PyA0: A Python Toolkit for Accessible Math-Aware Search

Wei Zhong and Jimmy Lin
David R. Cheriton School of Computer Science
University of Waterloo, Ontario, Canada

ABSTRACT

Mathematical Information Retrieval (MIR) has been actively studied in recent years and many fruitful results have emerged. Among those, the Approach Zero system is one of the few math-aware search engines that is able to perform substructure matching efficiently. Furthermore, it has been deployed in ARQMath2020, the most recent community-wide MIR evaluation, as a strong baseline due to its empirical effectiveness and ability to handle structured math content. However, in order to implement a retrieval model that handles structured queries efficiently, Approach Zero is written in C from the ground up, requiring special pipelines for processing math content and queries. Thus, the system is not conveniently accessible and reusable to the community as a research tool. In this paper, we present PyA0, an easy-to-use Python toolkit built on Approach Zero that improves its accessibility to researchers. We introduce the toolkit interface and report evaluation results on popular MIR datasets to demonstrate the effectiveness and efficiency of our toolkit. We have made PyA0 source code publicly accessible at https://github.com/approach0/pya0, which includes a link to a notebook demo.

CCS CONCEPTS

• **Information systems → Mathematics retrieval**.

KEYWORDS

Mathematical Information Retrieval

ACM Reference Format:
Wei Zhong and Jimmy Lin. 2021. 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 (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3404835.3462794

1 INTRODUCTION

Traditional full-text retrieval search engines are usually less effective when dealing with structured math formulas. The most common case is searching math formulas in educational content and scientific documents, where information is usually hard to find using keywords alone. For example, math symbols are often interchangeably used and two formulas are essentially equivalent if one can be transformed into the other by variable substitutions, e.g., $(1 + 1/n)^n$ and $(1 + 1/x)^x$. Also, commutative operands can be reordered e.g., $1 + 1/n$ and $1/n + 1$, without affecting semantics and users often query fragments of formulas, e.g., $(1 + 1/n)^n$ should be able to find $\lim_{x\to\infty}(1/x + 1)^x$. All these characteristics inevitably require important changes to traditional IR frameworks where tokenization, word representations, and index structures may need to be modified in order to match content structurally and to retrieve effective search results. Furthermore, highly relevant math formulas in response to user queries usually need strict substructure matching, and "bag of tokens" overlap or comparing embedding similarity will yield more effective results when combined with structure search [3]. These unique characteristics have made math formula retrieval a long-standing challenge in IR research.

To tackle this challenge, research in MIR (Mathematical Information Retrieval) has been active and has yielded fruitful results. Effort has been made to create new datasets and advance math formula search in terms of both effectiveness and efficiency. Popular datasets today include NTCIR-12 Math IR [7] and CLEF lab task ARQMath [9]. And over the years, evaluations have shown that search effectiveness has steadily improved. Search engines such as Approach Zero [11, 12] have achieved sub-second single-thread query latencies while being able to handle substructure matching in a semantics-aware manner. However, advancements in MIR have not been broadly disseminated in the general IR community, mostly due to the extra effort required to parse math markups in documents and the need for special tooling to handle structured content. In general, researchers unfamiliar with this domain need to devote extra effort to understand these nuances in order to run experiments and produce evaluation results.

In terms of concrete systems, Approach Zero has implemented a novel substructure search model in C.[1] However, its relatively large code base and implementation language is less accessible to the community today, where Python has emerged as the dominant language. In addition, processing math content can be time consuming: for example, LaTeXML[2] is a popular Perl library and important dependency used in most popular MIR systems to parse LaTeX markup into structured representations, can take several seconds to process a single LaTeXML string. This has made building indexes for math content generally much slower than building indexes for regular full-text search.

To address these issues, we have implemented an easy-to-use Python toolkit built on top of Approach Zero that abstracts away the preprocessing and handling of math formulas in documents. It avoids unnecessary time spent on preprocessing for the purpose of rapid prototyping by offering pre-built and publicly accessible indexes downloadable from simple API. The toolkit also exposes

SIGIR '21, July 11–15, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3462794

[1] https://github.com/approach0/search-engine/releases/tag/pya0
[2] https://dlmf.nist.gov/LaTeXML

```
1  import pya0, json
2  index_path = pya0.from_prebuilt_index("arqmath-2020-task1")
3  index = pya0.index_open(index_path, option="r")
4
5  JSON = pya0.search(index, [
6      { 'keyword': '2^n_>_n^k', 'type': 'tex'},
7      { 'keyword': 'induction', 'type': 'term'}
8  ], verbose = True, topk= 10, trec_output="./trec.out")
9  results = json.loads(JSON)
10 print(json.dumps(results, indent=4))
11
12 # Use a collection topics to generate TREC run file for evaluation
13 from pya0.eval import run_topics
14 run_topics(index, "arqmath-2020-task1", output="tmp.run")
15 pya0.index_close(index)
```

Figure 1: Search example with a pre-built math index.

lower-level functionalities to tokenize TeX markup content and retrieve raw documents from a math-specific index, and it builds on top of an efficient underlying search engine, Approach Zero, that is able to query both structured math formulas as well as standard unstructured text content. Approach Zero has been used as a strong baseline in the most recent MIR task, CLEF ARQMath2020 [9], and it was voted as the top 2020 Community Advertisement by Math StackExchange users.

In this paper, we demonstrate PyA0, a Python toolkit for math-aware search that provides bindings to Approach Zero. We believe that it can serve as a highly accessible playground for other IR researchers to experiment with MIR, gain the ability to handle structured math formulas and incorporate into their own systems.

2 SYSTEM OVERVIEW

PyA0 is available from PyPI and can be easily installed via `pip`. Using the toolkit, with a few lines of Python code, one is able to have a pre-built index downloaded in a few minutes, ready to accept search queries. The example searcher code snippet shown in Figure 1 automatically downloads a pre-built index for the most recent MIR test collection (ARQMath main task). An alternative pre-built index for the popular math collection NTCIR12-wiki is also available; this corpus contains only math formulas and is more manageable in size (156.3 MiB) and thus suitable for a quick start. The code between L4-9 calls a math-aware search interface, where each input keyword can be either `tex` or `term` type, and the result is a JSON-encoded string that contains search results. The utility function `run_topics` will generate a TREC-format output run file that can be easily evaluated using the `trec_eval` tool. The package is designed to automate index preparation and topic evaluation, and can be also used to fairly compare query latencies to other MIR systems with the greatest convenience.

The underlying math-aware search engine Approach Zero is a strong baseline and a good reference for math formula retrieval as it has already been used in evaluations to compare with recent top effective math-aware search systems in the literature [4, 6]. In the following, we will discuss the internals of functionalities for searching, indexing, and peripheral tooling.

2.1 Searcher

The searcher takes two types of keywords as query: normal text keyword (`term`) or math formula keyword (`tex`). The type must be specified by the user as there may be ambiguities, for example, matrix multiplication *AI* can also be the acronym for *Artificial Intelligence*. The searcher runs different query types on different indexes and uses a dynamic pruning algorithm [11] to generate structure-aware results efficiently.

Approach Zero has two stages in its query evaluation: In the first stage, entries in inverted lists are merged by tuple (document ID, formula document offset, path root-end node ID) or by document ID to identify relevant formulas as well as its substructures or its document occurrences. "Paths" are extracted structure features from math formula Operator Tree (OPT) representations at the index stage. After entries are merged at the query stage, the indexed path sets are grouped by their root-end nodes to restore the structure at query processing time, and the match between query structure and indexed structure is done cheaply by computing a minimum number between the number of paths from each of their unique path tokens, and these numbers are accumulated over their top-level substructures to determine the maximum match between query OPT and document OPT. This maximum match is used as *structure similarity*.

In the second stage, if structure similarity is not enough to make the hit enter into the top K results, the hit will be pruned. Otherwise, we continue to compute a *symbol similarity* [10] to further differentiate formulas with identical structure but potentially of different symbols, e.g., $E = mc^2$ and $y = ax^2$. The scoring algorithm uses BM25 ($b = 0.75, k_1 = 1.2$) to evaluate text terms and uses a metric similar to TF-IDF to evaluate formula keywords, except the empirical factor TF is replaced by a function of symbol similarity SF (short for *Significance Factor*, with a formula length penalty included). The intuition here is that math formula similarity depends little on the occurrence frequency of matched tokens. The *overall similarity* is simply the sum of BM25 and SF-IDF scores from text and formula terms, and formula structure similarity has already been captured during the summations in SF-IDF partial scores. Finally, search results will be summarized in a snippet by capturing up to a limit of keyword windows, and the system automatically highlights matched math formulas and text keywords in the snippet (using the HTML `<em>` tag).

The PyA0 package provides an intuitive high-level interface for the above process. As illustrated in Figure 1, the core `search` function takes a native Python array consisting of two types of query keywords and outputs JSON-formatted results. Each search result includes document ID, score, title, URL, and highlighted snippet.

2.2 Indexer

Indexing math formulas requires extracting features from structural representations to capture the structure or semantics of formulas. Recent tasks have shown that the top effective formula retrieval systems all take advantage of indexing tokens from structured tree representations [7, 9]. Currently, Approach Zero indexes prefix leaf-root paths from formula OPT representations, where each unique path corresponds to an inverted list, similar to regular search engines [11]. More specifically, a LaTeX markup is converted to OPT

```
import pya0
index = pya0.index_open("./tmp", option="w")
writer = pya0.index_writer(index)
pya0.writer_add_doc(writer,
    content="prove␣inequality␣by␣induction:␣" +
    "[imath]␣(a␣+␣b)^n␣\geq␣a^n␣+␣b^n␣[/imath]",
    url="https://math.stackexchange.com/questions/2528544"
)
pya0.writer_flush(writer)
if pya0.writer_maintain(writer, force=True):
    print('index␣merged')
pya0.writer_close(writer)
```

Figure 2: Simple example for PyA0 indexer usage.

and then the paths from the leaf to the internal nodes are extracted, for example, $x + y = 1$ will break down into five prefix paths: `x/+/=`, `x/+`, `y/+/=`, `y/+` and `1/=` (single token paths will not be generated, and we use "/" to visually separate individual path tokens). Prefix paths are used here because it helps to lookup sub-expressions in math formulas. The root-end node IDs from each prefix path are recorded in the index so later the shared common structures can be restored by grouping their root-end node IDs. The original path symbols are tokenized before they are indexed, resulting in only three indexing tokens in the above example, i.e., `VAR/ADD/EQ`, `VAR/ADD` and `VAR/EQ`, so that expressions of the same structure but with different symbols can also be matched. As a tie-breaker, the index has original leaf node symbols stored such that we can also differentiate exact symbol match and substituted symbol match (or *alpha-equivalence*). In addition, the document offset and size of the original expression are also stored in the index. The original expression size is used to penalize over-sized document formulas.

PyA0 provides indexer bindings at a high level that simulates regular search engine interfaces, hiding away parser calling, OPT representation construction, and path extraction. It also provides an interface to cache on-disk math index into memory with specified cache size limits for full-text index and math index separately (see Figure 2). Internally, it uses Indri[3] to handle full-text indexing, but implements its own math index format compressed using *Packed Frame of Reference* algorithm and accelerated using skip lists.

2.3 Peripheral Tooling

The ability to search math formulas relies on some important standalone tooling that may greatly impact system effectiveness. Components such as specialized math markup tokenizers or parsers are required to process math content, and an internal representation is needed to capture the structure and semantics of math formulas. In MIR tasks, LaTeXML is commonly used and internally relied on by many systems for such purposes; it converts LaTeX to MathML to build structured representations. However, LaTeXML does not provide convenient and efficient APIs for integration. Furthermore, MathML is often more bloated than LaTeX and processing such content is usually slower as it requires integration of an additional XML parser. Approach Zero has a LaTeX tokenizer and parser built into the search engine and is able to handle a large amount of user generated LaTeX markup in the wild and produce structured representations efficiently via a binary interface. Although the built-in tokenizer handles a comprehensive set (more than 500) math-mode LaTeX tokens, it is able to fallback to LaTeXML. The latter handles more types of math formula structures.

[3]http://www.lemurproject.org/indri.php

```
import pya0
tokens = pya0.tokenize("1␣+␣\\frac␣1␣{n␣+␣1}",
    include_syntatic_literal=True)
for tokenID, tokenType, symbol in tokens:
    print(tokenID, tokenType, symbol)
```

Figure 3: Simple example of PyA0 tokenizer usage.

In addition, the tokenizer in Approach Zero is aware of LaTeX token semantics. For example, two semantically similar math tokens such as `\sin` and `\cos` will map to the same token type `TRIGONOMETRIC`, and semantically identical tokens such as `\pm` and `\mp`, `\frac` and `\dfrac` will have the same token ID. The PyA0 package exposes the tokenizer functionality from Approach Zero conveniently. It scans LaTeX directly and maps each token to a (token ID, token type, original symbol) tuple (see Figure 3). Optionally, the PyA0 tokenizer can be specified to skip syntactic literals in LaTeX markups (such as "`{`" or "`\begin`") which only help syntactic construction without providing search information.

3 EXPERIMENTS

To demonstrate the effectiveness and efficiency of our system in searching documents with math content, we conduct experiments on two MIR datasets: NTCIR12-Math-Browsing and ARQMath2020, which are recent popular collections for MIR research. This setting covers formula-only retrieval and hybrid query with both math keyword and text keyword.

3.1 Datasets

NTCIR12-Math-Browsing [7] is the most recent MIR task from NTCIR-12; the corpus consists of 590,000 lines of isolated formulas in LaTeX from English Wikipedia articles containing `<math>` tag. The task is designed for formula-only retrieval and relevance is judged from a query and its returned results without any context information. The topics contain 40 queries, where 20 are wildcards queries where certain variables are specified to be matched with arbitrary subexpressions on candidate formulas. We evaluate over the 20 non-wildcards queries since our system does not support structure wildcards. Rather than using the default metrics in the NTCIR12-Math-Browsing task, we used BPref as it generates comparable results to previous runs.

ARQMath2020 [9] is a newer lab on Answer Retrieval for Questions about Math (ARQMath) from CLEF. The main task focuses on Community Question Answering (CQA), where participants need to retrieve relevant user-generated answer posts from Math Stack Exchange given a question about math. This task includes 98 carefully selected topics which cover a diverse range of math topics (including concept, proof, and computation) and different levels of difficulty (easy, medium, and hard). Among those topics, 77 are actually assessed. Three official evaluation measures are used in ARQMath2020: nDCG′ [5], MAP′, and P@10 where MAP′and P@10 use H+M binarization (hits with relevance score ≥ 2 are treated as relevant and relevance levels are collapsed to be binary) and

Dataset	Documents	Formulas	Index Size
NTCIR12-wiki	591 K	591 K	156.3 MB / 612 MB
ARQMath-main	1.45 M	14.0 M	9.6 GB / 5.6 GB

Table 1: Statistics of evaluation datasets and pre-built indexes. Index size is given in size of compressed gzip tarball (of file system image) and actual index size.

System	BPref		Query Latency			
	Full	Part.	Min.	Med.	Max.	Avg. / Std.
Tangent-S	0.6361	0.5872	**7**	561	3.7min	13450 / 43496
a0-best	**0.6726**	**0.5950**	89	1382	5543	1766 / 1574
a0-baseline	0.6586	0.5173	94	230	**809**	**274 / 165**
PyA0	0.6432	0.5426	16	**153**	1300	293 / 352

Table 2: NTCIR-12 Math Browsing collection evaluation. (query times are in milliseconds unless specified otherwise)

nDCG′and MAP′are identical to standard nDCG and MAP except that unjudged hits are removed before metric computation. For both task collections, the evaluation protocol requires a system to return a ranked list of up to 1000 hits for each topic. Relevance is scored on a graded scale, from 0 (not relevant) to 3 (highly relevant). The dataset sizes and resulting pre-built index sizes for PyA0 are shown in Table 1.

3.2 Baselines and Evaluation

MCAT [2], Tangent-S [1], and a previous version of Approach Zero (from Zhong and Zanibbi [12]) have reported effectiveness results on the NTCIR12-wiki dataset. MCAT also uses leaf-root paths extracted from tree structures as a subset of their features, but it has a costly query processing time, over 25 seconds on a server. Tangent-CFT [3] is a newer system that uses cosine similarity over embedding tuples from different tree representations to measure formula similarity, and it reports the best effectiveness on the NTCIR12-wiki dataset when combined with Tangent-S and Approach Zero. However, its effectiveness is comparable to Tangent-S in the recent ARQMath task for formula retrieval [9]. Approach Zero has a version (a0-best) that is able to match multiple common subtrees to generate top effective results, but its efficiency is inferior because important optimization techniques are difficult to apply under multiple tree matching. The a0-baseline run is an optimized version for efficiency where only the maximum common subtree is considered to assess formula similarity, and with the help of specialized dynamic pruning, it is able to obtain more practical running times. Compared to a0-baseline, the PyA0 underlying system adds IDF weights for each matched tree path and supports compression for the math index.

In the ARQMath task, MathDowsers [6] is currently the state-of-the-art system. It uses local information such as symbol pairs (with and without location attached) and terminal node (leaf) notation, and more global features such as compound symbol from *Symbol Layout Tree* [8]. Their system runs on top of Lucene, adapting BM25$^+$, a variant of BM25 that is shown to be effective for long documents. We include the MathDowsers best submission results from the ARQ2020 lab task (MD-best) and additionally a manual run (MD-manual) for comparison here. The MD-manual run, which uses the same set of manually selected query keywords as we use in this evaluation, is more comparable to our runs.

System	nDCG′	MAP′	P@10	BPref	Judged
MD-best	**0.345**	**0.139**	**0.161**	0.126	17.0%
MD-manual	0.298	0.074	0.079	0.050	15.5%
a0-baseline	0.247	0.099	0.062	0.116	8.2%
PyA0	0.270	0.115	0.066	**0.134**	8.7%

Table 3: Effectiveness comparisons between Approach Zero and the top effective system MathDowsers in the ARQMath2020 main task.

We conduct evaluations in a single-threaded environment, using the following hardware configuration: AMD EPYC 7601 CPU with 2 GiB memory. Query latencies are averaged over five independent runs and indexes are located on a SSD disk when they are retrieved for the first time. PyA0 runs use the Python package described in this paper.

Table 2 summarizes the performance comparison on the NTCIR-12 Math Browsing collection. We see that PyA0 provides a competitive MIR baseline that achieves good effectiveness on formula-only retrieval (better than Tangent-S in terms of BPref score). It has a similar level of effectiveness and efficiency as our previous baseline, offering a good balanced performance for a baseline in MIR research. As for mixed types of queries, Table 3 gives effectiveness scores for MathDowsers runs and our baseline (i.e., our submitted version to ARQMath2020). The precision metric (P@10) is easily affected by unjudged results so our P@10 scores are relatively low. However, other metrics are more stable on incomplete relevance data [5] and to that end, we see that the effectiveness of PyA0 is at a similar level as its counterpart MD-manual run, which uses the same query set. These results suggest that PyA0 can serve as a strong baseline for formula retrieval in a text context.

4 CONCLUSION

This paper introduces PyA0, a Python toolkit that provides an interface to the math formula search engine Approach Zero. Python bindings help users quickly get the system running on pre-built indexes, and the simple Python interface demonstrated in our provided code snippets will make our system more accessible for IR researchers and practitioners. Our experiments show that PyA0 can be used as a strong baseline for Math IR research, and that it features balanced structure-aware search performance with real-time structure query processing and effective search results for both formula-only retrieval and in searching mixed documents.

Given that Python has already been used extensively in machine learning toolkits, we believe that PyA0 will increase interdisciplinary cooperation between the MIR and machine learning communities. Many interesting work can also be anticipated, such as trying different approaches for embedding math semantics structures generated from Approach Zero, and applying sequence prediction to well categorized math tokens produced from Approach Zero to achieve math formula query auto-completion.

REFERENCES

[1] Kenny Davila and Richard Zanibbi. 2017. 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*. ACM, 1165–1168.

[2] Giovanni Yoko Kristianto, Goran Topic, and Akiko Aizawa. 2016. MCAT Math Retrieval System for NTCIR-12 MathIR Task. In *Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies*.

[3] Behrooz Mansouri, Shaurya Rohatgi, Douglas W Oard, Jian Wu, C Lee Giles, and Richard Zanibbi. 2019. Tangent-CFT: An Embedding Model for Mathematical Formulas. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*. 11–18.

[4] Shaurya Rohatgi, Jian Wu, and C Lee Giles. 2020. PSU at CLEF-2020 ARQMath Track: Unsupervised Re-ranking using Pretraining. In *International Conference of the Cross-Language Evaluation Forum for European Languages (Working Notes)*.

[5] Tetsuya Sakai and Noriko Kando. 2008. On Information Retrieval Metrics Designed for Evaluation with Incomplete Relevance Assessments. *Information Retrieval* 11, 5 (2008), 447–470.

[6] NG Yin Ki, Dallas J Fraser, Besat Kassaie, George Labahn, Mirette S Marzouk, Frank Wm Tompa, and Kevin Wang. 2020. Dowsing for Math Answers with Tangent-L. In *International Conference of the Cross-Language Evaluation Forum for European Languages (Working Notes)*.

[7] Richard Zanibbi, Akiko Aizawa, Michael Kohlhase, Iadh Ounis, Goran Topic, and Kenny Davila. 2016. NTCIR-12 MathIR Task Overview. In *Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies*.

[8] Richard Zanibbi and Dorothea Blostein. 2012. Recognition and Retrieval of Mathematical Expressions. *Int. J. Doc. Anal. Recognit.* 15, 4 (Dec. 2012), 331–357.

[9] Richard Zanibbi, Douglas W Oard, Anurag Agarwal, and Behrooz Mansouri. 2020. 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, 169–193.

[10] Wei Zhong and Hui Fang. 2015. *A Novel Similarity-Search Method for Mathematical Content in LaTeX Markup and its Implementation*. Master's thesis. University of Delaware.

[11] Wei Zhong, Shaurya Rohatgi, Jian Wu, C Lee Giles, and Richard Zanibbi. 2020. Accelerating Substructure Similarity Search for Formula Retrieval. In *European Conference on Information Retrieval*. Springer, 714–727.

[12] Wei Zhong and Richard Zanibbi. 2019. Structural Similarity Search for Formulas using Leaf-Root Paths in Operator Subtrees. In *European Conference on Information Retrieval (ECIR 2019)*. Springer.