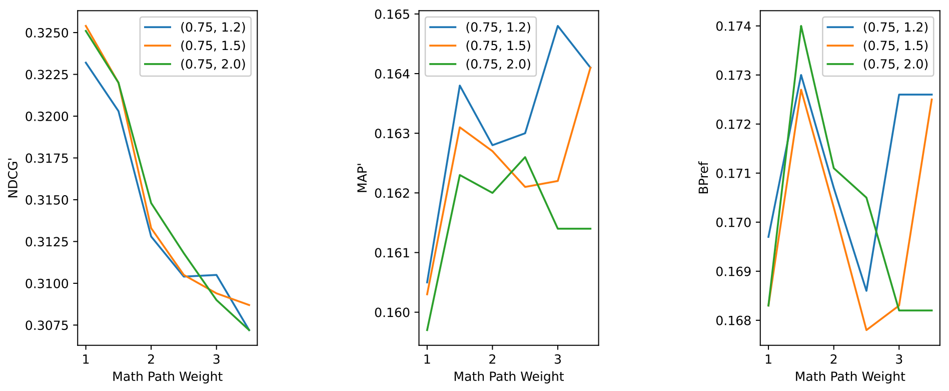
In our substructure search, an uniform weight is associated with each path for scoring, we first investigate how this weight affects overall retrieval effectiveness. We fix the BM25+ parameters (b, k_1) in Anserini pass to (0.75, 1.2), (0.75, 1.5) and (0.75, 2), and change math path weight from 1 to 3.5 with a pace of 0.5. An evaluation on CQA Task is conducted here since this task requires trade-off between text terms and math formulas.

As seen in Figure 3, measures from different BM25 parameters follow the similar trend with respect to math path weight. As path weight goes larger, NDCG' degrades consistently, this

Table 2
Query execution times (in milliseconds) for our submission in two separated passes (numbers are generated using single thread and the distribution is averaged over four independent runs).

	Approach Zero					**Anserini**				
	min	med.	max	avg.	std	min	med.	max	avg.	std
Task 1	23.0	1148.5	20518.0	2000.2	2735.3	241.0	609.0	1506.0	663.2	252.7
Task 2	167.0	1518.0	110997.0	3066.9	7008.8	44.0	164.0	797.0	212.3	155.6
Overall	23.0	1338.5	110997.0	2533.5	5343.4	44.0	386.5	1506.0	437.8	308.0

Figure 3: Math Path Weight Effect for Task 1 on 2020 Topics.



aligns with MathDowsers runs [17] as they observe best performance when "formula weight" is almost minimal ($\alpha \approx 0.1$). however, the other measures reach higher points when math path is weighted more than text terms, but they tend to be unstable. We believe this is because MAP' and BPref changes are very minor in this evaluation, they will have greater chance to flutter. Also, NDCG' is shown generally more reliable [4] among other measures used for incomplete assessment data.

Then we have investigated combining bag-of-words tokens from Anserini, and we choose fixing BM25 parameters to (0.4, 0.9) in Anserini pass. We first adapt $\alpha = 0.5$ linear interpolation ratio after normalizing scores to $[0, 1]$, and then merge results from Approach Zero and Anserini in the second stage. The interpolation is expressed by

$$\text{final score} = \alpha \cdot S_{app} + (1 - \alpha) \cdot S_{ans} \tag{8}$$

where S_{app} and S_{ans} are scores generated by Approach Zero (fixing math path weight to 1.5) and Anserini respectively.

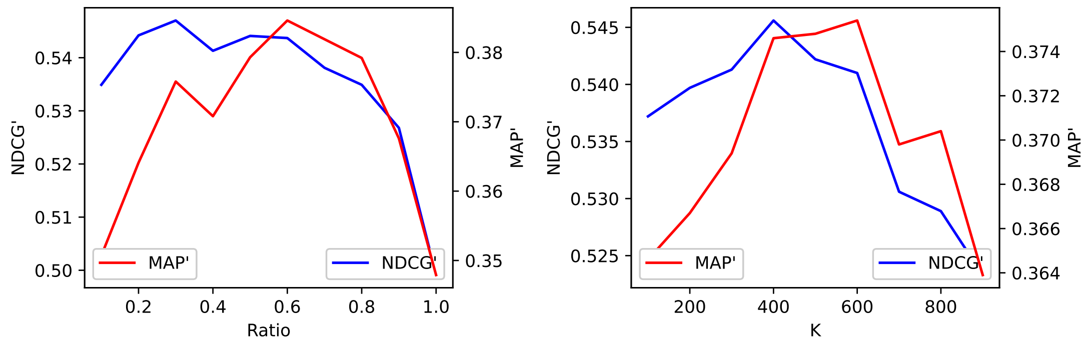
Three cases to combine with Anserini are examined, including using text terms only, using math paths only and using both text terms and maths (different type of tokens are treated the same in Anserini pass, all use BM25 scoring without substructure matching ability). For comparison, we also list the results from each individual system as well.

As shown in Figure 3, Anserini generally improves results if combined with Approach Zero. A boost of score from text-only Anserini is expected as Anserini alone achieves better results in

Table 3
Results using different type of tokens from Approach Zero, Anserini and even interpolation from both systems.

System	Text			Path			Text+Path		
	NDCG'	MAP'	BPref	NDCG'	MAP'	BPref	NDCG'	MAP'	BPref
Approach Zero	0.228	0.088	0.093	0.270	0.148	0.161	**0.326**	**0.156**	**0.168**
Anserini	0.231	0.094	0.098	0.220	0.087	0.095	**0.247**	**0.097**	**0.105**
Combined Ans ($\alpha = 0.5$)	**0.363**	**0.171**	**0.179**	0.339	0.157	0.161	0.345	0.160	0.168

Figure 4: Results from interpolation and concatenation using different parameters. Left to right: simple linear interpolation with different ratios. And concatenation of ranked hits from substructure-matching system Approach Zero at the top K, and hits from non-structural system Anserini for the lower-ranked positions.



text search, and given most of the keywords from query and results are text terms, combining Anserini can be beneficial. We notice that the path-only Anserini run also boosts scores, and we believe this is because paths tokens used in Anserini adds recall, whereas Approach Zero using substructure matching is good at adding precision, which are complementary to each other. However, text-only retrieval from Anserini contribute the most to structure-aware search in Approach Zero.

We are also interested at the combination effect in Task 2. Under our assumption that structure-aware search tends to produce good precision at the top, and path tokens are helpful to search recall, we have designed two possible ways to merge our math retrieval results from Approach Zero and Anserini: (1) Keep top-K results from Approach Zero using structure search, and the rest of results from top-1000 are concatenated from Anserini pass. (2) Uniformly apply score interpolation in Eq. 8 but with different ratios this time. Our official submissions are also named by above two conditions, i.e., a base run letter following the method we use to merge results. For example, A55 interpolates base run A with Anserini results using a ratio of 0.55, and P300 uses top-300 results from base run P and concatenates results from Anserini for the lower ranks.

Figure 4 shows the evaluation summary for combining results from Approach Zero and Anserini using two different methods. Approach Zero uses the same base configuration "P", we

Table 4
Math keyword expansion evaluation. Summary of effectiveness for baseline runs and results after applying math keyword expansion.

Base Run	**Baseline**			**Expansion Applied**		
	NDCG'	MAP'	BPref	NDCG'	MAP'	BPref
A	0.320	**0.164**	0.173	**0.323**	0.159	**0.168**
B	0.311	**0.163**	**0.170**	**0.317**	0.160	0.169
P	0.325	0.160	0.168	**0.328**	0.160	**0.171**

vary the interpolation ratio and K to see how effectiveness is affected.

We observe that the weighted merge of results with a ratio around 0.3 or 0.6 achieves higher NDCG' and MAP' in general. And concatenation of search results is shown slightly better in effectiveness in this case, the almost even concatenation achieves optimal NDCG' and MAP' scores. These have indicated the contribution from either system is essential to achieve good results for Task 2, and they are very complementary when they contribute evenly in general. On the other hand, the concatenation results have justified our assumption that the top results (i.e., top 400 in this case) from Approach Zero are very effective comparatively.

4.6. Text and Math Expansion

We use our best base runs (A, B and P) to test the effectiveness of math keyword expansion (as described in Section 3.1.1). The experiment is conducted for Task 1 only, and we are only expanding query keywords in Approach Zero pass. Math keyword expansion is applied both in index and in query to boost formula recall.

As only a small portion of math keywords can be expanded by our manual rules, and content containing formulas is just a partial of the collection, we do not observe a large advancement in effectiveness. However, the gain in NDCG' is consistent. And because the NDCG' measure is shown to be more stable than other measures here (see Table 4), we still think the effect of math expansion in Task 1 is beneficial. Nevertheless, the rules used in math keywords expansion have to be designed manually, and it may ignore other alternative synonyms and equivalent terms for math tokens.

We have noticed that the naive math keyword expansion applies uniform weight to keywords after expansion, this has important downside in contrast to query expansion methods such as RM3 [14] which will adjust boost weight to new query according to the relevance model. For example, many formula topics contain "greater or equal to" sign $\boxed{\geq}$, however, they are not always relevant to e.g., inequality, so expanding such terms using uniform weights is going to hurt effectiveness. On the other hand, RM3 will assign smaller weight in this case, because the term "inequality" is unlikely to co-occur with $\boxed{\geq}$.

In the following experiment, we also explore using RM3 for expanding query keywords. We simply treat math markup as terms so it can integrate into RM3 naively. RM3 has two parameters in our implementation, i.e., the number of keywords Q in the query after expansion, and the number of top documents K to be sampled for relevance model. We use (Q, K) to uniquely determine a RM3 run.

Table 5
Effectiveness results from different base runs with and without RM3 being applied. Some of the runs have math keyword expansion (Math Exp.) enabled.

Base Run	**Settings**		**Results**		
	RM3 Parameters	Math Exp.	NDCG'	MAP'	BPref
P	Not Applied		0.325	**0.160**	0.168
	(15,10)		0.323	0.153	0.158
	(20,10)		0.322	0.153	0.159
	Not Applied	✓	0.328	**0.160**	**0.171**
	(20,10)	✓	**0.329**	0.154	0.158
C	Not Applied		0.313	**0.163**	**0.171**
	(15,10)		0.323	0.160	0.166
	(20,10)		0.325	0.161	**0.171**
	Not Applied	✓	0.319	**0.163**	0.170
	(20,10)	✓	**0.335**	0.161	0.168

Table 6
Eight-fold averaged test results using learning-to-rank methods compared to results from baseline B without reranking.

Method	**Parameters**	**Measures**		
		NDCG'	MAP'	BPref
Baseline	(base run B)	0.314	0.148	0.160
Linear Regression		**0.324**	**0.156**	0.163
LambdaMART	trees=5, depth=3	0.303	0.132	0.144
	trees=5, depth=5	0.320	0.155	**0.167**
	trees=7, depth=3	0.302	0.131	0.142
	trees=7, depth=5	**0.324**	**0.156**	0.164
	trees=10, depth=5	0.323	0.153	0.158
	trees=10, depth=10	0.318	0.153	0.166

Two base runs are used in this experiment, P and C. As shown in Table 5, there is good improvement from RM3 in base run C, and it has a greater improvement compared to the gain from using math keyword expansion. However, this improvement is not consistent, as in run P, RM3 is actually harmful if not combined with math keyword expansion. Overall, benefit from query expansion is not notable, and our experiment shows the introducing of RM3 can be also harmful on some initial settings. But because math keyword expansion helps consistently across both experiments, we choose to apply it to all of our submissions for Task 1.

4.7. Learning to Rank

Inspired by the fact that previous year judgement is available for the first time in this lab, we want to study the effectiveness of reranking from utilizing these data. However, we do not have learning-to-rank results tuned correctly at the time we submit our runs, so these are completed as post-experiment runs.

Our experiment investigated two methods, simple linear regression and LambdaMART. We have our base run B as baseline and rerank its results with these two models. Experiment is conducted on Task 1 (because our features are mostly indicators for document-level similarity) with 39,124 relevance samples from previous year judgement pool, we split the data into 8 folds and validate model effectiveness by reporting averaged measures across each test data. We use the number of upvotes V, the number of tag matches T, and the ranking score S produced from Approach Zero as feature set.

As shown in Table 6, simple linear regression can achieve similar performance gain compared to LambdaMART model, presumably because the limited available data we have in ARQMath-1. The averaged coefficients for our resulting 8-fold linear regression model after training is $V, T, S \approx [0.002, 0.109, 0.007]$. Compared to the feature selection by Yin Ki NG et al. [17], we do not include user-wise metadata such as user reputation and their history upvotes. However, similar to their findings, our experiment echos that tag matches is a very important feature for this task.

4.8. Query Analysis and Case Study

To understand what is causing effectiveness changes in different methods, and to compare our math retrieval with the state-of-the-art system, we have gone through a query-by-query case analysis to understand results in different methods.

We plot the NDCG' scores per query for both Task 1 and Task 2 in Appendix E and F. Figure 5 compares Task 1 scores from different methods: A base run configuration (P), base run with math keyword expansion (P-mexp), base run with RM3 $= (20, 10)$, the same base run results merged with Anserini system using math tokens (P-50-ans0409desc), and the same base run results merged with Anserini system using text tokens (P50-ans0409title). Both merged results have a merge ratio $\alpha = 0.5$ and they use BM25 parameters (0.4, 0.9). Figure 6 consists of per-query results for the formula retrieval task, and here we also compare the results to Tangent-S system. Run P30-ans7515, P50-ans7515, and P300-ans7515 are different ways to merge results with Anserini, they all use BM25 parameters (0.75, 0.15).

4.8.1. The Effect of Different Methods

First, combination with Anserini almost uniformly improves effectiveness, either by using text tokens or math tokens in a bag-of-word model. In a few cases, the improvement from combining bag-of-word math tokens is profound, e.g., for topic A.19 $\boxed{p^4 - 1}$, A.68 $\boxed{a^n + 1}$ and A.93 $\boxed{det(xI - AB) = det(xI - BA)}$, when formulas should be matched entirely. However, in cases like A.40 $\boxed{a_1x_1 + a_2x_2 + a_3x_3 + ... + a_nx_n =}$ or A.83 $\boxed{1, 1 + \frac{1}{2}, 1 + \frac{1}{2} + \frac{1}{3}, ...}$, math bag-of-word tokens tend to suffer because these formulas require evaluating partial matches more structurally in order to assess similarities.

Second, adding only text tokens alone can greatly improve results, because many formula keywords are hurt by either malformed or irregular formula markups. For example, A.32 $\boxed{Empty(x) \iff ...}$ uses text without surrounding **text**, and A.55 has Unicode encoding in the markup and our parser could not handle. Other formula keywords do not produce similar formulas in search results and may need to rely on text keywords as they are more informative, notably A.80 and A.90. Similarly, less informative math formula keywords in the topic generally benefit from query expansion. For example, in topic A.99, formula keyword $\boxed{f : \mathbb{R} \to \mathbb{R}}$ adds expansion terms “rational number” which capture the semantic meaning of this math expression even it is hard to find many such structures in the indexed documents.

Math keyword expansion has boosted a few queries notably, but it can also hurt results such as in A.26, where it expands “fraction” keyword to the query because it contains a fraction in an integral $\boxed{\int_0^\infty \frac{\sin x}{x} dx}$, which is obviously more about “integral” than “fraction”. This confirms our assumption that weights assignment to expanded query keywords is essential in order to keep math keyword expansion generally beneficial. On the other hand, RM3 has mostly mild increase/decrease on baseline, and the overall improvement is minor.

In Table 6, we are comparing to one of the most effective systems in Task 2, i.e., Tangent-S [18]. However, we do notice there are queries we could not generate any result, mostly because our semantic parser is unable to handle some formulas. For example, topic B.11 has the following formula in the original topic where parentheses would not pair correctly.

$$\boxed{\iint_V f(x,y)dx\,dy = \iint_Q f(\Phi(u,v)\left|\frac{\partial\Phi}{\partial u} \times \frac{\partial\Phi}{\partial v}\right|}$$

Tangent-S on the other hand, uses both *Symbol Layout Tree* and Operator Tree to represent formulas, and if they fail to parse OPT, they can work from Symbol Layout Tree as fallback, the latter only captures topology of the nodes in a formula, and in that case, parentheses can be unpaired. This exposes one of our crucial weakness in searching formulas, i.e., we are heavily relying on well-defined parser rules to handle user-created data, and a failure in parsing would end up zero recall in our system.

Nevertheless, we have successfully demonstrated some advantages, for example, Table 7 is a comparison of results from our system and Tangent-S. We are able to identify commutative operands and rank highly relevant hit to the top. However, our result at rank 3 is not relevant because the exponential power in the query is a fraction, while our returned result does not have fraction as power, even if the number of operands matched in that case is large. In this particular query, our NDCG’ score is not competitive to Tangent-S, because after top results, Tangent-S is also able to return partially relevant (relevance level $= 1$) formulas such as $\boxed{\frac{1}{2}(1 + i\sqrt{3})}$ at a lower rank (not shown in the table), while our results at similar positions may match more operands at tree leaf, but they can be less relevant results due to missing key symbolic variable $\boxed{i}$, e.g., $\boxed{(1 + x)^{1/2}}$.

4.9. Strength and Weakness

In terms of effectiveness, our system is able to retrieve formulas with math structure awareness. Our system is very effective in formula search, our structure search is able to be applied to the very first stage of retrieval and produce highly effective results without reranking.

Table 7
Query case comparison for top-5 results generated from topic B.12 $\boxed{(1+i\sqrt{3})^{1/2}}$ in Task 2 (judged only).

Rank	**Approach Zero**		**Tangent-S**	
	Results	Relev.	Results	Relev.
1	$(1+\sqrt{3}i)^{1/2}$	3	$z=(4+4\sqrt{3}i)^{1/2}$	2
2	$(4+4\sqrt{3}i)^{1/2}=2(1+\sqrt{3}i)^{1/2}$	2	$4(1+\sqrt{3}i)^{1/2}$	2
3	$(1+i\sqrt{3})^{8}+(1-i\sqrt{3})^{8}=$	0	$(1+\sqrt{3}i)^{1/2}$	3
4	$(1+i\sqrt{3})^{8}$	0	$z=\dfrac{(2+2i)^{3}}{(1+i\sqrt{3})^{4}}$	0
5	$(1+i\sqrt{3})^{3}$	1	$z=\frac{(2+2i)^{3}}{(1+i\sqrt{3})^{4}}$	0

However, as indicated by Task-1 results, our method to handle text and math tokens together is not ideal. On the other hand, in Task 2, some of our results are screwed by failure to parse some math markups, our OPT parser is less robust to handle user created math content than SLT parser, because OPT requires higher level of semantic construction to be resolved (e.g., to pair parentheses vertical bars in a math expression).

So far, our experimental results using learning-to-rank methods does not understand a fine-grind level of math semantics, we could incorporate more features in lower level to further exploit these methods. Also, we have not applied embedding to formula retrieval yet. As demonstrated by other recent systems [7, 19, 20], embedding applies less strict matching than substructure search and it can often greatly improve effectiveness.

Finally, although our formula search results are effective and the structure search pass employs a dedicated dynamic pruning optimization [5] for querying math formulas, we are still not reaching the level of efficiency of text-based retrieval search system.

5. Conclusion and Future Work

In this paper, we have investigated different ways to combine our previous system Approach Zero based on substructure retrieval and another full-text retrieval system Anserini using bag-of-word tokens. We have evaluated and compared the effect of merging results by different tokens (i.e., text-only, math-only and mixed types), and by different methods (i.e., concatenation and linear interpolation). We demonstrate the usefulness of combining linear tokens into structure-aware math information retrieval using OPT prefix paths.

We also try using query expansion techniques to assist CQA task, we reported our preliminary evaluation results for math-aware search by applying RM3 model and a new math keyword expansion idea. We have also investigated using a few CQA task features to train and rerank search results, utilizing a small scale of labeled data. Our submissions to formula retrieval task of this year have achieved the best effectiveness over all official metrics. In the future, we need to add a more tolerant and efficient parser so that we can parse user created data more robustly. We are interested to introduce data driven models that target math retrieval more specially. Additionally, more features can be explored to achieve greater effectiveness boost by learning from existing labels.

References

[1] P. Yang, H. Fang, J. Lin, Anserini: Enabling the use of lucene for information retrieval research, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2017, pp. 1253–1256.

[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 CLEF Association (CLEF 2020), 2020, pp. 169–193.

[3] R. Zanibbi, B. Mansouri, 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 CLEF Association (CLEF 2021), 2021.

[4] T. Sakai, N. Kando, On information retrieval metrics designed for evaluation with incomplete relevance assessments, Information Retrieval 11 (2008) 447–470.

[5] W. Zhong, S. Rohatgi, J. Wu, C. L. Giles, R. Zanibbi, Accelerating substructure similarity search for formula retrieval, in: European Conference on Information Retrieval, Springer, 2020, pp. 714–727.

[6] W. Zhong, R. Zanibbi, Structural similarity search for formulas using leaf-root paths in operator subtrees, in: European Conference on Information Retrieval, Springer, 2019.

[7] 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, pp. 11–18.

[8] 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, pp. 1–10.

[9] V. Lavrenko, W. B. Croft, Relevance-based language models, in: ACM SIGIR Forum, volume 51, ACM New York, NY, USA, 2017, pp. 260–267.

[10] P. V. Glenn Fowler, L. C. Noll, Fowler/noll/vo hash, www.isthe.com/chongo/tech/comp/fnv, 1991.

[11] Y. Lv, C. Zhai, Lower-bounding term frequency normalization, in: Proceedings of the 20th ACM international conference on Information and knowledge management, 2011, pp. 7–16.

[12] C. Kamphuis, A. P. de Vries, L. Boytsov, J. Lin, Which bm25 do you mean? a large-scale reproducibility study of scoring variants, in: European Conference on Information Retrieval, Springer, 2020, pp. 28–34.

[13] W. Zhong, J. Lin, Pya0: A python toolkit for accessible math-aware search, in: Proceedings of the 44th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR), 2021.

[14] N. Abdul-Jaleel, J. Allan, W. B. Croft, F. Diaz, L. Larkey, X. Li, M. D. Smucker, C. Wade,

Umass at trec 2004: Novelty and hard, Computer Science Department Faculty Publication Series (2004) 189.

[15] C. J. C. Burges, K. M. Svore, Q. Wu, J. Gao, Ranking, Boosting, and Model Adaptation, Technical Report MSR-TR-2008-109, 2008. URL: https://www.microsoft.com/en-us/research/publication/ranking-boosting-and-model-adaptation/.

[16] P. Donmez, K. M. Svore, C. J. Burges, On the local optimality of lambdarank, in: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, 2009, pp. 460–467.

[17] N. Yin Ki, D. J. Fraser, B. Kassaie, G. Labahn, M. S. Marzouk, F. W. Tompa, K. Wang, Dowsing for math answers with tangent-l, in: International Conference of the Cross-Language Evaluation Forum for European Languages (Working Notes), 2020.

[18] R. Zanibbi, K. Davila, A. Kane, F. W. Tompa, Multi-stage math formula search: Using appearance-based similarity metrics at scale, in: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, 2016, pp. 145–154.

[19] S. Peng, K. Yuan, L. Gao, Z. Tang, Mathbert: A pre-trained model for mathematical formula understanding, arXiv preprint arXiv:2105.00377 (2021).

[20] Z. Wang, A. Lan, R. Baraniuk, Mathematical formula representation via tree embeddings, Online: https://people. umass. edu/˜ andrewlan/papers/preprint-forte. pdf (2021).

A. Approach Zero Parameter Settings

Table 8
Base parameters used in our system for experiment run.

Base Letter	**Math Path Weight**	**Formula η**	**BM25+ (b, k_1)**
P	2.5	0.3	0.75, 2.0
A	1.0	0.3	0.75, 2.0
B	1.5	0.3	0.75, 1.2
C	2.0	0.3	0.75, 1.2

B. Official Results and Post Experiments (Task 1)

Table 9
Official results from different submissions across years (Task 1). Our system used as baseline in last year did not contribute to the judgment pools (indicated by dagger).

Team	**Runs**	**Note**	**ARQMath-1**			**ARQMath-2**		
			NDCG'	MAP'	P'@10	NDCG'	MAP'	P'@10
(baseline)	Approach0†	M_1	0.247	0.099	0.062	-	-	-
(baseline)	TF-IDF + Tangent-S		0.248	0.047	0.073	0.201	0.045	0.086
MathDowsers	primary		**0.433**	0.191	0.249	**0.434**	**0.169**	0.211
DPRL	Task1-auto-both-P		0.422	**0.247**	**0.386**	0.347	0.101	0.132
TU_DBS	TU_DBS_P	T	0.380	0.198	0.316	0.377	0.158	**0.227**
Approach0	A55	M_1	0.364	0.171	0.256	0.343	0.134	0.194
Approach0	B55	M_1	0.364	0.173	0.251	0.344	0.135	0.180
Approach0	B60	M_1	0.364	0.173	0.256	0.351	0.137	0.189
Approach0	P50	M_1	0.361	0.171	0.255	0.327	0.122	0.155
Approach0	B60RM3	M_1	0.360	0.168	0.252	0.349	0.137	0.192
Post Experiments								
Approach0	B-L2R-LR	M_1, L	-	-	-	0.351	0.153	0.218
Approach0	B-L2R-LMART	M_1, L	-	-	-	0.340	0.140	0.217

C. Official Results and Post Experiments (Task 2)

Table 10
Results from official submissions of each year, and our post-experiment results (Task 2).

Team	**Runs**	**Note**	**ARQMath-1** NDCG’	MAP’	P’@10	**ARQMath-2** NDCG’	MAP’	P’@10
(baseline)	Tangent-S		0.691	0.446	0.453	0.492	0.272	0.419
DPRL	ltrall	L	**0.738**	**0.525**	**0.542**	0.445	0.216	0.333
DPRL	Tangent-CFTED		0.648	0.480	0.502	0.410	0.253	0.464
MathDowsers	FormulaBase2020		0.562	0.370	0.447	0.552	0.333	0.450
Approach0	B	M_2, F	0.493	0.340	0.425	0.519	0.336	0.461
Approach0	C30/B30	M_2, F	0.527	0.358	0.446	0.516	0.295	0.393
Approach0	P30	M_2, F	0.527	0.358	0.446	0.505	0.284	0.371
Approach0	P300	M_2, F	0.507	0.342	0.441	**0.555**	**0.361**	**0.488**
Post Experiments								
Approach0	B*	M_2	0.500	0.349	0.430	0.538	0.354	**0.503**
Approach0	C*30/B*30	M_2	0.547	0.376	0.457	0.550	0.339	0.443
Approach0	P*30	M_2	0.547	0.376	0.457	0.549	0.329	0.400
Approach0	P*300	M_2	0.550	0.372	0.454	**0.587**	**0.386**	0.502

D. Math Keyword Expansion Mappings

Table 11
Summary for the manual mapping rules from any markup containing a math token (on the left column) to expansion text keywords (on the right column).

Math Tokens	Mapped Term(s)
α ...	alpha, beta ...
$\mathbb{R}, \mathbb{N}$	rational number, natural number ...
π	pi
0	zero
∞	infinity
$=, \neq$	equality
$>, <, \leq, \geq$	inequality
$\int, \oint$	integral
$\sum$	summation
$\frac{x}{y}$	fraction
$\sqrt{x}$	root
∂	partial, derivative
$!$	factorial
$\pmod{}$	modular, mod
$\sin, \cos, \tan$	sine, cosine, tangent
function/operator names	(corresponding names)

E. Per-Query NDCG' Results (Task 1)

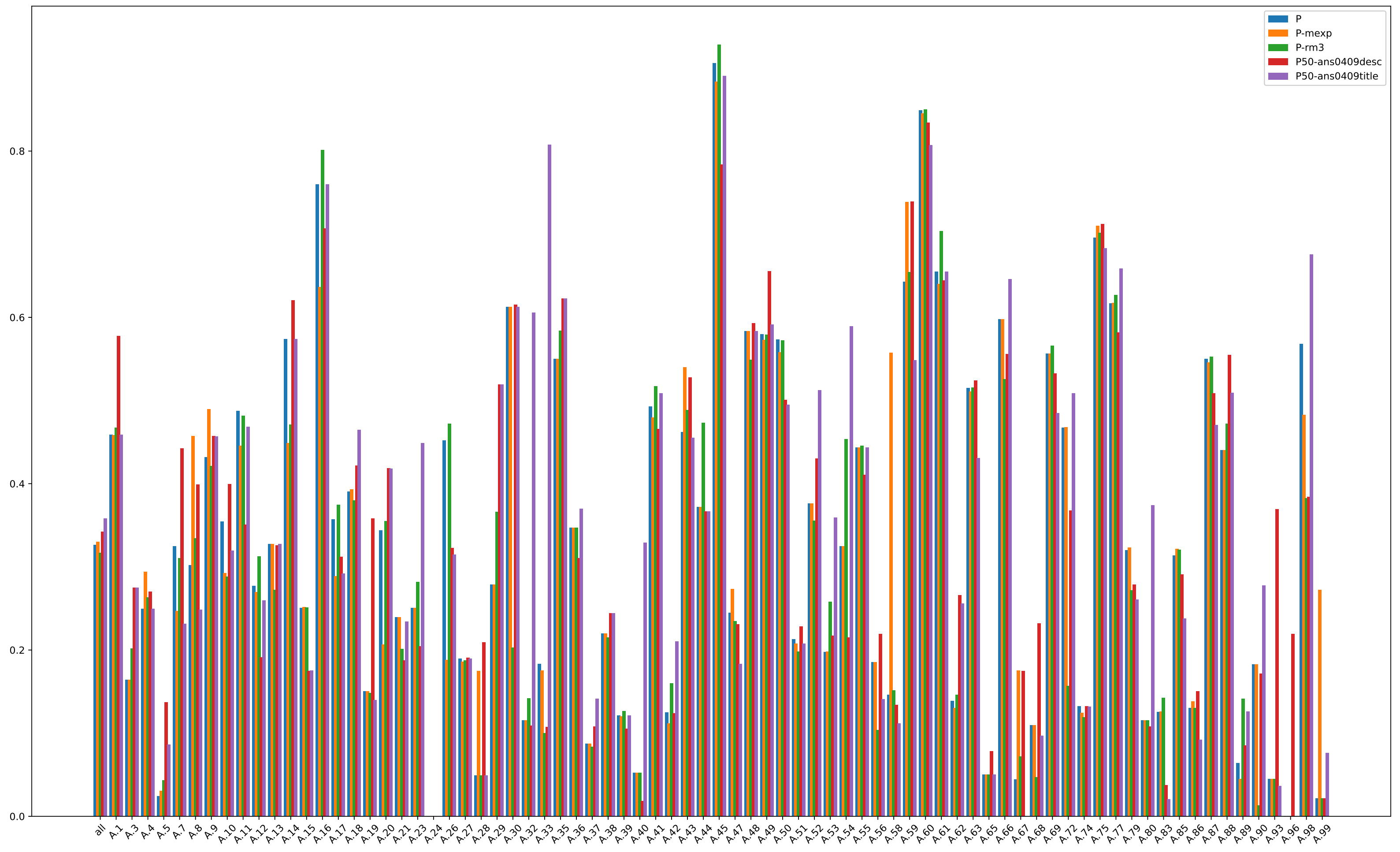
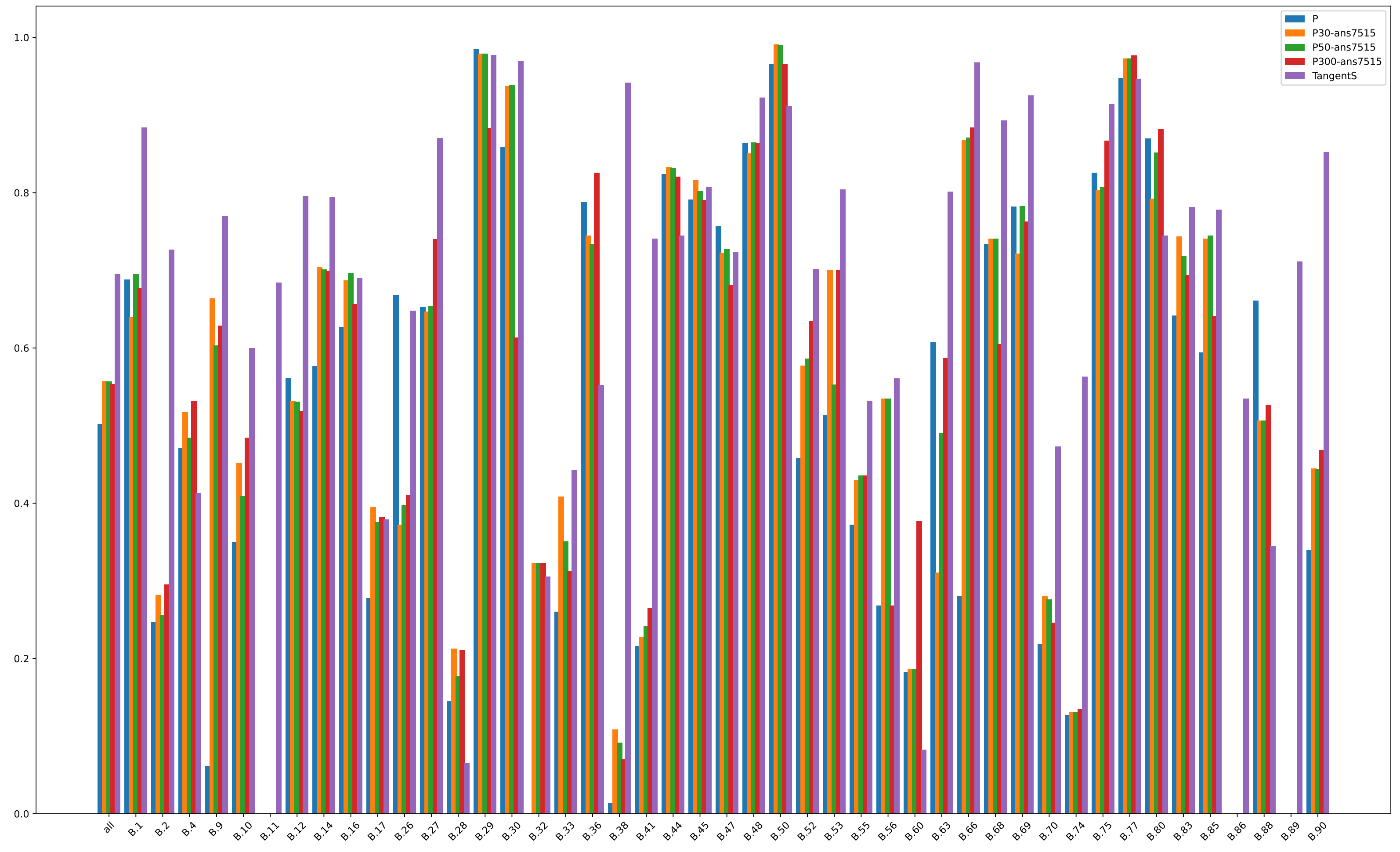


Figure 5: Individual query results compared to different methods using previous year topics in Task 1.

F. Per-Query NDCG’ Results (Task 2)

Figure 6: Individual query results compared to different methods using previous year topics in Task 2.


P
P30-ans7515
P50-ans7515
P300-ans7515
TangentS
all
B.1
B.2
B.4
B.9
B.10
B.11
B.12
B.14
B.16
B.17
B.26
B.27
B.28
B.29
B.30
B.32
B.33
B.36
B.38
B.41
B.44
B.45
B.47
B.48
B.50
B.52
B.53
B.55
B.56
B.60
B.63
B.66
B.68
B.69
B.70
B.74
B.75
B.77
B.80
B.83
B.85
B.86
B.88
B.89
B.90