Leveraging Formulae and Text for Improved Math Retrieval

by

Behrooz Mansouri

A dissertation submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in Computing and Information Sciences

B. Thomas Golisano College of Computing and
Information Sciences

Rochester Institute of Technology
Rochester, New York
July 2022

Leveraging Formulae and Text for Improved Math Retrieval

by

Behrooz Mansouri

Committee Approval:

We, the undersigned committee members, certify that we have advised and/or supervised the candidate on the work described in this dissertation. We further certify that we have reviewed the dissertation manuscript and approve it in partial fulfillment of the requirements of the degree of Doctor of Philosophy in Computing and Information Sciences.

Dr. Richard Zanibbi — Date
Dissertation Co-Advisor

Dr. Douglas W. Oard — Date
Dissertation Co-Advisor

Dr. Cecilia Alm — Date
Dissertation Committee Member

Dr. Yu Kong — Date
Dissertation Committee Member

Dr. Jimmy Lin — Date
Dissertation Committee Member

Dr. Dan Phillips — Date
Dissertation Defense Chairperson

Certified by:

Dr. Pengcheng Shi — Date
Ph.D. Program Director, Computing and Information Sciences

Leveraging Formulae and Text for Improved Math Retrieval

by

Behrooz Mansouri

Submitted to the
B. Thomas Golisano College of Computing and Information Sciences Ph.D. Program in Computing and Information Sciences
in partial fulfillment of the requirements for the
Doctor of Philosophy Degree
at the Rochester Institute of Technology

Abstract

Large collections containing millions of math formulas are available online. Retrieving math expressions from these collections is challenging. Users can use formula, formula+text, or math questions to express their math information needs. The structural complexity of formulas requires specialized processing. Despite the existence of math search systems and online community question-answering websites for math, little is known about mathematical information needs. This research first explores the characteristics of math searches using a general search engine. The findings show how math searches are different from general searches.

Then, test collections for math-aware search are introduced. The ARQMath test collections have two main tasks: 1) finding answers for math questions and 2) contextual formula search. In each test collection (ARQMath-1 to -3) the same collection is used, Math Stack Exchange posts from 2010 to 2018, introducing different topics for each task. Compared to the previous test collections, ARQMath has a much larger number of diverse topics, and improved evaluation protocol.

Another key role of this research is to leverage text and math information for improved math information retrieval. Three formula search models that only use the formula, with no context are introduced. The first model is an n-gram embedding model using both symbol layout tree and operator tree representations. The second model uses tree-edit distance to re-rank the results from the first model. Finally, a learning-to-rank model that leverages full-tree, sub-tree, and vector similarity scores is introduced. To use context, Math Abstract Meaning Representation (MathAMR) is introduced, which generalizes AMR trees to include math formula operations and arguments. This MathAMR is then used for contextualized formula search using a fine-tuned Sentence-BERT model. The experiments show tree-edit distance ranking achieves the current state-of-the-art results on contextual formula search task, and the MathAMR model can be beneficial

for re-ranking.

This research also addresses the answer retrieval task, introducing a two-step retrieval model in which similar questions are first found and then answers previously given to those similar questions are ranked. The proposed model, fine-tunes two Sentence-BERT models, one for finding similar questions and another one for ranking the answers. For Sentence-BERT model, raw text as well as MathAMR are used.

Acknowledgments

I am deeply grateful to Dr. Richard Zanibbi and Dr. Douglas W. Oard for their constant support. Dr. Anurag Agarwal, who provided me useful feedback on my research. I would like to express my sincere gratitude to Dr. C Lee Giles, Dr. Jian Wu and Shaurya Rohatgi with whom I collaborated on the MathSeer project.

I would like to extend my sincere thanks to Dr. Nicola Ferro and Dr. Kevyn Collins-Thompson who were my mentors during the SIGIR 2021 Doctoral Consortium [96].

Each member of the Document and Pattern Recognition Lab (DPRL) at Rochester Institute of Technology (RIT), including Abishai Dmello, Gavin Nishizawa, and Yancarlos Diaz, very often provided useful feedback and new interesting ideas that helped me improved my work.

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.

Dedicated to my father, who always supported me, and to my mother, who always encouraged me to do my best. Endless thanks to my three sisters for their support

I would like to thank Ricardo and Mohammad Sadegh for the insight they brought to me about doing a PhD. Finally, I would like to thank all my friends in Rochester for their support

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Math information retrieval is simply information retrieval where the user's information need involves math. There are several ways users might express their information need. Figure 1.1 shows a user with an information need about finding a limit. They could use an isolated formula as the query, a combination of formulas and text, or they could post a fluent math question on a community question answering (CQA) website. The goal is to find relevant information, in this case that helps to find the value for the limit. When users choose an isolated formula as the query, the task is called formula search and the goal is to find relevant formulas, hopefully associated with contexts that can help to address the user's information need. In our example, the related formula is in the dashed line in the retrieved answer.

Another example of math search is a real search session on a general search engine [106] in which a user issued the following queries:

- C(10,5)
- C(n,K)
- How can I select 5 items from list of 10 item
- Selecting k items from n items
- Solving C(n, k).

As can be seen, the user started with a specific formula, moved to a general one, and then perhaps as the search engine failed to support formula search or provided just general information about

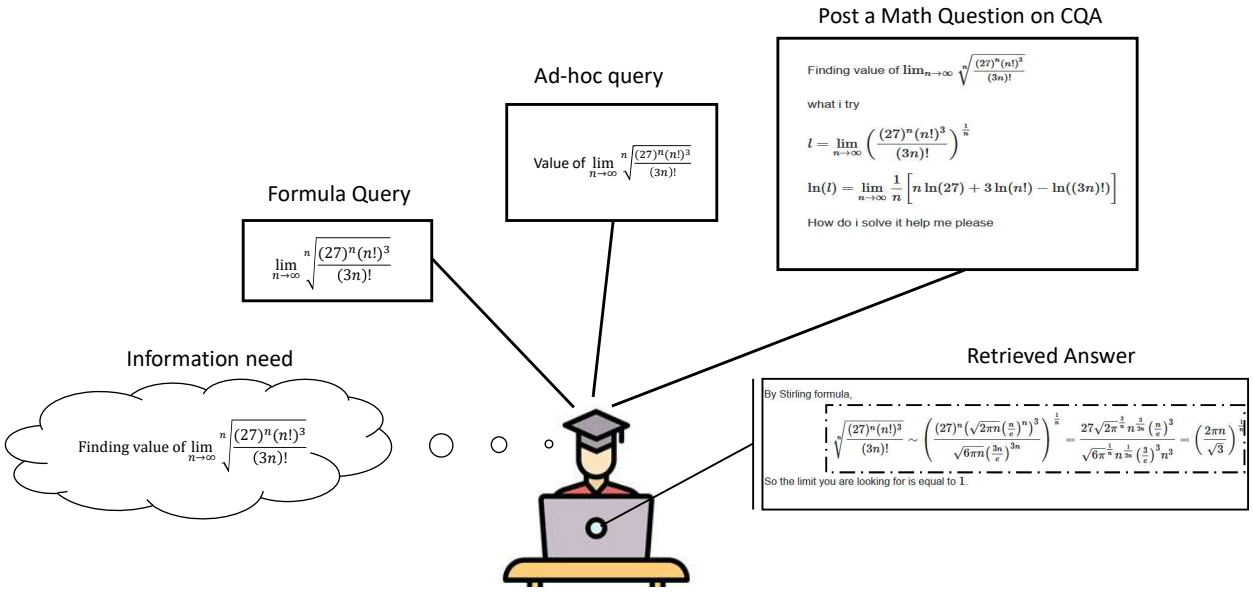


Figure 1.1: Example information need related to finding a limit. There are three different searches the user can do using formula or ad-hoc queries or posting a math question on a forum. The goal is to find relevant information, such as the answer shown on the right.

the formula, moved to ask the original question with words. Finally, the user used both a formula and text, and ended the search session by clicking on a web page, perhaps finishing the session with the information need satisfied.

There are platforms such as MathStackExchange[1] and MathOverflow[2] that are online communities allowing users to ask questions and post answers. Also, there are several search engines that support math search, such as SearchOnMath,[3] WolframAlpha,[4] Approach0[5] and MathDeck.[6] Each of these platforms provides an interface which users can express their mathematical information needs. For example MathDeck [39, 119], provides different input means for text+formula queries. Once the formula is inserted, the users can search. The users can also search for summary cards [40]. The cards contain a formula chip, a title, and a short, focused description. Card data is taken from Wikipedia. Figure 1.2 shows the MathDeck interface, where a user inserted $c^2 = a^2+b^2$ as the input query. The summary cards are shown, with the "Pythagorean Theorem" card being expanded. The

[1] `https://math.stackexchange.com/`
[2] `https://mathoverflow.net/`
[3] `https://www.searchonmath.com/`
[4] `https://www.wolframalpha.com/`
[5] https://approach0.xyz/search/
[6] `https://mathdeck.org/`

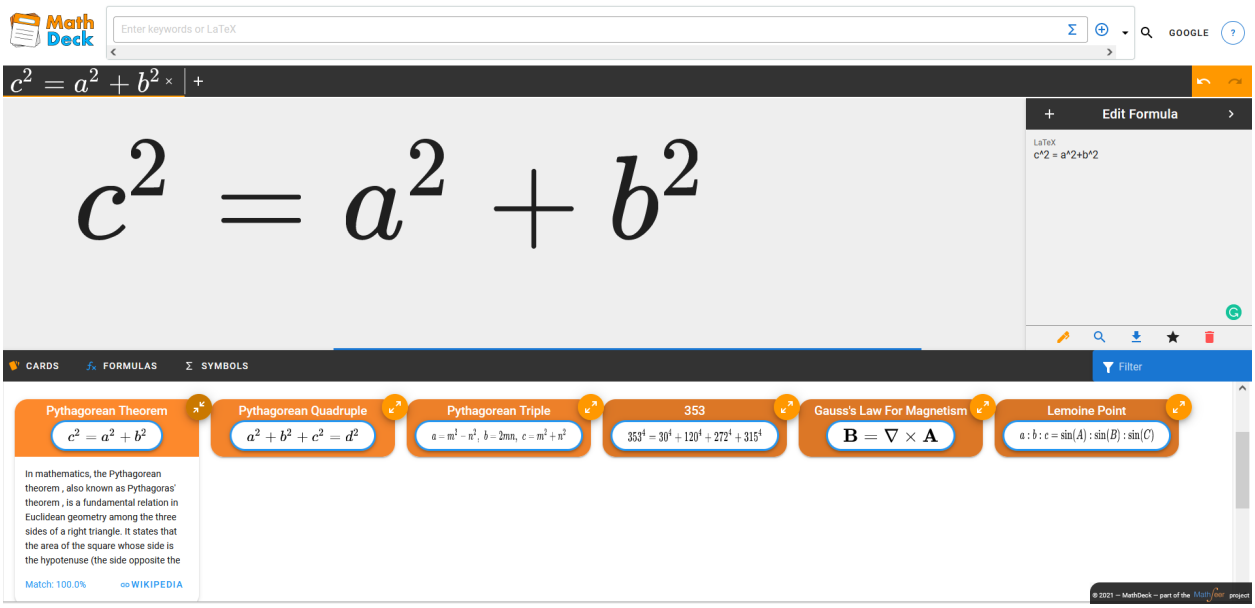


Figure 1.2: MathDeck search interface. In this example, the user is searching for the formula $c^2 = a^2 + b^2$. Wikicards are showing the relevant formulas retrieved from a set of cards.

users of this search engine can save their formula as a favorite to avoid the effort of rewriting it again. Also, each user has a profile that keeps a record of his/her searches. Users are also able to search their log history. Currently, MathDeck principally uses the Tangent-CFT [105] system (developed as part our research) for formula search.

Math information retrieval is more complicated than traditional information retrieval where user information needs are expressed with only text. Users can use math notation to express math. Math is highly structured, whereas text structure is linear [32]. Youssef [167] defines the objectives of math search as:

- **Math-awareness**. Math search systems should be able to recognize mathematical symbols and structures.
- **A natural math-query language**. The math query language used by the search system should be similar to what scientists and engineers use to write math.
- **Fine granularity of retrievable information units**. The retrieval units should be targeted toward fine-grained granularity of user need such as an equation, a definition, or a theorem. A larger retrieval unit is not desirable for a user.

- **Perfect ranking**. The search system should aim to maximize both recall and precision. The relevant hit(s) should appear before the non-relevant ones. Recall shows the ratio of relevant retrieved units to all relevant items, while precision shows how much of the retrieved units are relevant.
- **Useful highlighting**. The user should know why the retrieved unit is matched to the query. Highlighting is one solution for the larger units such as articles.

The foundation of math search is a growing body of information retrieval research in which the principal focus has been on matching mathematical notation [171]. Several challenges have been addressed, most notably mapping between surface forms and logical forms of mathematical notation [53], computing similarity based on logical forms [64], and linking mathematical notation with associated text [75]. This work has led to the deployment of specialized search engines designed to demonstrate the potential of this rapidly evolving search technology. For instance, "Search on Math" searches Web pages, providing a keyboard that users can employ to include formulas in their queries. There are also specialized search engines for mathematical content, such as Mathematics Stack Exchange which supports search over previously created content in a community question answering site focused on mathematics questions. Of course, users also employ general search engines when looking for information about mathematics, or in which mathematical concepts might be useful.

A key focus of mathematical searching is formulas. In contrast to simple words or other objects, a formula can have well-defined sets of properties, relations, and applications, and often also a 'result'. There are many (mathematically) equivalent formulas which are structurally quite different. Moreover, it is of fundamental importance to ask what information a user wants when searching for $x^2 + y^2 = 1$: is it the value of the variables x and y that satisfy this equation, all indexed objects that contain this formula, all indexed objects containing $a^2 + b^2 = 1$, or the geometric figure that is represented by this equation?

Despite the existence of math-aware search systems, we have limited insight into the characteristics of math searches. We still do not know much about user behavior in a math search session (i.e. a series of search activities that the same user does, such as issuing the first query, changing the query, and clicking on search results). Another challenge is the type of math queries users issue, which has not yet been investigated. For example, there is no information on how often users prefer to use a math question as their input query (known as a question query). Without knowing the users' behavior, designing a math-aware search system is not practical. One possible solution to this is to use the query logs of a search engine to explore users' behavior in math searches. Query

logs are valuable resources, keeping track of the users' and search engine's interactions.

Knowing the users' behavior can help us design a math-aware search system, but how do we know if our design is good? How do we know that one search system has better search results than another one? To answer these questions, we need test collections. In a test collection, the goal is to simulate a user and evaluate a search system. A user issues a query to a search system, then the search system returns a set of results, and based on the relevance of the results to the user's information needs, the users information need may be satisfied. A test collection usually consists of:

1. **A Collection of items to be found.** The search task is done over a collection, and a set of ranked results are retrieved from it.
2. **A Set of Topics.** A topic in test collection is like a user in the real world. They issue a query related to an information need. A topic typically includes a query together with some further description of the information need. In some test collections, that description may be manually generated, and in others, a description can be integrated into the query itself (e.g., a question query).
3. **Relevance Judgements.** Relevance indicates how well a retrieved item, such as a document, a paragraph, or a formula, addresses the information need of the user. In a test collection, multiple levels of relevance can be considered, such as full, partial, and non-relevant.
4. **Evaluation Protocol.** After knowing the relevance of results retrieved by a system, the next question is which system is better? What evaluation measures should be used, and how should they be applied? For this, a set of protocols known as evaluation protocol are defined.

One common way of developing test collections is through a shared-task. In shared-task labs, one or more search tasks are defined with the components necessary for a test collection other than the relevance judgements. Then participating search systems each provide their candidate results for each topic (for each task). Assessing all the candidates is hard due to financial and timing constraints. Therefore, a pooling strategy is defined to select the candidates to be assessed. For example, the top-k results for each participating system can be pooled for assessment. Assessors then create relevance judgements for those pooled candidates.

Despite the existence of math-aware search systems, efforts to enhance search quality for math still continue. Math-aware search is hard, as there can be both formula(s) and text in queries and

collection. Formulas are usually represented as tree (see Chapter 4, Section 4.1) where text has linear structure. Therefore, math-aware search can be considered as a multi-modal search task where there can be data with different modalities in queries and in the collection. One approach for multi-modal search is mapping different modalities to a unified space and searching over that unified modality. These models are known as mono-modal search systems. Another approach is to have a separate search model for each modality, and then combine the search results.

In this research, we focus on two math-aware search tasks. First, we study the contextual formula retrieval task. In this task, given a formula as the input, the goal is to find relevant formulas. A relevant formula is defined by the retrieved formula being associated with information that answers user's information need. We first consider three retrieval models, an n-gram embedding model, a ranking model with tree-edit distance, and a rank-SVM model (see Chapter 4, Section 4.3). These three models only consider formulas in isolation. We then introduce a mono-modal representation of text+formula, MathAMR (Math Abstract Meaning Representation) and develop a search model using this representation for the contextual formula search task.

Second, we investigate the answer retrieval task for math questions. We propose a two-step model, first finding similar questions to the input question, and then rank the answers that had previously been given to similar questions with different ranking functions. We use both raw text and MathAMR representations for this.

1.1 Research Questions

Here we summarize the main research questions we address in this dissertation.

1. How do users utilize existing general-purpose search engines for math-aware search, and what types of information needs can we observe in their query logs?
2. How should we (1) define relevance for contextual formula search tasks and (2) evaluate contextual formula search systems?
3. Can an embedding model (any model to map formulas to points in space) be used beneficially for the formula search task?
4. Can a unified, single representation of text and formulas be beneficial for contextual formula retrieval?

5. Can first finding similar questions and then ranking answers given to them be effective for the math answer retrieval task?
6. Can answer retrieval be performed effectively using a unified representation for text and formulas?

1.2 Contributions

Here we list our main contributions:

1. Conducted the first query log analysis on math searches and compared them to general searches
2. Developed new design considerations for math-aware test collections and search systems
3. Created an annotated collection of Math Stack Exchange question and answer posts for math information retrieval
4. Developed test collections for answer retrieval and contextual formula search. For each task:
   (a) Developed over two hundred topics
   (b) Used topic categories to increase diversity
   (c) Defined evaluation protocol
   (d) Designed pooling strategies
   (e) Developed new relevance definitions
5. Proposed an efficient n-gram embedding model for math formulas
6. Developed the current state-of-the-art formula search model using tree-edit distance ranking
7. Introduced a rank-SVM model for formula search
8. Designed MathAMR, a novel mono-modal representation for text + formulas
9. Developed a mono-modal search model with MathAMR for the contextual formula search and answer retrieval tasks
10. Designed a two-step search model for answer retrieval; first finding similar questions and then ranking answers given to similar questions

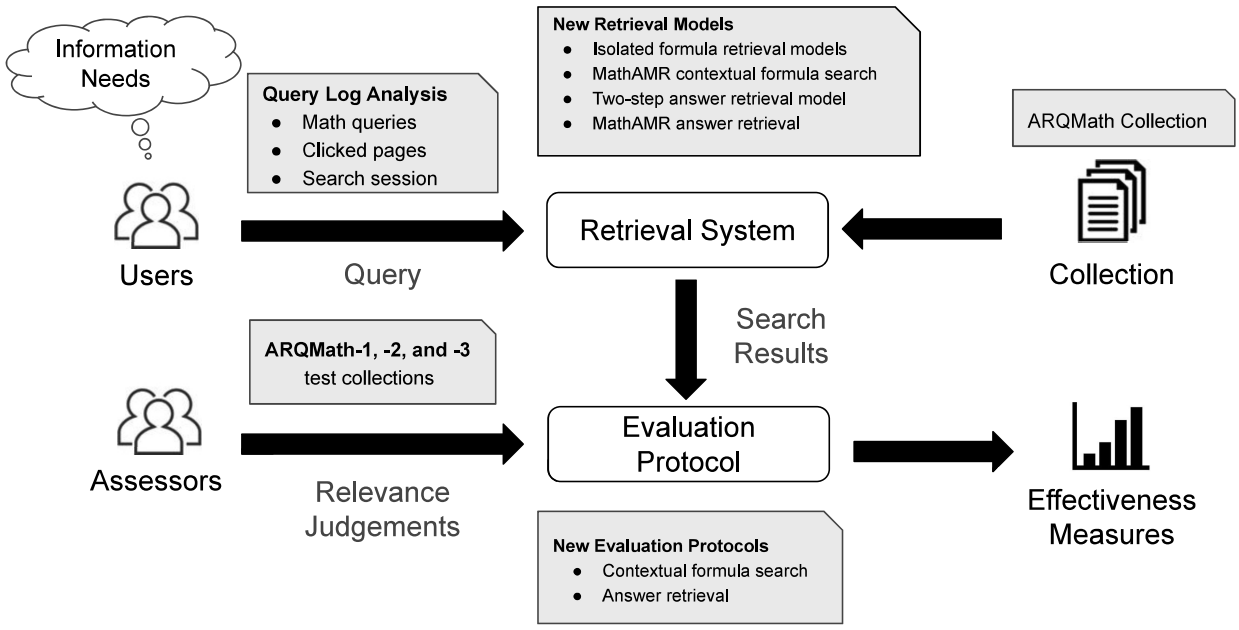


Figure 1.3: Contributions of this dissertation to each component of a math information retrieval research system.

Considering the main components of a search activity, Figure 1.3 shows our main contributions for each component.

1.3 Test Collections and Source Code

The test collections and source code that we have made publicly available include:

1. **Test Collections**: We have introduced the ARQMath test collections over a period of three years with hundreds of topics for the contexual formula search and answer retrieval tasks. We have made all the code to generate these test collection publicly available on ARQMath GitHub Repository.[7] This includes annotating formulas with identifiers inside the collection and providing different formula representations (Presentation and Content MathML). Also the test collections including the topics with their relevance assessments, are available online.[8]

2. **Retrieval Models**:

[7] https://github.com/ARQMath/ARQMathCode
[8] https://www.cs.rit.edu/~dprl/ARQMath/

(a) **Tangent-CFT.** An n-gram embedding model for math formulas.[9]

(b) **Learning to Rank for Math Formulas.** A learning to rank model for mathematical formulas that combines different formula similarity matching scores, including sub-tree, full-tree, and embedding.[10]

(c) **MathFIRE.** Math Formula Indexing and Retrieval for ElasticSearch[11], framework for indexing and retrieving formulas.

The details for these models can be found in Chapter 4, Section 4.3.

1.4 Publication and Co-Authorship

Table 1.1 provides a list of publications related to each chapter.

1.5 Outline

We have organized the chapters based on the steps needed to develop a math-aware search system; starting by reviewing characteristics of user behaviors in math-aware search, then test collections used for evaluating math-aware search systems (including collections created for this dissertation), and finally retrieval models for formula search and math-aware formula+text search.

The first step for developing any information retrieval system is to know what are the information needs and how users interact with the system. We need to know how users express their information need, the type of queries they issue, the resources they are looking for, and the way they use the search results. In Chapter 2, we study mathematical information needs and how users search for math. We review the existing work on user behavior for math-aware searches and systems supporting math search, and we present results from our query log analysis study.

After characterizing information needs, we move to the test collections for math-aware search tasks in Chapter 3. This includes tasks such as formula search, formula+text search, answer retrieval, and question answering for math questions. We review the existing test collection that predate our

[9] `https://github.com/BehroozMansouri/TangentCFT`
[10] `https://github.com/BehroozMansouri/LtRMathIR`
[11] `https://gitlab.com/dprl/mathfire`

Table 1.1: List of publications related to each chapter of this dissertation.

Title	Venue	Year	Authors
Chapter 1			
The MathDeck Formula Editor [39]	CHI	2020	Diaz, Y., Nishizawa, G., Mansouri, B., Davila, K. and Zanibbi, R.
Chapter 2			
Characterizing Searches for Mathematical Concepts [106]	JCDL	2019	Mansouri, B., Zanibbi, R. and Oard, D.W.
Chapter 3			
Finding old answers to new math questions: the ARQMath lab at CLEF 2020 [97]	ECIR	2020	Mansouri, B., Agarwal, A., Oard, D. and Zanibbi, R.
Overview of ARQMath 2020: CLEF Lab on Answer Retrieval for Questions on Math [173]	CLEF	2020	Zanibbi, R., Oard, D.W., Agarwal, A. and Mansouri, B.
ARQMath: A New Benchmark for Math-Aware CQA and Math Formula Retrieval	SIGIR Forum	2020	Zanibbi, R., Mansouri, B., Agarwal, A. and Oard, D.W.
Advancing Math-Aware Search: The ARQMath-2 Lab at CLEF 2021 [98]	ECIR	2021	Mansouri, B., Agarwal, A., Oard, D.W. and Zanibbi, R.
Overview of ARQMath-2 (2021): Second CLEF Lab on Answer Retrieval for Questions on Math [108]	CLEF	2021	Mansouri, B., Zanibbi, R., Oard, D.W. and Agarwal, A.
Effects of Context, Complexity, and Clustering on Evaluation for Math Formula Retrieval [101]	ArXiv	2021	Mansouri, B., Oard, D.W., Agarwal, A. and Zanibbi, R.
Advancing Math-Aware Search: The ARQMath-3 Lab at CLEF 2022 [99]	ECIR	2022	Mansouri, B., Agarwal, A., Oard, D.W. and Zanibbi, R.
Overview of ARQMath-3 (2022): Third CLEF Lab on Answer Retrieval for Questions on Math [100]	CLEF	2022	Mansouri, B., Novotný, V., Agarwal, A., Oard, D.W. and Zanibbi, R.
Chapter 4			
Tangent-CFT: An Embedding Model for Mathematical Formulas [105]	ICTIR	2019	Mansouri, B., Rohatgi, S., Oard, D.W., Wu, J., Giles, C.L. and Zanibbi, R.
DPRL Systems in the CLEF 2020 ARQMath Lab [102]	CLEF	2020	Mansouri, B., Oard, D.W. and Zanibbi, R.
DPRL Systems in the CLEF 2021 ARQMath Lab [103]	CLEF	2021	Mansouri, B., Oard, D.W. and Zanibbi, R.
Learning to Rank for Mathematical Formula Retrieval [107]	SIGIR	2021	Mansouri, B., Zanibbi, R. and Oard, D.W.
Contextualized Formula Search Using Abstract Meaning Representation	CIKM	2022	Mansouri, B., Oard, D.W. and Zanibbi, R.
Chapter 5			
DPRL Systems in the CLEF 2021 ARQMath Lab [103]	CLEF	2021	Mansouri, B., Oard, D.W. and Zanibbi, R.
DPRL Systems in the CLEF 2022 ARQMath Lab [104]	CLEF	2022	Mansouri, B., Oard, D.W. and Zanibbi, R.

work. We then address the need for developing a new test collection and introduce the ARQMath

test collections that we have built.

Chapter 4 focuses on the formula search task. For that, we first introduce the representations commonly used for mathematical formulas in systems. We then review and categorize the existing formula search models. We consider two tasks of isolated and contextual formula search. Finally, we introduce our proposed formula search systems for both isolated and contextual formula search tasks and present our experiment results on ARQMath test collection.

In Chapter 5 we review systems supporting formula+text search. We first call attention to the similarities of multi-modal information retrieval and formula+text search task. Then we summarize existing models for formula+text search and for answer retrieval tasks. Finally, we explain our approaches for answer retrieval for math questions and present experiment results on ARQMath test collection.

Finally, chapter 6 summarizes this dissertation. We review the research goals, the limitations of our work and we describe possible future work that our research has enabled.

Chapter 2

Characterizing Math Searches

To develop an information retrieval system, it is essential to know the users' behavior. There are different aspects of users' behavior that can be explored to improve the design of an information retrieval system. For example, it is essential to understand how often queries issued to a search engine are questions. A high ratio of question queries indicates the need to have a system that supports answer retrieval. This is not only useful for the design of search systems, but it is also beneficial for the design of test collections, as it indicates the importance of answer retrieval systems. Another example is to study how users change their input queries, known as query reformulation. Knowing what type of reformulation leads to a successful search can be beneficial for search engine when suggesting queries to the users.

Some specifications for developing a math search system have been proposed by researchers. Their validity is not yet confirmed. In this research, we conduct the first query log analysis on math searches, using query logs from a general-purpose search engine, Parsijoo.[1] We have studied three main aspects of users' search behavior: (1) queries, (2) clicked pages, and (3) search sessions. We have made a comparison between math and general searches and have shown that users' behavior is different in math searches. In the next section, we review user studies on math information retrieval and then introduce our study on query logs investigating how math search is done.

[1] `http://parsijoo.ir/`

2.1 Related Work

There have been a few attempts to understand how users search for math. Zhao et al. [176] interviewed a small group of people to provide insight into user requirements in math information retrieval. The study was conducted with 13 individuals affiliated with the Math Department at the National University of Singapore. Participants included two undergraduate students, seven graduate students, one professor, and three librarians. They were interviewed with questions regarding the kind of material they look for, what resources they use, and how they do the search.

The main findings of this work can be summarized as follows:

- The participant's information-seeking behavior for finding mathematical concepts can be categorized into three approaches: doing a search with a general search engine using math keywords, browsing math-specific resources like books to find material, and finding people they know, such as the student's advisor to locate information.
- With one exception, the participants had doubts about the value of having input and retrieval of formulas. The participants expressed several reasons, including the lack of mathematical expressions in their research, the inconvenience of entering expressions, or the high specificity of formulas.
- The users were looking for two categories of material: (1) information such as definition, explanation, example, and application, (2) resources such as a paper, tutorial, toolkit, and book.

This was the first study of math information retrieval user requirements, but the sample size was small, and all participants had a strong mathematical background. The study of Tausczik et al. [153] is less focused on information-seeking behavior, but it investigates how users collaborate to solve a math problem on MathOverflow.[2] They use three methods. First, the authors utilize quantitative analysis, relating collaborative acts to solution quality. Second, they apply grounded theory to 150 questions from MathOverflow to provide a taxonomy of collaborative acts, coded by whether a contribution provided information, clarified the question or, critiqued, revised, or extended an answer. Grounded theory is a research method to derive new theories by collecting and analyzing real world data in an iterative process. Finally, they conducted semi-structured interviews with 16 active MathOverflow contributors to better understand the collaborative acts, the roles they played

[2] `https://mathoverflow.net/`

in the collaborations, and how they contributed to the development of solutions. This work shows how combining quantitative and qualitative methods can build up a rich picture of a concept in crowd-sourced mathematics.

2.2 Query Log Analysis

At the early stages of our research, we did a study on users' search behavior using a Persian general search engine (Parsijoo) query logs [106]. This was the first study of math search based on actual queries and search sessions for users of a real-world search engine. The query log used in this study contains 27 million records of user interactions over two years. The Parsijoo transaction logs record communication between users and the search engine. Table 2.1 shows the transactions logs, which include Session IDs, submitted queries, query issue times, search type (e.g., Web or video), URLs for search results that users click on, click times, and session end times.

In this work, three editors compiled a list of mathematical concepts in Persian from two widely used college-level math textbooks (Thomas' Calculus - Thirteenth Edition (Volumes I, II) [155] and Essential Calculus with Applications (Volumes I, II, III) [147]) along with textbooks from the elementary through high school levels. The editors created a list of 681 mathematical concepts such as "Gaussian distribution," "Taylor series" and "cosine" (in Persian). The length of these concept phrases was between 1 to 3 words, with an average of 1.8 words per phrase. Queries containing any complete math concept phrase were included in the initial pool of math queries. At the moment, Parsijoo, does not support user profiling, so there is no information available about the users' knowledge of mathematical concepts, but the keywords covered concepts from elementary school to undergraduate degree. To have a set of math queries with high precision, phrases such as "division" that have both mathematical and more general meanings were not considered. Also, some mathematical concepts were used in the titles of movies or as the name of music albums. So in the cleaning steps, we removed queries containing words such as "music", "album" or "cinematic movies." In total, 392,586 mathematical query log records were extracted.

Each unique browser within the first connection is assigned a unique Session ID by Parsijoo that typically persists (using persistent HTTP) until that browser is closed, or the connection is lost. Some of the sessions are far longer than expected for a user to be working on a specific task, so these sessions are broken into smaller math search session units for the analysis. This is done by partitioning Parsijoo sessions into math search sessions whenever there is no transaction for 60 minutes or more and then removing all partitions that do not contain at least one math query.

Table 2.1: Parsijoo transaction logs for two search sessions with math queries. (Translated from Persian to English.)

Session ID	Query	Issue Time	Search Type	Clicked URL	Click Time	Session End Time
A33C14AC80CD4	double integral concept	2016-08-23 10:19:03	Web			
A33C14AC80CD4	double integral solving methods	2016-08-23 10:19:28	Web	fa.wikipedia.org	2016-08-23 10:19:44	
A33C14AC80CD4	double integral solving	2016-08-23 10:20:09	Web			
A33C14AC80CD4	double integral examples	2016-08-23 10:20:35	Web	riazisara.ir	2016-08-23 10:22:09	2016-08-23 10:57:14
A33C14ACB5698	cauchy schwarz inequality problems pdf	2017-02-05 16:49:13	Web			
A33C14ACB5698	cauchy schwarz inequality explained notes	2017-02-05 16:52:20	Web	kanoon.ir	2017-02-05 16:52:39	
A33C14ACB5698	cauchy schwarz inequality	2017-02-05 16:59:41	Web			
A33C14ACB5698	Tutorial on cauchy schwarz inequality	2017-02-05 17:19:43	Web			
A33C14ACB5698	Tutorial on cauchy schwarz inequality	2017-02-05 17:19:52	Video	kelasdars.org	2017-02-05 17:20:14	2017-02-05 18:01:30

This results in 69,014 math search sessions. Note that math search sessions need not start or end with a math query; they simply must contain at least one math search query. In order to compare math searches with general searches, we partitioned every session in the same way and then selected partitions at random until we reached the number of queries that we had started with (392K) when focused on math queries.

After extracting the math queries, the queries, clicked pages, and search sessions are studied and in the following sections, the findings are explained.

2.2.1 Math Queries

In this section, we focus on math queries in isolation and study their properties.

Content Type. Resources for math can vary from a paper on arXiv[3] in PDF format to a video like the ones on Khan Academy.[4] Therefore, different search models are needed, even if the queries are similar. The primary question one might ask is what sorts of content is it that users are searching for? As with many search engines, Parsijoo allows users to specify a vertical to search (e.g., Web, News, Video). One simple source for the desired content type would therefore be the request type field in the query log. However, that field is not very informative because 95.8% of math queries were issued as general Web searches with no more specific vertical selected. Users also will frequently specify the type of content they are looking for in the query, and Web search engines will typically parse and interpret such terms as soft content type specifications. we therefore also looked for content type terms in the queries. As Table 2.2 shows, a content type term was present

[3] `https://arxiv.org/`

[4] `https://www.khanacademy.org/`

Table 2.2: Frequency of common content type terms in math queries. Results are cumulative, starting with the most frequent content type (e.g., "PDF" is counted only in queries that do not contain "Tutorial").

Resource Type	**Math Query Percentage**
Tutorial	12.3%
PDF	9.7%
Video	8.4%
Download	6.1%
Notes	5.9%
PowerPoint	5.3%
Total	47.7%

surprisingly often in math queries. For resource types such as Video, a list of video formats such as .mp4, .avi, and .3gp was considered.

As shown in the table, nearly, half of the math queries have the content type specified in them. Note that this is a rather eclectic blend of types, some of which specify file types and others of which describe a document genre. We selected these terms that we believe to be indicative of content type manually after inspecting enough of the query log to have confidence that we had seen the most frequent content type terms.

Tutorials (which might suggest text, slides, video, or any combination of that) were the most frequently requested content type (12.3%), while PDF and Video were the next most popular requests (9.7% and 8.4%, respectively). From this one can conclude that Math search engines will ultimately need to be able to do more than just search HTML; video search, and specialized handling for specific formats such as PDF and PowerPoint that might include math in either text or image form, would be useful. There may moreover be some scope for personalization [41] here since although user preferences are not always explicit in their queries, to the degree those preferences are persistent across sessions the search engine might learn to bias the results towards content types that have previously been requested.

Query Length. Queries are typically short, and that can make it challenging for a search engine to understand the user's search intent [10]. For example, the set of general Parsijoo queries that we assembled as a basis for comparison has an average length of 3.4 words. The average math query, by contrast, is nearly twice as long, at 6.7 words. Math queries are much longer on average:

indeed, the third quartile of the query length for general queries is about where the first quartile is for math queries. 46% of the math queries are between 4 to 6 words long, producing a right-skewed distribution for math queries.

Cut-and-Paste Queries. One user behavior that can lead to issuing long queries is text reuse, when searchers copy long passages of text into the query box. To check for this, we selected the top-2000 most frequent unique math queries with length ≥ 20 and issued them to Parsijoo. By using quotation marks to obtain exact matches on longer sequences of text, 49.8% of these queries found documents containing exactly the query text, making it likely that these queries were entered into the query field using cut-and-paste from some document in the indexed collection.

Verbosity. Another possible reason for having longer queries is the presence of unnecessary words (a phenomenon referred to as verbosity [38]). For example, the query "How can I expand Taylor series?" from the query log, is considered verbose. Bendersky and Croft [12] have shown a correlation between users' click behavior and the query length – generally, the effectiveness of retrieval decreases as query length increases. To check for verbosity, we considered the top-2000 most frequent unique math queries with length greater than 4 (i.e., longer than the average query length over general search queries). Three annotators then labeled the queries as verbose or not. The annotators were given the definition of the verbose query as a "long query with words that are less informative and can be ignored without changing the purpose of the query", along with 30 verbose query examples. To measure the inter-rater agreement between annotators, we used Fleiss' kappa statistic [44]. Overall, the level of agreement was 0.79, which represents a high agreement between annotators. 17.8% of the queries under study were tagged as verbose.

Question Queries. Question queries are long and add verbosity to the queries. To check if that is the reason for having longer and more verbose queries in math searches, we explore the use of actual questions as math queries. To extract question queries, we used the approach of Zahedi et al. [169], where they studied how Persian question queries are posed to a search engine. We use their keyword-based approach, creating a list of Persian question words to identify math queries that are question queries. Our study found that 18.4% of math query instances (and 19.8% of unique math queries) are issued as some form of a question. This number is much higher than the 1.8% reported in [169] for general queries, and close to the 17.8% of queries that the annotators marked as verbose. Indeed, 90% of question queries were marked as verbose, so a very substantial number of the queries marked as verbose were marked in that way because they contain a question structure that is not related to the content of the query. Question type analysis can, however, be used to refine search engine results, for example, returning different results for "Why" questions

Table 2.3: Frequent question words and accompanying terms in math question queries. 'Other' question words include Which, Where, Whether, and Who. '?' indicates no question word is used, but a question mark is included. Query percentages are of question queries; Accompanying term percentages are for that question type.

Queries	**Cue**	**Accompanying Term**
69.5%	What	Formula (60%), Equation (11%), Used for (9%)
13.8%	How	Prove (51%), Exact (10%), Accurate (5%)
6.2%	Why	Is (20%), Not (15%), Correct (7%)
3.9%	When	Use (52%), Apply (21%), Consider (4%)
4.3%	Other	Formula (32%), Equation (12%), Invent (9%)
2.3%	?	True (14%), Answer (12%), Principle (6%)

than for "How" questions [156].

To get better insight into math question queries, the distribution of question cues and associated terms in math question queries are presented in Table 2.3. The table shows that some question words are frequently paired with other words. For instance, "What" is the most frequent question word, used in 69.5% of math question queries. "What" is often followed by the words "Formula" or "Equation," suggesting that users are looking for mathematical notation. Such information can be used in query suggestion, auto-completion, and query expansion. For instance, when a user issues the query "What is Poisson distribution ..." probable words for auto-completion are "formula" or "equation". Another example is the question word "How," which was more than half of the time paired with the word "Proof", indicating that users want to know how to prove a mathematical statement. Question type analysis is a subtask studied in the broader field of Question Answering, a complex task that requires a retrieval system to correctly answer questions posed in natural language.

Overall, the analysis shows that math queries are longer than general search queries due to the use of copy-and-paste queries, formulation of a substantial number of queries as questions, and some degree of verbosity. These results suggest that multiple complementary strategies may be needed to produce the best results for longer queries.

Query Uniqueness. One advantage that search engines often seek to leverage is that a small number of identical queries are often issued by many different users. This phenomenon has important implications for both search quality (since the clicks of early users can be used to predict

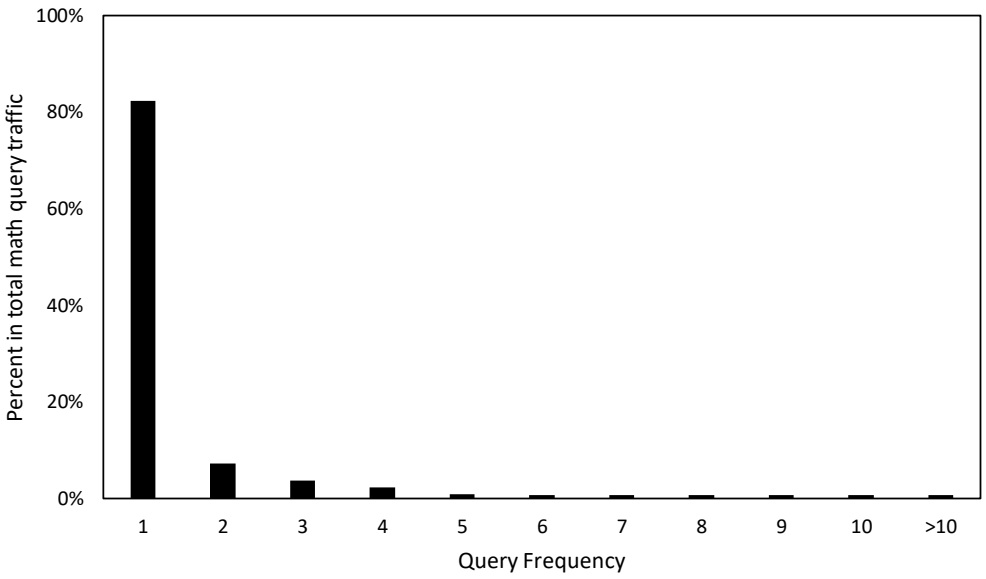


Figure 2.1: Distribution of math query frequencies.

what later users are most likely to want to see) and efficiency (since responses to frequently issued queries can be cached, which can dramatically reduce time-consuming disk and network activity). To understand how often math queries are repeated, we plotted the distribution of math query instances by frequency over the two-year period in Figure 2.1.

As can be seen from this figure, more than 80% of the math query instances appear in the query log only once. One implication of this result is that query suggestion techniques that rely on rich data about query frequency are not likely to work well in this setting. Recently, researchers have tried using side content as a basis for query suggestion in a manner similar to pseudo-relevance feedback [85, 149]. Mitra and Craswell [113] have also proposed a vector representation approach for query auto-completion of rare queries. Similar methods might be tried to produce better query suggestions for relatively infrequent math queries.

2.2.2 Clicked Pages

In this section, we review the clicked pages for math queries.

Frequently Visited Websites. First, we consider which Websites users view during math search. Patterns of this type can be leveraged to bias search results in favor of sites that are known to be useful. The top-10 Websites (i.e., the top-level domain of the Web page) clicked on in math

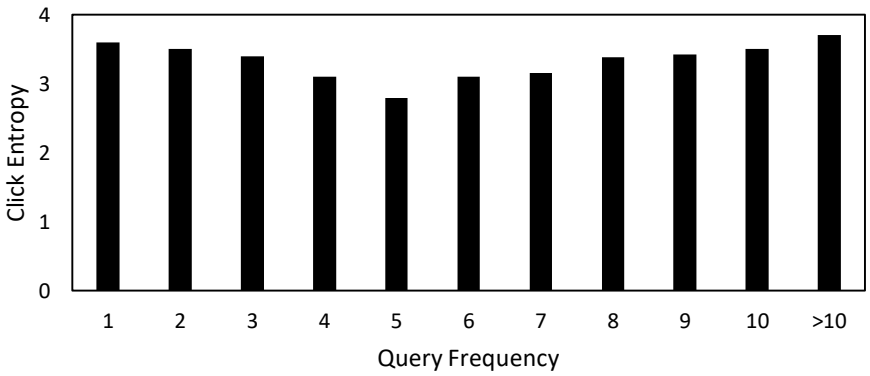


Figure 2.2: Average click entropy for math queries by their frequency.

search sessions, account for more than 25% of all page clicks. Considering these Websites, with the exception of Wikipedia, all of them focus on mathematical or scientific content. Wikipedia, of course, includes many pages with that type of content as well.

Click Entropy. Going further, we investigate the variability of Web pages clicked in math searches. There were two reasons to study this. First, the top-10 clicked sites account for more than a quarter of clicked pages, which suggests that users prefer certain Web sites for math search. The second reason is to check if the diversity of resources specified by users will lead to diversity in clicked pages. To measure the diversity of clicked pages, we used the click entropy introduced in [41] as follows:

$$S(q) = -\sum_{URL\ u\in\ U(Q)} P(u|q)\log_2 P(u|q) \tag{2.1}$$

Where $U(q)$ is the set of URLs clicked by users after issuing query q and $P(u|q)$ is the probability that the URL u is clicked by users when the search results for query q is shown. The results obtained show the entropy for the average math query is 3.33, which is quite close to the average click entropy of 3.18 for the general queries. Figure 2.2 shows the click entropy for unique queries stratified by frequency.

For unique queries issued just once, the entropy is about 3.4. Converting entropy to perplexity by taking the antilog, this equates to an average of $2^{3.4} = 10.6$ clicks per query which, for singleton queries, are by definition uniformly distributed. For instance, for the query "prove that the diagonals of a parallelogram bisect each other using vectors?", a user viewed 8 different pages to find the information they were looking for. As queries are reused, position bias and perceived content quality effects result in greater probability mass accruing on some queries, causing a consistent reduction in entropy. However, as queries become even more common – here, once unique queries

have been issued more than five times – the entropy begins to climb. Common queries are often short, and short queries are often ambiguous, so one possible explanation for this observed effect is that as queries become more commonly issued, a point is reached where the ambiguity effect (which tends to increase entropy) overwhelms the quality and position bias effects, thus resulting in a net increase in entropy. For frequent queries, different users preferred different pages. For instance, for the query "triangle inequality", the click entropy was 4.7. Examining the most clicked-on URLs, there appear to be different user intentions; different users were looking for pages related to the definition, proof, equation, or application of this concept. Diversity ranking techniques that use result clustering to maximize the coverage of alternative interpretations or alternative facets of a query [27] can be a helpful response to the ambiguity, and thus are potentially useful for more commonly issued queries.

No-Click Queries. One way to study failure is to analyze the characteristics of queries that led to no clicks before the next query was issued (or the end of the session), which we call No-click queries. Because we expect long queries to be challenging, we chose to look at the distribution of query length (in words) both for queries in general and for No-click queries in particular. We consider query lengths between 1 to 30 words and studied the distributions for general queries, all math queries, and No-click math. While the distribution of both general and all math queries fall off as the query length increases beyond some point, the distribution of no click math queries has a longer tail. So we see that math queries are long, and that long queries are often unsuccessful. To get a clearer view, we randomly selected 100 No-click queries of length 5 or 6 (a range where there were also many successful queries) and examined them to see if we might guess the reason why there had been no click. We found three plausible explanations for why users are not clicking on results for queries of that length. One is that the users may be unsure about the mathematical concept they are searching for, with the user's initial query being different from the final query, mostly moving from more general to more detailed and specific mathematical concepts. Consider one search session where the initial query was "examples for math series expansion," reformulated to "Series expansion methods" and ending the session with the query "Maclaurin Series." Another example is a search session starting with the query "area and perimeter of geometric shapes," where the top results are for general shapes such as circles, rectangles, and squares. The user then made the query more specific by issuing the query "area and perimeter of shapes with multiple sides." The last two queries issued by this user were, "area and perimeter of Octagon" and "area and perimeter of Pentagon." Another type of No-click query that we observed resulted from the user requesting exact matches (by using quotation marks). For instance, one user first issued the query "Cholesky decomposition and Newton's method comparison" and then reformulated the initial query to the new query "matrix inversion solutions" and finally posed the query "comparison of

matrix inversion solutions" with the entire query in quotation marks. That did not work well. Finally, a third apparent cause for No-click queries was invalid input, either because of spelling or because of deficiencies in vocabulary. For example, there was a search session where the initial query was "the median value theorem for integrals." However, the correct mathematical concept is "the average value theorem for integrals." Another example is a session where the user meant to issue the query "Kruskal's algorithm in graph theory," but instead of the word "Kruskal" used the word "Truskal." That user finished his/her search session with no click. All three of these cases point to the value of query suggestion. Considering the first scenario, suggesting related queries for more specific mathematical concepts could help guide users to desired pages faster. For exact-match queries and invalid input, perhaps query auto-completion might help users to avoid entering No-click queries in the first place.

2.2.3 Math Search Sessions

So far, we have explored queries and clicks individually, but together these two phenomena form a search session. Users start the search activity with an intent in mind, they can issue multiple queries and change their search strategy to achieve their goal. The sequence of queries and clicks for this often-complex search process is called a "search session." In this section, we focus on math search sessions from the perspectives of user effort, math query refinement, and failed searches. We then examine some long math search sessions to obtain additional insight into math search behavior.

User Effort. One simple way of quantifying user effort is to look at the number of queries in a session. Other measures can be the search duration, defined as the time between the first query and the last click of a session, or the total number of clicks. As Table 2.4 shows, all of these measures are markedly greater for math sessions than for general search sessions. On average, math sessions are more than three times longer, both in duration and in the number of queries, than general search sessions; the average number of clicks is about two and a half times greater. Some of this might be attributed to math searches being predominantly informational, using Broder's query type taxonomy [22], whereas the higher prevalence of navigational queries in general Web search would skew the average for general searches toward shorter sessions. Indeed, it may be that exploratory search [161], a particularly challenging type of informational search, is more common for math, since one common use case we have observed for math search is math learners of school age who are working on assignments.

User Satisfaction. Regardless of the cause, searchers seem to be working pretty hard on their

Table 2.4: User effort metrics for math vs. general search. The mean and standard deviation for each measure is shown.

Session Type	Queries (μ, σ)	Duration (mins) (μ, σ)	Clicks (μ, σ)
Math	(5.82,1.62)	(10.42,5.58)	(3.28,1.03)
General	(1.86,0.90)	(3.11,2.89)	(1.32,0.94)

Table 2.5: User satisfaction metrics for math vs. general search.

Session Type	Zero-click	Click-final	Sat-click
Math	23.9%	25.2%	60.7%
General	10.5%	42.4%	82.2%

math searches. But the question is are they satisfied with the search outcome? To answer this question, we looked at what Kim et al. have called "sat clicks" [70]– interaction patterns that suggest (fairly reliably, when viewed in aggregate) user satisfaction. Specifically, we quantify three outcomes that are expected to be informative with regard to user satisfaction:

- **Zero-click sessions**. Sessions in which the user did not click on any search engine result. This can be interpreted as clear evidence of dissatisfaction.
- **Click-final sessions**. Sessions where the user ended the search session by visiting a Web page, suggesting that the user's information need may have been satisfied.
- **Sat-click sessions**. Sessions in which the user stays on a page for more than 30 seconds, suggesting that at least some part of the user's information need may have been satisfied. Note that many click-final sessions will also be Sat-click sessions since a Click-final session in which the browser remains open for 30 seconds is also a Sat-click session.

Table 2.5 compares these satisfaction measures between math and general search sessions, showing the percentage of sessions in which each outcome occurs. From these results one can reasonably conclude that math search users are not just working harder, they are also getting worse outcomes. More than twice as many math search sessions result in no clicks at all, a bad outcome, markedly fewer math search sessions are Click-final, and even Sat-clicks are less common for math search sessions.

Query Reformulation Query reformulation has been widely studied, both with the goal of improving retrieval effectiveness [52] and with the goal of supporting effective reformulation [60]. Seven types of query reformulation are considered in our work:

1. **Reordering words.** Words from the previous query are re-ordered (e.g., Series Taylor → Taylor Series). This type of reformulation might be motivated by the user seeing the words in some other order in a document.

2. **Removing words.** Removing at least one word from the previous query. This is often associated with a generalization strategy in which a user elects to search for some broader, and thus less fully specified, concept (e.g., shortest path in graph with Prim → shortest path in graph).

3. **Adding words.** At least one word is added to the previous query (e.g. optimization method → optimization newton method). In contrast to removing words, here the user makes the query more specific.

4. **Substituting words.** Having the same number of words, with at least one word in common with the previous query (e.g. Standard normal distribution → general normal distribution). This behavior can occur when refining a previous query, or exploring related concepts.

5. **New query.** There is no common word with the previous query (e.g. Taylor Series → Fourier Transform). This situation sometimes occurs where the user's goal changes, but it might equally well reflect an iterative strategy in which the user is working through a set of possibilities without having changed their goal.

6. **Multi-reformulation.** Here more than one of the reformulation types defined above occurs (e.g., Taylor Series Expansion Example → Taylor Series Formula: "Expansion" is removed, and "Example" substituted by "Formula").

7. **Revisiting.** The user returns to using a query they previously issued in the same search session. This might occur when the user abandons a strategy and wants to start over from a known point with a different strategy, or it may simply reflect the user satisfaction by using a previous query to return to some document that they had previously seen, perhaps after learning that no better document can be found.

Every consecutive pair of queries within the same session defines a reformulation, and these reformulation types are easily detected with simple string edit measures. A total of 83.6% of the math

Table 2.6: Math query reformulation frequencies, and percentage of reformulations leading to user clicks by type.

Reformulation Type	Frequency	Led to click
Substituting words	32.1%	62.9%
New query	21.5%	44.7%
Multi-reformulation	15.2%	58.3%
Adding words	13.5%	17.4%
Removing words	9.6%	18.3%
Reordering words	4.3%	12.9%
Revisiting	3.8%	15.9%

search sessions contain more than one query, and thus at least one reformulation. Table 2.6 shows the frequency of each reformulation type among sessions containing at least one reformulation, along with a simple click measure for the quality of whether the reformulation was useful.

As the table shows, substitution was the most common reformulation type (accounting for nearly one-third of all reformulations), and the most effective reformulation type (leading to a click nearly two-thirds of the time). Multi-reformulations, 76% of which included at least one substitution, were also nearly as effective as substitution alone (at least at finding something to click on). Reordering words and revisiting queries submitted earlier were the least common reformulations, and also the least successful. These results suggest that as query interfaces for math search become more capable, users might benefit not just from the ability to enter equations, but also from the ability to easily reformulate equations that they have entered in earlier queries. These insights might also help with the design of query suggestion techniques, which today often focus more on adding words (e.g., for auto-completion) than on substitution. We also looked at how often users switch from using search queries to asking questions, given the high frequency of question queries in math searches compared to general searches. Liu et al. [90] had observed that users unsatisfied with their search results sometimes changed their queries into a question. However, in 40.3% of cases where users posed a question, the question query was the initial query issued in the search session. For instance, in one search session, the user's initial query was "how to prove a relation is symmetric." This session was concluded after visiting two of the returned pages. From this relatively large prevalence of question-initial search sessions, one can speculate that users may be adopting this strategy because they expect the additional context to help the search engine to return better results.

Long Sessions As shown, users submit more queries and spend more time searching for math compared to general searches. To get a sense of the most extreme cases, we examined the ten math search sessions with the largest number of queries to try to determine why users are issuing more queries. Overall, in these sessions two types of behavior were noted: 1) the user is trying to solve a set of mathematical problems (perhaps homework), or 2) the user is trying to fully understand a mathematical concept. The largest number of math queries issued by a user in a math search session was 26. The user is probably a student working on geometry homework as different math queries, each related to one of five geometry concepts discussed in high school, were issued. The user first apparently tried all of the questions using copy-and-paste. For two out of the five geometry concepts, (s)he clicked on Web pages after issuing the copied text (which was in the form of a question), and neither of those two queries were reformulated. This suggests that the user may have found the answer to the question. However, for the three other queries, after not being satisfied with preliminary search results, the user tried to issue different queries to learn the geometry concept. By starting from the basic concept, the user tried different reformulations to locate more complex material on that concept. Overall, the user finished searching for the first four concepts with a click, but seems not to have been satisfied with results for the last concept as no clicks resulted.

Six more of the ten longest sessions seem to have been users looking for answers to certain math problems. In the other three very long sessions, users seemed to be trying to understand a mathematical concept in detail, continuing the search even after visiting Web pages. For instance, in the third-longest math search session, 20 queries were issued by a user concerning "trigonometric identities." In this search session, a 'new query' reformulation occurred 15 times as the user tried different queries, including "sine formula", "tangent" and "cotangent." The analysis of these ten long math search sessions suggests that math searches can be long for different reasons, and that it might therefore be useful for math search engines to include functions for inferring user intent [59].

2.3 Summary

While there are ideas about how a math search system might be used, there are a few instances of research on that topic. Before our query log analysis, there was only one lab study investigating the users' requirements in math search [176]. Our research was the first query log analysis for math search carried out using a general-purpose search engine.

The main finding is that users put in more effort and are less satisfied with math search. These

findings have important implications for future math-aware search system design. Based on our observation of the specified content type in math searches, nearly half of the math searches demand other resources than a web page, for which we recommend diversity of content in search results.

We are now aware that math search systems should support question queries and should anticipate longer queries than general searches. In long queries, around 50% of queries were copy-and-paste (copied from an online resource and pasted as a query). It is possible that exact matching for long queries may be effective. Another possible reason for long queries is verbosity, which is common in math searches, and search systems should tackle this issue. To the best of our knowledge, there is no existing research on handling verbose math queries. More than 80% of math queries are unique in the query log that we studied. This shows perhaps query suggestion techniques based on users' search logs are less effective for math searches.

In math searches, there are commonly clicked search results pages. In our study, 10 websites accounted for more than 25% of all page clicks. Our study on click entropy showed that both unique and frequent queries have high click entropy, suggesting that diversity in math search results is important. This is also supported by our analysis on users' behavior in math search sessions. Several of the longest search sessions appeared to be searching for solutions to a set of homework problems, but others seemed to focus on trying to understand a mathematical concept in detail.

Our finding in this Chapter provided new ideas for developing both test collections and math-aware search systems. In Chapter 3 we introduce the ARQMath test collection, with answer retrieval task developed based on the finding that questions are common math queries. Chapter 4 then introduces our search models such as Tangent-CFT [105] that were developed based on the observation that more than 80% math queries are unique and issued only once.

Chapter 3

Test Collections for Contextual Formula Search and Math Question Answer Retrieval

To study the effectiveness of math search systems, we need standard benchmarks with which systems can be compared. Evaluation of math search systems is challenging; to develop a test collection, several aspects need to be considered, including query selection, the technical complexity of the documents in the collection, relevance definitions, and evaluation protocols. It is important that the defined tasks simulate real user search behavior. For example, a test collection might use a set of lab-generated queries that are expected to be similar to real users' queries, or it can use the real queries that are issued by users. Another aspect related to the queries can be their quantity and diversity. Having a higher number of queries can help evaluate systems better. If queries have different categories, having queries with a representative distribution can help better discriminate between systems.

Collections can have documents from different resources, each having a different technical level. The technical level can influence the assessment process. For highly technical documents, experts in the field are needed to assess the content. Moreover, proper training for assessment is needed. Related to assessment, it is important that each task has a clear definition of relevance, especially if there are different relevance degrees such as relevant, partially relevant, and not relevant. The assessors need clear instructions to distinguish between these categories. Relevance definition can be different based on the task design, and therefore when moving from one test collection to another,

one can see changes in the definition, which can affect system evaluation.

Different evaluation measures can be used for the systems' effectiveness. Some of the measures consider the top-k returned results, such as Precision@10, and some of them look at the whole list of retrieved results, such as nDCG. Also, it is important to decide how the hits that are not assessed should be treated. One approach is to consider them as not relevant, while another approach is ignoring them.

This chapter introduces the available test collections and evaluation measures used in previous math-aware search evaluation. Prior to our work, shared task evaluation were introduced in NTCIR (NACSIS (National Center for Science Information Systems) Testbeds and Community for Information access Research project)-10, -11, and -12. We explore their tasks, data sources, queries, and the evaluation protocols used in them. Then, we introduce the ARQMath test collections for math information retrieval that we have developed.

3.1 Related Work

An earlier effort to develop a test collection started with the Mathematical REtrieval Collection (MREC) [94], a set of 439,423 scientific documents that contained more than 158 million formulas. This was initially only a collection, with no shared relevance judgments (although the effectiveness of individual systems was measured by manually assessing a set of topics). Therefore, this collection was commonly used to evaluate system performance (speed). The Cambridge University MathIR Test Collection (CUMTC) [151] subsequently built on MREC, adding 160 test topics derived from 120 MathOverflow discussion threads (although not all queries contained math). These test topics are selected from sub-parts of question from the 120 threads. The criteria for selecting these threads were: (1) not being too broad or too vague, (2) the accepted answer addressing all the sub-parts of the questions and being available in MREC documents.

In CUMTC, 184 relevance judgements are provided for the topics which are the accepted answer(s) for the selected questions on MathOverflow. Relevance is determined using two criteria: (1) *totality:* a resource is *total* if it contains all information for the sub-questions in a topic, and *partial* if only one part is addressed, (2) *directness:* a resource is *direct* if the answer can be derived with a little intellectual effort reading the text, otherwise it is *indirect*. The majority of topics (81%) have only one relevant document, and 17.5% have two relevant documents. To evaluate systems, mean average precision (MAP) is used. Average precision is the average of precision at each relevant

document retrieved.

There are three shared task evaluations focusing on math-aware searching introduced in NTCIR-10 [3], -11 [4] and -12 [170]. To the best of our knowledge, in NTCIR-10 [3] the first shared task on math-aware search was introduced, considering three scenarios for searching:

- Formula Search: find similar formulas for the given formula query.
- Formula+Text Search: search the documents in the collection with a combination of keywords and formula queries.
- Open Information Retrieval: search the collection using text queries.

In addition to the main math search task, there was a subtask in NTCIR-10, math understanding. The goal of this task was to extract natural language descriptions of mathematical formulas from a document to help with their semantic interpretation. For each formula, participants could return a full or a short description. For example, for the query $log(x)$, a full description is "a function that computes the natural logarithm of the value x", and "a function that computes the natural logarithm" or "a function" could be short descriptions.

NTCIR-11 [4] considered the formula+keyword search task as the main task and introduced an additional task called the Wikipedia open subtask, using the same set of topics as the main task with a different collection and different evaluation methods. Finally, in NTCIR-12 [170], the main task was formula+text search on two different collections. The second task was Wikipedia Formula Browsing (WFB), focusing on formula search. A "Similarity task" (*simto*) was another task where the goal was to find formulas 'similar' (not identical) to the formula query. Here, we discuss the differences between these test collections. For each section, we consider formula search and formula+text search, separately.

3.1.1 Queries and Documents

Documents. Collections used to evaluate math-aware search have come from different sources. NTCIR-10 relied on 100,000 technical papers from arXiv, including papers from mathematics, physics, and computer science. NTCIR-11 and -12, considered 105,120 papers from arXiv. Each arXiv document was divided into paragraphs, forming a total of 8,301,578 paragraphs, which are the retrieval units.[1] Due to the high technical complexity of arXiv papers, NTCIR-11 and 12 added

[1] retrieval unit is the object such as document, paragraph, or formula retrieved by an information retrieval system

Wikipedia articles, however, the processing step and the number of articles were different.

Formula Search Queries. Formula queries in the test collections differ in their number, diversity, and nature. *Wildcard* queries contain symbols that may be replaced by variables and/or subexpressions. For example, in the formula query:

$$\frac{?f(?v+?d)-?f(?v)}{?d}$$

there are three query variables: ?f, ?v and ?d. The query matches the argument of the limit in formula

$$g'(cx) = \lim_{h \to 0} \frac{g(cx+h) - g(cx)}{h}$$

by replacing g for ?f, cx for ?v, and h for ?d. *Concrete* queries are complete formulas without wildcards.

In NTCIR-10, 21 formula queries were chosen by the organizers for arXiv papers. 18 of these 21 queries included wildcards; 3 were concrete queries. A search scenario and query-specific judgment criteria were specified for each formula query.

In NTCIR-11 (*simto* task), a total of 100 queries were selected randomly from Wikipedia pages. 59 of these 100 queries included wildcards; 41 were concrete queries. NTCIR-11 was a known-item retrieval task, so no relevance judgment criteria were specified.

In the NTCIR-12 Wikipedia Formula Browsing (WFB) task, there were 40 queries, divided into 20 concrete queries and 20 wildcard queries. The wildcard queries were created by replacing one or more subexpressions in each concrete formula query with wildcards. For example, the query $O(mn \log m)$ also appears as $O(\textbf{*1*} \log \textbf{*2*})$, where ***1*** and ***2*** represent two independent subexpressions.

NTCIR-12 had another task related to formula search called *simto*, with 8 queries. The *simto* operator identifies subexpressions to be matched using similarity, with other parts of the expression matched exactly. Two instances of the same wildcard in the same query required exact matches.

Formula+Text Search Queries. In NTCIR-10, there were 15 formula+text queries and 19 text queries. Among these queries, all the formula+text queries and none of the text-only queries were assessed for relevance. In NTCIR-11, more topics were assessed compared to NTCIR-10, and overall 50 queries are available for the formula+text task. All the topics contained at least one keyword and one formula. In NTCIR-12, topics were developed for search on two different collections: arXiv and Wikipedia. 29 arXiv and 30 Wikipedia topics were assessed. All the topics contained at least one formula, but 5 in the arXiv and 3 in the Wikipedia set had no keywords.

3.1.2 Pooling

Pooling is the process of selecting retrieved documents by different retrieval systems for relevance judgement. We explain how pooling was done for formula search and formula+text search in NTCIR test collections.

Formula Search. The set of formula instances to be judged was created by pooling submitted runs for three of the four NTCIR test collections. No pooling was used for NTCIR-11, a known-item retrieval task. In NTCIR-10, participating teams could submit up to 100 formula instances[2] per formula query. The highest-ranked formula instances were selected from each submitted run, one rank at a time, until the pool to be judged contained at least 100 unique formula instances, each of which had been highly ranked by at least one system. In the NTCIR-12 WFB and NTCIR-12 *simto* tasks, participating teams could submit up to 1,000 results per topic. In each case (separately), the top-20 formula instances from each submitted run were pooled. This instance-based definition of uniqueness sometimes resulted in limited diversity in the judgment pools. As the most extreme example, for the NTCIR-12 WFB query β (a short formula consisting of a single symbol), every formula instance in the pool of formulas to be judged was β.

Formula+Text Search. In NTCIR-10, the pooling process for formula and formula+text queries was the same. As with NTCIR-10, each participating team in the NTCIR-11 formula+text search task was allowed to submit up to 4 runs, returning the top-1000 results per query. For each topic, 50 retrieval units were chosen from the union of the retrieval results. In NTCIR-12, the top-20 ranked results per run were included in the pool for assessment.

3.1.3 Judging and Encoding Relevance

Formula Search. Determining the relevance of a retrieved formula to a formula query can be challenging. First, relevance naturally depends on the user's reason for issuing that query, only parts of which may be signaled by the content or form of the query. Second, even with an understanding of that context, mathematical knowledge may be needed to recognize whether a retrieved formula would likely be useful.

NTCIR-11 was a known-item retrieval task. For each topic, the single Relevant (R) formula instance was defined as the formula instance that had been used as the formula query. Note that there may

[2]Two formula instances were considered different if they occur in different documents, even if they were visually identical.

have been other instances of the same or similar formulas in the collection, but like all instances of other formulas, they would be scored as Non-relevant (N).

For NTCIR-10, the assessors were mathematicians or math students. Assessors viewed each formula instance from the judgment pool in isolation and assigned each a grade of Relevant (R), Partially relevant (P), or Non-relevant (N) to that pool's query, considering the query-specific scenario and judgment criteria that had been specified when the query was created.

For the NTCIR-12 WFB task, there were two groups of assessors; each group independently judged each pooled formula. One group consisted of computer science graduate students, the other consisted of computer science undergraduates. Assessors viewed each formula instance from the judgment pool for a query in context and assigned each a relevance grade of Relevant (R), Partially relevant (P), or Non-relevant (N) to that pool's query, informed by the scenario and judgment criteria that had been specified when the query was created. The pooled formula instances were shown to the assessors in context (highlighted in the text where they had appeared in the collection), but assessors were not asked to interpret the pooled formula in that specific context; the assessment was to be done based on the pooled formula itself, with reference to the scenario and relevance criteria that were provided to the assessor with that query. For the NTCIR-12 *simto* task, two assessors judged each pooled formula, following the same approach as for the NTCIR-12 WFB task.

Formula+Text Search. The assessment process in NTCIR-10 for text+formula searches was similar to the formula search task, with the same assessors. The relevance was decided based on the retrieved formula, not the document. Using the SEPIA system,[3] the assessors were shown the retrieved formula and its context. Using the same assessment system, in NTCIR-11 the assessors were shown the title of the topic, the relevance description, and an example hit (if any) as supplementary information. In this collection, in contrast to NTCIR-10, the relevance was assessed not on a formula basis, but a retrieval unit basis. That is to say, the assessors judged the relevance of each retrieval unit to the query based on the keywords, as well as the formulas included in the submission files. To make sure that assessors are familiar enough with math, they were chosen from third-year undergraduate and from graduate students in mathematics. For each retrieval unit, the assessors were asked to select either relevant (R), partially-relevant (P), or non-relevant (N). The relevance levels were similarly defined in NTCIR-12, however, the assessor for Wikipedia topics were computer science students. Each hit was evaluated by one undergraduate and one graduate student.

[3] https://code.google.com/archive/p/sepia/

3.1.4 Evaluation Protocols

NTCIR-10 and -12 combined 3-level judgments from two assessors to form a 5-level "Aggregate" relevance judgment for each assessed document. This was done by mapping N to 0, P to 1, and R to 2, and then summing the two scores. The resulting integer scores ranged from a low of 0 (both assessors judged N) to a high of 4 (both assessors judged R). For computing evaluation measures, this 0 to 4 levels are binarized by treating relevance grades 0, 1, and 2 as non-relevant and treating 3 and 4 as relevant. In all NTCIR formula search tasks, evaluation measures were computed on formula instances. NTCIR-10 reported P@5, P@10, P@hit (i.e., for all returned results), and MAP. P@K shows the ratio of relevant documents retrieved in the top-k results. The NTCIR-11 main task replaced P@hit, with bpref [23] a preference-based measure designed for incomplete relevance judgements. Bpref is an effectiveness measure that considers whether relevant documents are ranked above the non-relevant ones, ignoring the unassessed hits defined as:

$$bpref = \frac{1}{R} \sum_{r} \left(1 - \frac{|n\, ranked\, higher\, than\, r|}{\min(R, N)} \right)$$

where R is a set of relevant documents and N is a set of non-relevant documents. The evaluation measure for the NTCIR-11 formula search task was different because that was a known-item retrieval task. Mean Reciprocal Rank (MRR) for the one known relevant formula instance per query was the reported evaluation measure. MRR is the average of the reciprocal ranks of the first relevant document. NTCIR-12 tasks reported P@5, P@10, P@15, and P@20. In all cases, relevance judgments for formula instances that were missing from the pools (as can happen for P@hit and MAP) were treated as not relevant.

3.2 The ARQMath Test Collections

This section presents our work on the ARQMath-1 [173], ARQMath-2 [108], and ARQMath-3 [100] test collections. All three test collections had two main tasks: (1) answer retrieval for math questions, (2) contextual formula search. We consider these main differences between ARQMath and previous test collections:

1. **Queries.** ARQMath collections have more queries, which are also more diverse.

2. **Relevance.** In ARQMath, the definition of relevance is more strongly tied to the contexts in which the which formulas originally appeared.

3. **Pooling.** In NTCIR, many instances of the same formula might be retrieved, so assessment pools could be rich in duplicates. In ARQMath, identical formulas were clustered prior to pooling, leading to more diverse pools.

4. **Evaluation.** In NTCIR, systems were given credit for every retrieved formula instance that was relevant. In ARQMath, systems received credit only for the first instance of each distinct relevant formula retrieved.

While the NTCIR collections contained the arXiv and Wikipedia articles, ARQMath uses questions and answers from the Community Question Answering (CQA) website Math Stack Exchange (MathSE).[4] In terms of technical complexity, MathSE varies from simple questions to expert-level mathematical inquiries. The Internet Archive provides free public access to MathSE snapshots.[5] For ARQMath-1 we processed the 01-March-2020 snapshot, which in its original form contained the following in separate XML files:

- **Posts**: Each MathSE post has a unique identifier, and can be a question or an answer, identified by '`post type id`' of 1 and 2 respectively. Each question has a title and a body (content of the question) while answers only have a body. Each answer has a '`parent id`' that associates it with the question it is an answer for. There is also other information included for each post, such as the post owner id and creation date.

- **Comments**: MathSE users can comment on posts. Each comment has a unique identifier and a '`post id`' indicating which post the comment is written for.

- **Post links**: Moderators sometimes identify duplicate or related questions that have been previously asked. A '`post link type id`' of value 1 indicates related posts, while value 3 indicates duplicates.

- **Tags**: Questions can have one or more tags describing the subject matter of the question.

- **Votes**: While the post score shows the difference between up and down votes, there are other vote types such as '`offensive`' or '`spam.`' Each vote has a '`vote type id`' for the vote type and a '`post id`' for the associated post.

[4] https://math.stackexchange.com/

[5] `https://archive.org/download/stackexchange`

- **Users**: Registered MathSE users have a unique id, and they can provide additional information such as their website. Each user has a reputation score, which can be increased through activities such as posting a high quality answer, or posting a question that receives up votes.
- **Badges**: Registered MathSE users can also receive three badge types: bronze, silver and gold. The '`class`' attribute shows the type of the badge, value 3 indicating bronze, 2 silver and 1 gold.

Questions and answers from 2010-2018 are included in the collection to be searched.[6] The ARQMath collection contains roughly 1 million questions and 28 million formulas. Formulas in the collection are annotated using `<span>` tags with the class attribute `math-container`, and a unique integer identifier given in the `id` attribute. Formulas are also provided separately in three TSV files for different formula representations (LaTeX, Presentation MathML, and Content MathML). Each line of a TSV file represents a single instance of a formula, containing the formula id, the id of the post in which the formula instance appeared, the id of the thread in which the post is located, a post type (title, question, answer or comment), and the formula representation in either LaTeX, Symbol Layout Tree (SLT) (Presentation MathML), or Operator Tree (OPT) (Content MathML).

HTML views of question threads, similar to those on the Math Stack Exchange web site (a question, along with answers and other related information) are also included in the ARQMath collection. The threads are constructed automatically from Math Stack Exchange snapshot XML files. The threads are intended for those performing manual runs, or who wish to examine search results (on queries other than evaluation queries) for formative evaluation purposes. These threads are also used by assessors during evaluation. The HTML thread files were intended only for viewing threads; participants were asked to use the provided XML and formula index files to train their models.

All three ARQMath test collections, used a same collection. However, in ARQMath-2021,[7] we added a new field for a *visually distinct* formula identifier, which is used in evaluation for task 2 (Formula Retrieval). The idea is to identify formulas sharing the same appearance. So for example, two occurrences of x^2 in a TSV formula index will have different formula *instance* identifiers, but the same *visually distinct* formula identifier. In our last release of our collection in 2022, we fix existing issues we were notified about in the previous years. These fixes are explained in the Appendix.

In the next subsections, we review the two tasks. As the formula search task has been evaluated

[6] which we call the ARQMath collection

[7] We refer to the test collections as ARQMath-1, -2 and -3. ARQMath-2020, -2021, and -2022 refers to the labs

before, we first make comparison with the previous test collections.

3.2.1 Contextual Formula Search Task

We first review ARQMath contextual formula search task. This is the second task in ARQMath, but providing search models for this task is the primary goal of this research.

Task definition. The goal of ARQMath task 2 (contextual formula search) is, given a formula inside a math question, retrieve the top-1000 relevant formulas from the questions and answers in the collection. Table 3.1 shows an example query, B.4 in ARQMath-1 (shown in blue), with relevant (shown in green) and non-relevant (shown in red) retrieved formulas.

Here we provide details on the ARQMath-1, -2 and -3 test collections and also compare them against the previous formula search tasks in NTCIR.

Queries. In ARQMath-1, 87 mathematical formulas were provided for the participants from the questions posted on MathSE in 2019. In ARQMath-2, 100 formulas were chosen from the 2020 questions and distributed to the participants. As ARQMath-2, 100 formulas from questions in 2021 were used in ARQMath-3. Topics in task 2 had the question title, body, and question tags from Math Stack Exchanges. For all the formulas inside the questions in which the query appeared, the LaTeX, SLT, and OPT representations are in the same TSV file format as the collection. All the queries are chosen from the questions originally developed for ARQMath task 1. To select these, a set of criteria were considered.

For ARQMath-1, a total of 74 concrete formula queries were assessed. Of these 74, 45 formula queries were assessed initially and used to evaluate systems (the test queries), with the remaining 29 designated for use in training future systems (the additional queries). In ARQMath-2, 58 queries were assessed and used for the evaluation, and then 12 additional queries were assessed. Note that some of the additional queries in both ARQMath-1 and -2 are queries that were removed from the evaluation set because of the small number of relevant formulas, as score quantization for MAP′ can be quite substantial when only a single relevant formula contributes to the computation. In ARQMath-3, 76 queries were assessed and used for evaluation. At the time of writing this dissertation, the assessment of additional queries are ongoing.

ARQMath provided a manually annotated complexity label for its formula queries. Based on the number and diversity of symbols in a formula, 3 levels of complexity were defined: Low (L), Medium (M), and High (H). For this analysis, we have also manually annotated the NTCIR-10 queries and

Table 3.1: Example of ARQMath Contextual Formula Search task query and results.

FORMULA QUERY INSIDE A MATH QUESTION(TOPIC B.4) I have the sum $$\sum_{k=0}^{n} \binom{n}{k} k$$ know the result is $n^2 - 1$ but I don't know how you get there. How does one even begin to simplify a sum like this that has binomial coefficients.
RELEVANT ... which can be obtained by manipulating the second derivative of $$\sum_{k=0}^{n} \binom{n}{k} z^k$$ and let $z = p/(1-p)$ ...
NON-RELEVANT Yes, it is in fact possible to sum this. The answer is $$\sum_{k=0}^{n} \binom{n}{k}\binom{m}{k} = \binom{m+n}{n}$$ assuming that $n \leq m$. This comes from the fact that ...

the 20 concrete NTCIR-12 queries in the same way; these new annotations were checked by the same mathematician who checked the complexity labels for the ARQMath test collection (Dr. Anurag Agarwal). Table 3.2 shows the distribution of topic complexity for each test collection, along with an example of each complexity level. In NTCIR-10 and ARQMath, there are more low-complexity queries than in NTCIR-12.

As we [106] found, low-complexity formulas are common in queries posed to general-purpose search engines. Therefore, NTCIR-10 and ARQMath queries might be representative models for that use case. We also note that current systems find high-complexity formula queries challenging, so over-reliance on high-complexity formulas might decrease our ability to distinguish between systems if

Table 3.2: Concrete query complexity for NTCIR-10/12 and ARQMath.

Complexity		NTCIR-		ARQMath-1		ARQMath-2		ARQMath-3
Level	Example	10	12-WFB	test	additional	test	additional	test
Low	$O(mn \log m)$	11	4	21	15	22	4	23
Medium	$\int_0^\infty \frac{\sin x}{x^a}$	6	6	16	11	24	6	31
High	$\sum_{r=1}^{n} (-1)^{(n-r)} \binom{n}{r} (r)^m$	5	10	8	3	12	2	22

all systems do poorly on that subset of the queries. On the other hand, high-complexity queries can be easily produced by searchers using cut-and-paste, so it is useful for test collections to contain some high-complexity queries to characterize systems on queries of that type.

Pooling. One of the issues with previous test collections was that, for some queries, there were low numbers of unique assessed formulas. To have more diverse assessed formulas, ARQMath pooling is based on visually-distinct formulas. Specifically, when two formulas have identical symbolic representations, they are defined as *visually identical*, and otherwise as *visually distinct*. Symbolic identity was defined as identical Symbol Layout Trees, as represented by Tangent-S [35], when both were parseable, or identical LaTeX strings otherwise. Table 3.3 illustrates this distinction by providing examples of visually identical formulas with different LaTeX representations.

In ARQMath 2020, the participants were asked to retrieve the top-1000 relevant formula instances. In ARQMath-1 the pooling process, visually distinct formulas were used by first clustering all formula instances from all submitted runs to identify visually distinct formulas, and then proceeding down each list until at least one instance of some number of different formulas had been seen. For primary runs, and for the baseline run, the pool depth was the rank of the first instance of the 25th visually distinct formula; for alternate runs, the pool depth was the rank of the first instance of the

Table 3.3: Visually-identical formulas with different LaTeX representations.

Formula	Different LaTeX Strings		
$a^2 = 2b^2$	`{a^{2}=2b^{2}}`	`{a^2}=2{b^2}`	`a^2=2b^{2}`
$m \neq 0$	`{m\ne0}`	`m\not=0`	`m\ne\\0`
$\frac{n}{m}$	`{\dfrac{n}{m}}`	`n\overm`	`\fracnm`

Table 3.4: Formula retrieval test collections: sources, number of participating teams, and number of runs (including baseline systems).

Task	Collection	#Teams	#Runs
ARQMath-3	Math Stack Exchange	5	20
ARQMath-2	Math Stack Exchange	6	18
ARQMath-1	Math Stack Exchange	3	11
NTCIR-12 WFB	Wikipedia	2	7
NTCIR-12 *simto*	arXiv	2	8
NTCIR-11	Wikipedia	7	21
NTCIR-10	arXiv	6	12

10th visually distinct formula. Assessment was done on formula instances, so for each formula we selected at most five instances to assess. The 5 instances that were contributed to the pools by the largest number of runs were selected, and ties were broken randomly. Out of 5,843 visually distinct formulas that were assessed, only 93 (1.6%) had instances in more than 5 pooled posts.

In ARQMath 2021, the participants had the visual-id for each formula in advance, with clustering performed over the full collection. Pooling for ARQMath-2 task 2 queries was performed by then proceeding down each result list until at least one instance of some number of visually distinct formulas had been seen. For primary runs, and for the baseline run, the pool depth was the rank of the first instance of the 20th visually distinct formula; for alternate runs, the pool depth was the rank of the first instance of the 10th visually distinct formula. For each visually distinct formula, at most five instances were selected for assessment. In order to prefer highly-ranked instances and instances returned in multiple runs, the 5 instances were chosen using a voting protocol, where each instance votes by the sum of its reciprocal ranks within each run, breaking ties randomly. Out of 8,129 visually distinct formulas that were assessed, 117 (1.4%) had instances in more than 5 pooled posts.

In ARQMath 2021, the participants also had the visual-ids in advanced. Pooling for ARQMath-3 task2 queries was done with depths of 25 and 15 visually distinct formulas for primary and alternate runs. For each query, on average there were 154.35 visually distinct formulas to be assessed, and only 6% of visually distinct formulas had more than 5 instances.

Table 3.4 shows the number of runs that were pooled in each of the seven formula retrieval test collections that used pooling. No pooling was used for NTCIR-11, a known-item retrieval task.

Table 3.5: Relevance scores, ratings, and definitions for ARQMath task 2.

Score	Rating	Definition
3	High	Just as good as finding an exact match to the formula query would be
2	Medium	Useful but not as good as the original formula would be
1	Low	There is some chance of finding something useful
0	Not Relevant	Not expected to be useful

Judging and Encoding Relevance. ARQMath-1 assessment was done with 3 undergraduate mathematics students. There were two rounds of assessor training. ARQMath-2 and -3 had 3 assessors (undergraduate and graduate computer science and mathematics students), but with three rounds of training. In all test collections, after the assessment, each assessor was given two topics assessed by the other two assessors to allow us to calculate agreement. The average Cohen's kappa coefficient for ARQMath-1 was 0.30 on the four-way assessments with High+Medium binarization[8] the average kappa was 0.48. In ARQMath-2, these values increased to 0.33 and 0.69, respectively. In ARQMath-3, a kappa of 0.44 was achieved which was higher than the previous test collections, however, with High+Medium binarization, kappa was 0.51; lower than ARQMath-2 and higher than ARQMath-1. The average assessment time in ARQMath-1 was 38.1 seconds per formulas, 39.5 in ARQMath-2, and 26.6 in ARQMath-3.

In ARQMath-1 the relevance judgment task was defined for assessors as follows:

For a formula query, if a search engine retrieved one or more instances of this retrieved formula, would that have been expected to be useful for the task that the searcher was attempting to accomplish?

Assessors were presented with formula instances, and asked to decide their relevance by considering whether retrieving either that instance or some other instance of that formula would have been useful, assigning each formula instance in the judgment pool one of four scores as defined in Table 3.5.

For example, if the formula query was $\sum \frac{1}{n^{2+\cos n}}$, and the formula instance to be judged is $\sum_{n-1}^{\infty} \frac{1}{n^2}$, the assessors would decide whether finding the second formula rather than the first would be expected to yield good results. To do this, they would consider the content of the question post containing the query (and, optionally, the thread containing that question post) to understand the

[8]Considering hits with relevance score of 2 and 3 as relevant, and 0 and 1 as not relevant

searcher's actual information need. Thus, the question post fills a role akin to Borlund's simulated work task [20], although in this case the title, body, and tags from the question post are included in the topic and thus can optionally be used by the retrieval system. The MathSE question post was used in place of the scenarios that had been provided with NTCIR-10 and NTCIR-12 queries; no query-specific relevance judgment criteria were provided. The assessor can also consult the post containing a retrieved formula instance (which may be another question post or an answer post), along with the associated thread, to see if in that case the formula instance would indeed have been a useful basis for a search. Note, however, that the assessment task is not to determine whether the specific post containing the retrieved formula instance is useful, but rather to use that context as a basis for estimating the degree to which useful content would likely be found if this or other instances of the retrieved formula were returned by a search engine.

We then defined the relevance score for a formula to be the maximum relevance score for any judged instance of that formula. This relevance definition essentially asks "if instances of this formula were returned, would we reasonably expect some of those instances to be useful?" Figure 3.1 shows the Turkle[9] interface used for assessment in ARQMath. As shown in the left panel of the figure, the formula query $\sum_{k=0}^{n} \binom{n}{k} k$ was highlighted in yellow. The assessors could use the question context to understand the user's information need. In the right panel, two instances of one visually-distinct formula, $\sum_{k=0}^{n} \binom{n+k}{k}$, are shown in two different posts. For each instance, the assessor could consider the post in which the instance appeared when deciding the relevance degree.

Relevance Assessment in ARQMath-1 vs ARQMath-2 and -3. Although this definition of relevance was unchanged between ARQMath test collections, we did make one potentially significant change to the way this relevance definition was interpreted for ARQMath-2 and -3. It should be noted that one of the ARQMath lab organizers who is a mathematician (Dr. Anurag Agarwal) reviewed these two examples and agreed with the assessors. In 2020, ARQMath-1 assessors had been instructed during training that if the query and candidate formulas were the same (in appearance), then the candidate was certainly highly relevant. During assessor training in 2021, this issue received considerable attention and discussion, and we ultimately concluded that our guidance in 2020 had not been fully consistent with our relevance definition. We, therefore, clarified the interpretation of 'exact match' for ARQMath-2 assessment in 2021 and for ARQMath-3 assessment in 2022 to take the formula semantics and context directly into account, even in the case of identical formulas (so for example, variables of different types would not be considered the same, even if variable names are identical). This means that an exact match with the formula query may in some cases (depending on context) be considered not relevant. This change may affect the utility of some

[9] `https://github.com/hltcoe/turkle`

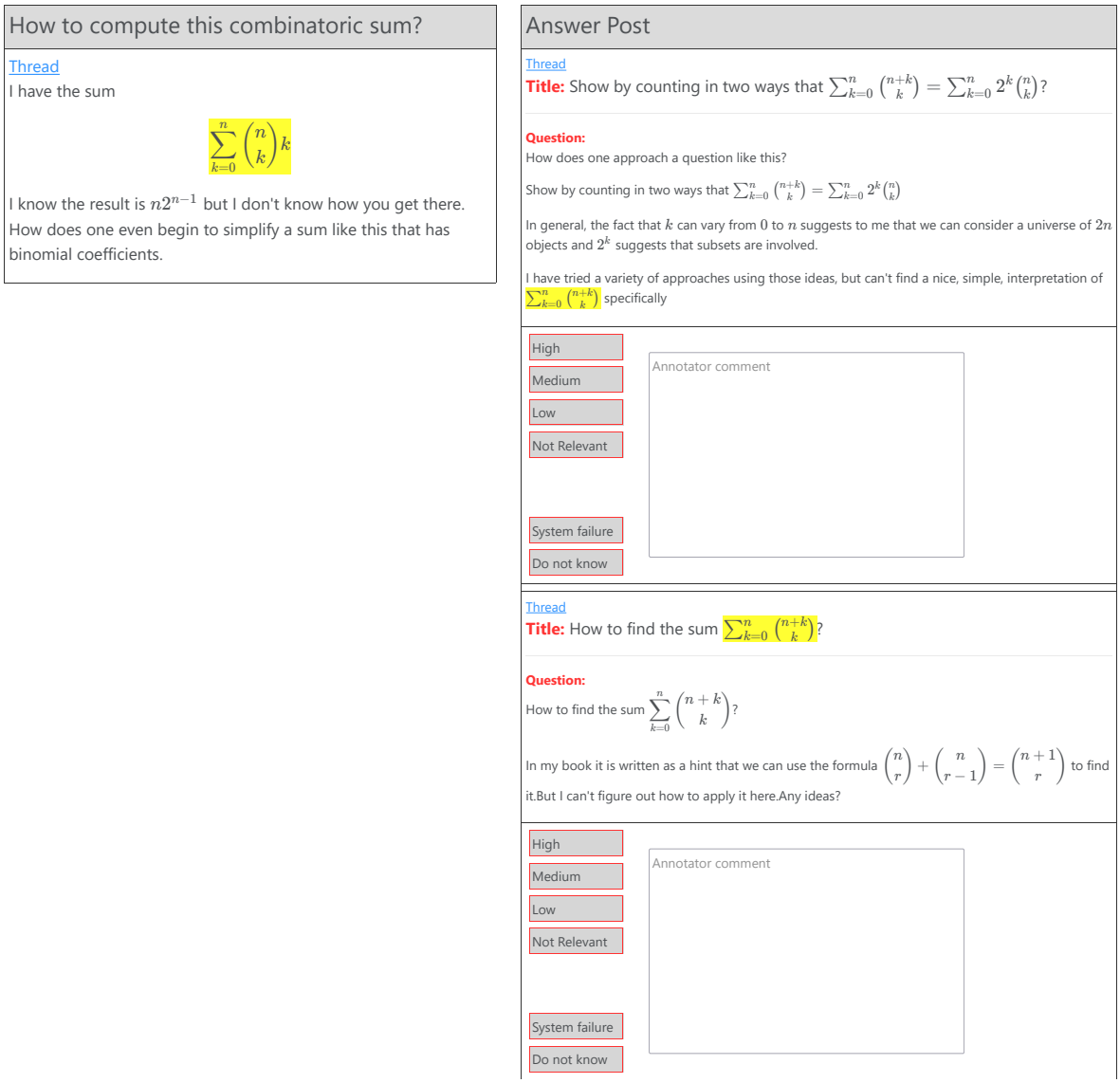


Figure 3.1: Turkle assessment interface for ARQMath Formula Search task. In the left panel, the formula query is highlighted. In the right panel, two posts (both of which are questions) containing the same retrieved formula are shown. Assessors considered context by looking at the question the query appears in, and the posts in which retrieved formulas appear.

ARQMath-1 relevance judgments for training systems that will be evaluated using ARQMath-2 or -3 relevance judgments.

For example, for the formula query $x^n + y^n + z^n$ (B.289 from ARQMath-2) x, y, and z could be any real numbers in the original question post in which that formula had originally appeared. The assessors considered all exact matches in the pooled posts in which x, y, and z referred not to real numbers but specifically to integers as not relevant. On the other hand, formulas that do not share the same appearance or syntax as the query might be considered relevant. This is usually the case where they are both referring to the same concept. For the formula query $\frac{S}{n} \geq \sqrt[n]{P}$ (ARQMath query B.277), formula $\frac{1+2+3+...+n}{n} \geq \sqrt[n]{n!}$ has medium relevance. Both formulas are referring to the AM-GM inequality (of Arithmetic and Geometric Means).

As in ARQMath-1, for ARQMath-2 and ARQMath-3 we defined the relevance score for a formula to be the maximum relevance score for any judged instance of that formula.

Evaluation Measures. One risk when performing a new task for which rich training data is not yet available is that a larger than a typical number of relevant hits may be missed. Measures that treat unjudged documents as not relevant can be used when directly comparing systems that contributed to the judgment pools, but subsequent use of such a new test collection can be disadvantaged by treating unjudged documents (which, as systems improve, might be relevant) as not relevant. We, therefore, chose the nDCG′ measure (read as "nDCG-prime") introduced by Sakai [139] as the primary measure for ARQMath Tasks 1 and 2. NDCG (Normalized Discounted Cumulative Gain) is the ratio of DCG@k to the ideal ranking at rank k. DCG is the total gain accumulated at rank k defined as:

$$DCG_k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_2(i+1)}.$$

The nDCG measure on which nDCG′ is based is widely used when graded relevance judgments are available, as we have in ARQMath, that produces a single figure of merit over a set of ranked lists. We compute Mean Average Precision (MAP′), and Precision at 10 posts (P′@10), after removing the unjudged hits. For MAP′ and P′@10 we used High+Medium binarization.

Participants submitted the lists of formula instances, but we computed these measures over visually distinct formulas. Task 2 evaluation script replaces each formula instance with its associated visually distinct formula, and then deduplicates from the top of the list downward, producing a ranked list of visually distinct formulas, from which our "prime" evaluation measures are then computed using

Table 3.6: Example of an ARQMath-1 answer retrieval task topic (A.4) with a relevant and a non-relevant answer.

Question
I have spent the better part of this day trying to show from first principles that this sequence tends to 1. Could anyone give me an idea of how I can approach this problem? $$\lim_{n\to+\infty} n^{\frac{1}{n}}$$
Relevant
You can use AM $\geq$ GM. $$\frac{1+1+\cdots+1+\sqrt{n}+\sqrt{n}}{n} \geq n^{1/n} \geq 1$$ $$1-\frac{2}{n}+\frac{2}{\sqrt{n}} \geq n^{1/n} \geq 1$$
Not Relevant
If you just want to show it converges, then the partial sums are increasing but the whole series is bounded above by $$1+\int_1^{\infty} \frac{1}{x^2}dx = 2$$

trec_eval.[10]

3.2.2 Answer Retrieval Task

Task definition. The goal of this task is to find relevant answers for mathematical questions. The participants were asked to retrieve the top-1000 relevant answers in the collection. Note that in contrast to the ARQMath formula search task, the retrieved documents are only the answers, not the questions. Table 3.6 shows an example topic (topic A.4 in ARQMath-1) with a relevant and a non-relevant retrieved answer.

Topics. In ARQMath-1, 101 questions from posts in 2019 were distributed to the participants. Similarly, ARQMath-2 provided 100 questions from 2020 posts. Finally, in ARQMath-3 100 questions from 2021 posts were selected.[11] The main constraint for selecting topics was that the question post should contain at least one formula.

[10] `https://github.com/usnistgov/trec_eval`

[11] In both ARQMath-1 to -3, some questions were that assessed had 0/1 relevant answers in the judgement pools.

As ARQMath-1 was the first test collection, and because ranking quality measures can distinguish between systems only on topics for which relevant documents exist, we calculated the number of known duplicate and related posts for each question and chose only from those that had at least one duplicate or related post.[12] We removed this constraint in the ARQMath-2 topic selection process. In ARQMath-2 we had included 11 topics for which there were no known duplicates on an experimental basis. Of those 11, 9 had turned out to have no relevant answers found by any participating system or baseline.

In ARQMath-3, We selected 139 candidate topics from among the 3313 questions that satisfied both of our strict criteria: (1) having a minimum of one duplicate questions in the ARQMath collection, (2) having at least one formula in the title or body. We also applied additional soft criteria based on the number of terms and formulas in the title and body of the question, the question score that Math Stack Exchange users had assigned to the question, and the number of answers, comments, and views for the question. From those 139, we manually selected 100 topics in a way that balanced three desiderata: (1) A similar topic should not already be present in the ARQMath-1 or ARQMath-2 test collections, (2) we expected that our assessors would have (or be able to easily acquire) the expertise to judge relevance to the topic, and (3) the set of topics maximized diversity across four dimensions (question type, difficulty, dependence, and complexity).

The topics were selected from various domains (real analysis, calculus, linear algebra, discrete mathematics, set theory, number theory, etc.) that represent a broad spectrum of areas in mathematics that might be of interest to expert or non-expert users. The difficulty level of the topics spanned from easy problems that a beginning undergraduate student might be interested in though difficult problems that would be of interest to more advanced users. The bulk of the topics were aimed at the level of undergraduate math majors (in their 3rd or 4th year) or engineering majors fulfilling their math requirements.

Overall, 226 assessed topics are produced for testing in ARQMath with 77 in ARQMath-1, 71 in ARQMath-2, and 78 in ARQMath-3. When choosing topics, 3 kinds of categories were considered. The first category was based on their type (topic). As organizers, we labeled each question with one of three broad categories, *computation*, *concept* or *proof*. Out of the 77 assessed topics in ARQMath-1, 26 were categorized as *computation*, 10 as *concept*, and 41 as *proof*. In ARQMath-2, out of the 71 assessed topics, 25 were categorized as *computation*, 19 as *concept*, and 27 as *proof*. In ARQMath-3, out of 78 assessed topics, 21 were categorized as *computation*, 16 as *concept*, and 41 as *proof*. We also categorized the topics based on their perceived difficulty level. ARQMath-1

[12]Note that participating systems did not have access to this information.

has 32 topics categorized as easy, 21 as medium, and 24 as hard. With a lower number of hard questions, ARQMath-2 has 32 categorized as easy, 20 as medium, and 19 as hard. With a higher number of medium questions, ARQMath-3 has 17 easy, 43 medium, and 18 hard topics. Our last categorization was based on whether a topic is dependent on text, formulas, or both. In ARQMath-1, 10 topics were dependent on text, 21 on formulas, and 40 on both. In ARQMath-2, 13 topics were dependent on text, 32 on formula, and 32 on both. Finally, in ARQMath-3, 10 topics were dependent on text, 22 on formula, and 46 on both. Figure 3.2 shows counts of topics per category in ARQMath-1, -2 and -3.

Runs and Pooling. In all ARQMath test collections, we considered 4 systems as baselines. In ARQMath-1, there was a fifth baseline, Approach0 [141], a formula search engine. In ARQMath-3, we included a fifth baseline system using PyTerrier [95] for the TF-IDF model, which uses PyTerrier with symbols in LaTeX strings first mapped to English words to avoid tokenization problems. Here is a description of our baseline runs.

1. **TF-IDF.** A term frequency inverse document frequency) model using the Terrier system [124], with formulas represented using their LaTeX strings. Default parameters from Terrier were used. One problem with this baseline is that Terrier removes some LaTeX symbols during tokenization. The second uses PyTerrier [95], with symbols in LaTeX strings first mapped to English words to avoid tokenization problems.

2. **Tangent-S**. A formula search engine using SLT and OPT formula representations [35]. One formula was selected from each task 1 question title if possible; if there was no formula in the title, then one formula was instead chosen from the question's body. If there were multiple formulas in the selected field, the formula with the largest number of symbols (nodes) in its SLT representation was chosen; if more than one formula had the largest number of symbols, we chose randomly between them. Note that this system only considers formulas and not text.

3. **TF-IDF + Tangent-S.** A combination of scores from the TF-IDF and Tangent-S baselines. The relevance scores from both systems were normalized in [0,1] using min-max normalization, and then combined using an unweighted average.

4. **Linked Math Stack Exchange Posts.** This is a simple oracle "system" that is able to see duplicate post links from the original question post in the Math Stack Exchange collection (which were not available to participants). It returns *all* answer posts from 2018 or earlier that were in threads that Math Stack Exchange moderators had marked as duplicating the

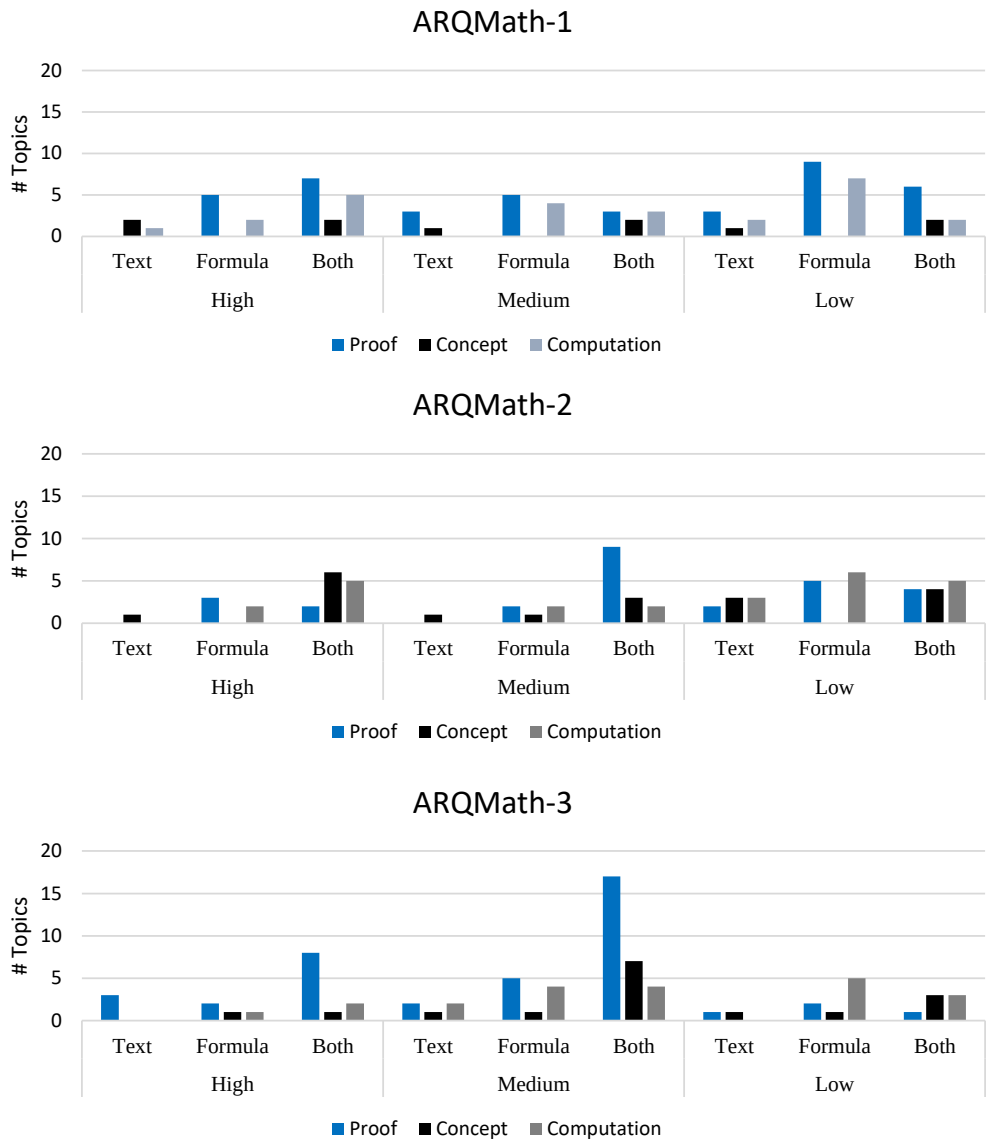


Figure 3.2: Histogram of Topic Counts over 3 Categories in ARQMath-1, -2 and -3 Task 1.

topic question post. Answer posts are sorted in descending order by their vote scores. The vote scores is the difference between up and down votes given by Math Stack Exchange users to an answer.

5. **Approach0.** A path-based formula search engine [177] using OPT formula representations (See Chapter 4, Section 4.2.2.). This system uses both text and formula for search. The inputs queries are manually generated from the questions.

The participating teams in ARQMath were allowed to submit up to 5 runs, with one being the primary run and the others being alternate runs. In ARQMath 2020[13] there were 5 participating teams, with 18 runs. One of the runs was reported as manual. Manual runs are those that had manual processing at any stage of the search. In ARQMath 2021, 9 teams participated in task 1, and there were 36 runs, double that of ARQMath 2020. 8 runs were reported as manual by the participating teams. In ARQMath 2022, a total of 33 runs were received from 7 teams. Of these, 28 runs were declared to be automatic, with no human intervention at any stage of generating the ranked list for each query. The remaining 5 runs were declared to be manual.

Participants were asked to rank 1,000 (or fewer) answer posts for each task 1 topic. Top-k pooling was then performed to create pools of answer posts to be judged for relevance to each topic. As in each year, the number of runs was different in ARQMath labs, the pooling depth was different for each test collection. In ARQMath-1, the top 50 results were combined from all 7 primary runs, 4 baselines, and 1 manual run. To this, we added the top 20 results from each of the 10 automatic alternate runs. In ARQMath-2, the top 45 results were combined from all primary runs. To this, we added the top 15 results from each alternate run. Two baseline systems, TF-IDF + Tangent-S and Linked Math Stack Exchange Posts, were pooled as primary runs and the other two (TF-IDF and Tangent-S) were pooled as alternate. In ARQMath-3, the top 45 answer posts were included in pooling for primary runs; for alternate runs, the top 20 were included. Two baseline runs, PyTerrier TF-IDF+Tangent-S and Linked Math Stack Exchange Posts, were pooled as primary runs (i.e, to depth 45); other baselines were pooled as alternate runs (i.e., to depth 20). These pooling depths were chosen based on assessment capacity, with the goal of identifying as many relevant answer posts as possible.

Judging and Relevance. The ARQMath-1 task 1 assessment was done by eight undergraduate mathematics students. Three of the assessors later moved to the formula search task. There were four rounds of assessor training. ARQMath-2 had five undergraduate mathematics student

[13] ARQMath 2020 refers to ARQMath lab in year 2020

assessors. Three assessors (also undergraduate mathematics students) who initially worked on the formula search task later joined the five task 1 assessors and then finally moved back to assess the additional task 2 queries. There were four rounds of task-1 training for the main assessors and two rounds of training for formula search task assessors. The ARQMath-3 assessment was done by five assessors (undergraduate and graduate computer science and mathematics students), with three rounds of training. Same as ARQMath-2, three assessors (undergraduate and graduate computer science and mathematics students) who initially worked on the formula search task later joined task 1 assessor.

In ARQMath-1, after the assessment, each assessor was given two topics assessed by the other two assessors to calculate agreement. Due to time constraints, in ARQMath-2 and -3, each assessor assessed only one topic assessed by some other assessor. The average Cohen's kappa coefficient for ARQMath-1 was 0.29 on the four-way assessments, with High+Medium binarization the average kappa was 0.39. In ARQMath-2, these values were 0.21 and 0.32, respectively. In ARQMath-3, these values were 0.24 and 0.25, respectively. The average assessment time in ARQMath-1 was 63.1 seconds per answer post, compared to 83.3 in ARQMath-2, and 44.1 in ARQMath-3.

Some questions might offer clues as to the level of mathematical knowledge on the part of the person posing the question; others might not. To avoid the need for the assessor to guess about the level of mathematical knowledge available to the person interpreting the answer, we asked assessors to base their judgments on the degree of usefulness for an expert (modeled in this case as a math professor) who might then try to use that answer to help the person who had asked the original question. We defined four levels of relevance, as shown in Table 3.7.

Assessors were allowed to consult external sources to familiarize themselves with the topic of a question, but the relevance judgments for each answer post were performed using only information available within the collection. For example, if an answer contained an MathSE link such as `https://math.stackexchange.com/questions/163309/pythagorean-theorem`, they could follow that link to better understand the intent of the person writing the answer, but an external link to the Wikipedia page `https://en.wikipedia.org/wiki/Pythagorean_theorem` would not be followed.

Evaluation Measures. For the answer retrieval task, the same evaluation measures as the formula search task were used: nDCG′, MAP′ and P′@10. The only difference is that there is no deduplication used for this task. We removed unjudged posts as a preprocessing step where required, and then computed the evaluation measures using trec_eval.[14]

[14] `https://github.com/usnistgov/trec_eval`

Table 3.7: Relevance assessment criteria for the ARQMath Answer Retrieval task.

Score	Rating	Definition
3	High	Sufficient to answer the complete question on its own
2	Medium	Provides some path towards the solution. This path might come from clarifying the question, or identifying steps towards a solution
1	Low	Provides information that could be useful for finding or interpreting an answer, or interpreting the question
0	Not Relevant	Provides no information pertinent to the question or its answers. A post that restates the question without providing any new information is considered non-relevant

3.2.3 ARQMath Reusablity

ARQMath-1 and -2 had two main tasks: answer retrieval for math questions and formula search. In ARQMath-3, we kept these two as the main tasks. However, a new pilot task developed for ARQMath-3 (Task 3) Open Domain Question Answering was introduced using the ARQMath collection and the same topics introduced for the answer retrieval task (Task 1). Unlike the answer retrieval task, system answers are not limited to content from any specific source. Rather, answers can be *extracted* from anywhere, automatically *generated*, or even written by a person.

For example, suppose that we ask a Task 3 system the question "What does it mean for a matrix to be Hermitian?" An extractive system might first retrieve an article about Hermitian matrices from Wikipedia and then extract the following excerpt as the answer: "In mathematics, a Hermitian matrix (or self-adjoint matrix) is a complex square matrix that is equal to its own conjugate transpose." By contrast, a generative system such as GPT-3 can directly construct an answer such as: "A matrix is Hermitian if it is equal to its transpose conjugate." For a survey of open-domain question answering, see Zhu et al. [180].

3.3 Summary

This chapter reviewed the existing collections for math-aware search. MREC was initially only a collection, with no shared relevance judgments. CUMTC then used this collection and devel-

oped 160 assessed topics using Math Overflow questions. NTCIR-10, -11, and -12 were the three evaluation venues that had built test collections prior to our research. NTCIR-10 was the pilot math-aware search shared task, providing topics with text, a formula, or both. The goal was to search a subset of the arXiv collection. NTCIR-11 and -12 added Wikipedia to the collection, as arXiv documents were more technical. The NTCIR-11 main task was similar to NTCIR-10, but a new task for formula-only search was added. This was a known item-retrieval task, and the participants were asked to find the single intended relevant formula in the collection. There were three tasks in NTCIR-12. The main task was similar to previous evaluations, but topics for both Wikipedia and arXiv collections were separated. The formula browsing task only used Wikipedia as the collection. The third task was called "Similar-To", with eight topics from the arXiv collection. The goal was to find similar but not identical matches for the given formula query.

In our research, we have introduced three new test collections: ARQMath-1, -2, and -3. There were two main tasks defined: answer retrieval and contextual formula search. ARQMath-3 introduced a new pilot task, open domain questions answering for math questions. In the first task, the goal is to find relevant answers to math questions, whereas in the second task, the goal is to find relevant formulas for a given formula query. In all three test collections, the Math Stack Exchange posts from 2010 to 2018 are used as the collection. To select the topics, in ARQMath-1, questions posted on Math Stack Exchange from 2019, in ARQMath-2 from 2020, and in ARQMath-3 from 2021 were used. Overall, we provided 241 topics for answer retrieval, 220 for formula retrieval, and 78 topics for open-domain question answering. We can summarize the main differences between ARQMath collections and those from NTCIR as follows:

- **Tasks.** In the ARQMath test collections, the answer retrieval for math questions task was introduced with using exact questions as the input, which is different from CUMTC topics. The benefit of this task is that the information needs are expressed by real users and the topic is not lab-generated.
- **Collection.** Compared to the previous test collections that used arXiv and Wikipedia, we considered Math Stack Exchange, which contains more posts and brings more diversity in terms of complexity and domain.
- **Topics.** The number of topics in ARQMath is much larger than in the previous test collections.
- **Relevance encoding.** In previous test collections, for the formula search task, relevance was mostly decided based on the visual similarity of formulas. In ARQMath-1 we began to

consider the context in which formulas appear. We believe in ARQMath-2 and -3, we have an even better formulation of the relevant assessment process.

Considering tasks in ARQMath, in all three test collections, assessors had more agreement on formula search task compared to answer retrieval. Also, assessment for formula search was nearly twice faster than answer retrieval task.

- **Evaluation.** We consider nDCG′ as the primary measure of effectiveness. We also used mAP′ and P′@10 as our secondary measures, using binarized relevance scores. We believe our measures can provide a fair comparison among the runs participating in the pooling and later post-runs. We also introduce visually-distinct formula clustering for the formula search task, where formulas are clustered based on their visual similarity. Our goal was to assess more unique formulas compared to the previous test collections, and to ensure that systems that retrieve multiple instances of the same formula would not get extra credit.

Compared to what was existing before our research, we now have a bigger test collections available for math-aware search evaluation. There are more number of topics that systems can use for both training and evaluation. With ARQMath test collection available, we move to introducing our math-aware search systems in the next two chapters.

Chapter 4

Formula Search

This chapter introduces the isolated and contextual formula search task. In this task, the goal is to find relevant formulas for the given formula query. Based on the task that users have in mind, relevance can be defined differently. For example, a user can search for a formula to understand its application other than the domain of interest. To find relevant formulas, only isolated formulas should be considered in this case, and the context is not important. However, in another scenario, a user might be interested only in similar formulas that are related to the information need. The relevant formula in this case is associated with the context that can help answer the information need. For example, a user can search for a formula that has constraints on its variables (e.g., $x > 0$) and wants to find formulas with exact constraints. We regard to the first scenario as an isolated formula search task and the second as a contextual formula search where the context of formulas is also considered for retrieval.

To discuss approaches for formula search, we need to first discuss the representations used for mathematical notations (formulas) in documents. We would review the early encodings used for formulas and then review Presentation and Content MathML, which are commonly used in formula search systems. Then, we review the previous approaches to formula search. These approaches can be categorized into three groups: text-based, tree-based, and embedding-based. We also explore existing models that use context for formula search. Finally, we explain our approaches for isolated and contextual formula search.

4.1 Formula Representations

Math notations are complex. They range from a single symbol that can be a character or a number to a more complex structure that can include several mathematical operators. As stated by J.R. Pierce [129], mathematics and its notations are not the same thing. Notations are used to represent math. Therefore, it is essential to have notations that are readable and easy to understand. Another challenge in mathematical notations is that they keep evolving. Books on the history of mathematical notations such as the one by Florian Carjori [26] that were comprehensive at their time now seem to be outdated [15].

The problem of encoding mathematics for computer usage is older than the web. To use math in scientific documents, several document type definitions (DTD) were in use by 1992. The existing DTDs were reflecting the presentation (visual) structure, and none were considering the syntax of a formula [130]. While the commonly used notation was a version of TeX coding [79] in LaTeX, there were other DTDs such as EuroMath [158] and AAP [123].

With the growth of the web and its role as an effective means of communication, there was a lack of support for math. With no appropriate HTML tags for mathematical notations, images saved in Graphics Interchange Format (GIF) were commonly used in technical documents. In April 1998, the World Wide Web Consortium (W3C) released Mathematical Markup Language (MathML) [58], a mathematical markup language for mathematical notations. The second and third versions were released in 2003 [8] and 2010 [58]. The design goals of MathML were [58]:

- encoding math for all levels of teaching and scientific communication.
- encoding both mathematical notation and meaning (syntax).
- facilitating the conversion from/to other formats.
- allowing passing of information intended for specific renderers and applications.
- supporting efficient browsing for lengthy expressions.
- providing for extensibility.
- being well suited to template and other math editing techniques.
- being human legible, and simple for software to generate and process.

As seen in the goals, MathML should encode both the appearance and syntax of a formula. Therefore, MathML was split into Presentation MathML and Content MathML. The Presentation markup encodes the appearance of a formula, consisting of 28 elements that accept over 50 attributes.[1] Most of the elements correspond to layout schemata. There are several classes of schemata, such as the one concerned with scripts containing elements such as *msub*, *munder*, and *mmultiscripts*. The Content markup consists of 75 elements accepting roughly a dozen attributes. This markup captures the syntax of the formula.[2] The majority of these elements are corresponding to a wide variety of operators, relations, and named functions. Considering formula $x^2+4x+4=0$, Figure 4.1 shows its Presentation and Content MathML. The 'invisibletimes' MathML character entity informs rendered that there is a special spacing rule between 4 and x. Table 4.1 shows the definition of each of Presentation and Content MathML tag used to represent the formula $x^2 + 4x + 4 = 0$.

OpenMath [24] is another format that is similar to Content MathML, maintaining content dictionaries (CDs) in a standard format. In this notation, mathematical entities are represented as objects. Content dictionaries are used to assign informal and formal semantics to all symbols used in the OpenMath objects. They define the symbols used to represent concepts arising in a particular area of mathematics. This format is less used compared to MathML used by the researchers in the field, perhaps due to re-usability issues [42].

Regarding the tools that are used to convert LaTeX string to MathML representations, the most commonly used for this conversion is LaTeXML[3], in the *Perl* language. While this tool was known to be slow, there were attempts to increase efficiency and scalability in [50]. Another conversion tool is SnuggleTex[4] developed in *Java* using rule-based methods for disambiguation and translation, but this tool fails for complex formulas [57].

MathML representations can be shown in tree structures. Presentation MathML can be shown in a symbol layout tree (SLT) and Content MathML can be shown as an operator tree (OPT) [171]. Nodes in SLTs and OPTs represent symbols and explicit aggregates in the form: (**Type**!value). Nodes can be identifiers such as variable names (**V!**v), numbers (**N!**n), text fragments, such as 'lim' (**T!**t), fractions (**F!**), radicals (**R!**), explicitly specified white-space (**W!**) and finally, matrices, tabular structures, and parenthesized expressions (**M!**frxc); with **f** showing fence characters such as parentheses, r the number of rows, and c the number of columns. In the SLTs, operators do

[1] `https://www.w3.org/TR/WD-math/chapter2.html#sec2.1.3`

[2] Many of the previous works referred to this as semantics or content of a formula

[3] `https://dlmf.nist.gov/LaTeXML/`

[4] `https://www2.ph.ed.ac.uk/snuggletex/documentation/overview-and-features.html`

Presentation MathML

```
<mrow>
  <mrow>
    <msup>
      <mi>x</mi>
      <mn>2</mn>
    </msup>
    <mo>+</mo>
    <mrow>
      <mn>4</mn>
      <mo>&invisibletimes;</mo>
      <mi>x</mi>
    </mrow>
    <mo>+</mo>
    <mn>4</mn>
  </mrow>
  <mo>=</mo>
  <mn>0</mn>
</mrow>
```

Content MathML

```
<apply>
  <eq/>
  <apply>
    <plus/>
    <apply>
      <power/>
      <ci>x</ci>
      <cn>2</cn>
    </apply>
    <apply>
      <times/>
      <cn>4</cn>
      <ci>x</ci>
    </apply>
    <cn>4</cn>
  </apply>
  <cn>0</cn>
</apply>
```

Figure 4.1: Presentation and Content MathML for formula $x^2 + 4x + 4 = 0$.

not have a type attribute, but for OPTs commutative and non-commutative operators have type (**U!**) and (**O!**), respectively. For commutative operators, such as equal that the order of operands are not important the edge label are ignored (same edge labels) but for non-commutative operators the edge labels shows the order of operands.

Table 4.1: Presentation and Content MathML tags used to represent $x^2 + 4x + 4 = 0$.

Presentation MathML Tag	Definition	Content MathML Tag	Definition
mrow	Grouped sub-expressions	apply	Apply an operation to an expression
msup	Superscript	eq	Equals
mi	Identifier	plus	Plus
mn	Number	power	Power
mo	Operator	ci	Identifier
		cn	Number
		times	Multiplication

In SLTs, edge labels give the spatial arrangement of symbols on writing lines. There are nine types of edge labels, representing the spatial relationships between symbols (nodes):

1. **next** ('n') indicates that a symbol appears to the right on the same writing line.
2. **within** ('w') references the argument in a radical or the first element appearing in row-major order in a matrix/grid of type M!.
3. **element** ('e') references the next element appearing in row-major order inside a structure represented by M!.
4. **above** ('a') indicates a new writing line in the superscript position.
5. **below** ('b') references a new writing line in the subscript position.
6. **pre-above** ('SUP') indicates a prescripted superscript.
7. **pre-below** ('SUB') indicates a prescripted subscript.
8. **Under** ('U') indicates a writing line appearing below a node (e.g., a fraction's denominator).
9. **Over** indicates a writing line above a node (e.g., a fraction's numerator).

For instance, edge label a in Figure 4.4 shows '2' is superscripted *above* 'x,' and edge label n shows that '+' is located *next* to 'x'. Edge labels in OPTs indicate argument position. For commutative operators, edges have the same label. For non-commutative operators, edge labels are indexed from 0.

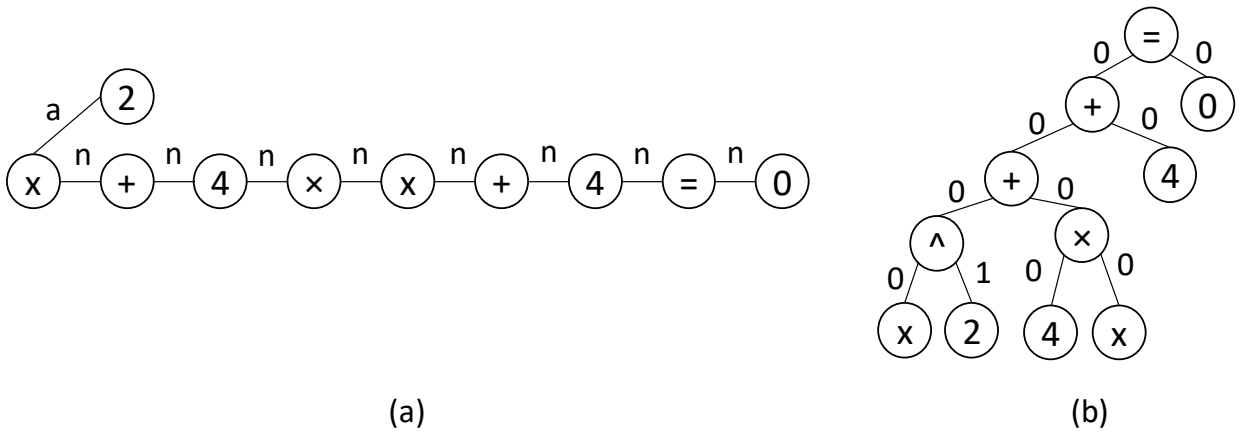


Figure 4.2: $x^2+4x+4=0$ represented as a (a) Symbol Layout Tree (SLT) and an (b) Operator Tree (OPT). SLT represents appearance by the placement of symbols on writing lines, with edge labels indicating spatial relationships (e.g., superscript (a), next on writing line (n)). OPT represents operations, with edge labels ordering arguments: arguments for commutative operators all have edges labeled '0'.

4.2 Related Work

In this section, the previous approaches for formula search are reviewed. The previous work can be categorized into these categories: Text-based, Tree-based, and Embedding models. Considering two simple formulas, $a + b$ and $b - a$, Figure 4.3 shows how these two formulas are compared in each category. In text-based models, mostly the LaTeX or linearized MathML or OpenMath representations are compared. The example shows tuples created on Presentation MathML, arrows showing the second symbol being located next to the first symbol. In tree-based models, one or both of SLT and OPT representations are used, and tree comparison models determine the similarity of the formulas. Finally, in embedding-based models, each formula is represented as a d-dimensional vector, and to calculate similarity measures such as cosine similarity are commonly used.

4.2.1 Text-Based Formula Search

In text-based approaches, formula tree representations are linearized to sequences appended to text and then traditional information retrieval models such as TF-IDF (term frequency-inverse document frequency) are applied for retrieval. Another approach is to use the LaTeX representation of a mathematical formula. Some existing text-based formula search models are summarized here:

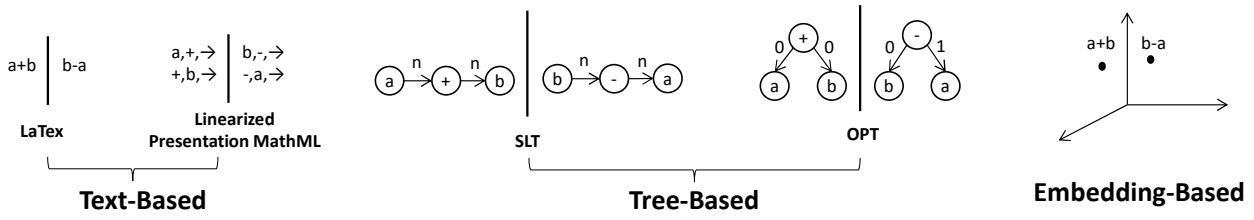


Figure 4.3: Text, tree, and embedding-based formula search approaches using formulas $a+b$ (on left side) and $b-a$ (on right side) as example. In text-based models, LaTeX representation or linearized presentation MathML can be used. In tree-based approaches, SLT and OPT representations can be used. In embedding-based models, each formula is mapped to a point in d-dimensional space.

- **MathFind** [114] uses Presentation MathML and breaks the math expression into a sequence of text-encoded fragments. Then each fragment is considered as a term in a traditional information retrieval model. This system applies normalization on the MathML input to address the alternative representations of a math formula.

- **EgoMath** [112] considers Presentation MathML and uses normalization of variables and constants. A formula is not represented as one word, but multiple in this system. The first representation is the ordered input formula and the next representations are created by applying transformation and generalization on the output of the previous step. As the later representations are more generalized, more formulas will match the query. The algorithm considers N representations that are appended to the simple textual query using the Boolean operator AND. The results are N sequentially executed search queries.

- **DLMF** (Digital Library of Mathematical Functions) [111] considers the LaTeX representation of the formulas and then uses TF-IDF for exact formula matching. They augmented an existing textual search engine for expression retrieval by implementing a keyword-based expression representation and developing a custom mathematical query language resembling TEX.

- **ActiveMath** [87] indexes math formulas in the OpenMath library by encoding them as sequences of semantic tokens using a custom syntax and storing them in a Lucene index. The system retrieves content, converting the formula query to a sequence of semantic tokens (similar to the indexing step) and comparing those tokens to entries found in the index.

- **MathWebSearch** [73] supports both MathML and OpenMath formula representations. Formulas are stored in a substitution tree, where each node corresponds to a function. A mathematical expression is represented as paths starting from the root of the tree. Retrieval is done

using a standard term retrieval algorithm for substitution trees. This system supports search for formulas that match the query terms up to α-equivalence (formulas that are equivalent for all purposes if their only difference is the renaming of the variables).

- **MIaS** (Math Indexer and Searcher) [148] uses term frequency-inverse document frequency (TF-IDF) for indexing XHTML documents containing MathML expressions. Math expressions go through several modifications and are broken into sub-expressions. Each math expression has a weight based on the level of modification. This system uses Presentation MathML.

- **WikiMirs** [56] considers Presentation MathML, after applying a preprocessing step on the LaTeX representation. It uses the term normalization process to generate original terms and generalized terms. Then, it uses an inverted index and a modified similarity score based on the TF-IDF scheme to evaluate the distance of the matched terms on different levels. In another work, Lin et al. [88] adapt TF-IDF retrieval for SLTs by using vectors of subexpressions, considering canonicalization to simplify expressions and to identify commutative operators and equivalences.

- Kumar et al. [78] uses the LaTeX representation of formulas and considers the Longest Common Subsequence between the query and the indexed formulas. This model needs a quadratic algorithm to evaluate all expressions.

- **MathDowsers** [116] is a recent text-based model that makes use of the Tangent-L [45] system that converts a formula into a bag of math tokens (tuples) that each capture local characteristics of the SLT. Three changes are made to the original Tangent-L system: 1) introducing repeat tokens, 2) changing the ranking function, and 3) normalizing formulas.

Text-based approaches lose the hierarchical nature of formulas and may fail to characterize formula structure well as a result. Previous studies have shown that these approaches are less effective [4, 170], and in this research, we do not make comparison with these models.

4.2.2 Tree-Based: Full and Sub-tree Matching Formula Search

In tree-based models, formulas are represented directly as trees, often with sub-trees to support partial matching. Systems have made use of SLT, OPT, or both tree representations of the formulas. Tree-based approaches can also be categorized into two groups: (1) using sub-tree matching, and (2) using full-tree matching. The following systems use sub-tree matching:

- **Tangent-S** [35] combines retrieval over both SLTs and OPTs. Candidates are first retrieved and scored using tuples representing relative paths between pairs of symbols [172]. The top-k candidates are then aligned with the query to produce formula similarity scores (the Maximum Sub-tree Similarity, MSS). SLT results and OPT results are next combined via linear regression over alignment measures from each representation to produce final similarity scores.

- **MCAT** [75] encodes path and sibling information from SLT and OPT representations, where paths act as the retrieval units. They also make use of a hashing-based formula structure encoding scheme, and text information at three levels of granularity. The first level considers words within a context window of size 10, descriptions, and noun phrases in the same sentence as the formula. The second level considers all the words in the paragraph where the formula appeared. At the third level, the title and abstract of the document, keywords in the document, descriptions of all the formulas, noun phrases, and all the words in the document are considered.

- **Approach0** [177] retrieves formulas using only paths from operator trees generated by parsing LaTeX with a relatively small expression grammar. Candidates are scored based on up to three best-matching sub-trees. In the second version of this system [179], two other similarity scores were combined with Approach0 scores: Textual similarity score using Lucene BM25 [66], and another structure-based scoring using IDF of paths and symbol similarity.

The other approaches use full-tree matching as their similarity score. The **SimSearch** [65] system, uses tree-edit distance (TED) on the symbol layout tree as the similarity measure by considering three edit operations: insertion, deletion, and substitution, to find the minimum cost of converting one tree to another. Their system includes TED accelerations, e.g., cost-based pruning of candidates and caching sub-trees. Operation costs are defined in a table (manually), using a set of conditions based on the similarities of node labels, their parent's label, and whether they are leaf nodes. For each condition, there is a cost defined for each edit operation. They provide some intuition as to why these assumptions are made (e.g., that replacing an operator has a larger impact on the interpretation of an expression than replacing a variable), but specific operation cost are not provided, aside from an inequality over three costs variables.

4.2.3 Embedding-Based Formula Search

With great advances in natural language processing and information retrieval tasks using embedding models, several formula search models are proposed using formulas embedding. In these works, different formula representations are embedded. Some of the works embed the linearzied tree representation of formulas, some use the image of the formulas, and some combine text and formula embeddings. Some of the systems are described here:

- The work proposed by Thanda et al. [154] from the Samsung group is the first known embedding model for math formulas, where a variant of the doc2vec algorithm [83] is introduced. They use binary expression trees and assigned each formula a real-valued vector such that formulas with similar structures are close to each other in vector space.
- Gao et al. [47] introduce embeddings for both symbols (**symbol2vec**) and formulas (**formula2vec**). Symbol2vec is based on a Continuous Bags-of-Words (CBOW) architecture using negative sampling, while formula2vec uses a distributed memory model of paragraph vectors [83].
- **Equation embeddings** [77] generates embeddings for both words and equations, with a larger context window size for equations than words. They also propose equation unit embedding, treating equations as sentences where the words are symbols, variables, and operators, each referred to as a "unit."
- **TopicEq** [166] generates the context from a mixture of latent topics, and the equation have generated by a recurrent neural network that depends on the latent topic activation and enables intelligible processing of equations by considering the relationship between the mathematical equations and topics.
- Pfahler and Morik [128] propose embedding formula images using graph convolutional neural networks and improved their representations using two self-supervised tasks (contextual similarity, and a masking task) to preserve information about the raw input symbols
- **MathBERT** [127] is pre-trained model for mathematical formulas, jointly trained with formulas (as strings), contexts, and their corresponding OPTs. Three pre-training tasks are defined, each capturing specific information. In the Masked Language Modeling task, the goal is to capture textual information. The Context Correspondence Prediction task aims to predict the relatedness of a context to a formula, being trained the same as the next sen-

tence prediction task. The last task is Masked Substructure Prediction, aiming to capture syntax-level structure of formulas, using OPT representations.

- **EARN** [2] considers image embedding (with LaTeX rendered as image) and a graph embedding of a math expression and combines the result using a simple linear regression model. The distances from graph-based and image-based embeddings are considered as the features.
- **Forte** (FOrmula Representation learning via Tree Embeddings) [160] considers an encoder-decoder model for mathematical formulas with a new tree beam search algorithm for the decoder. Using the OPT representation of formulas, the encoder does tree traversal to extract content information and also tree positional encoding to extract structural information.
- **XY-PHOC** [9] considers different levels of granularity to encode relative spatial locations of symbols in a formula. Using the cosine similarity measure, the embedding of the formula query is compared with the embeddings for the location of individual symbols stored in an inverted index for formulas in the collection.
- **NTFEM** (N-ary Tree-based Formula Embedding Model) [34] uses N-ARY tree representation for the formulas to keep the hierarchical structures of the operators and variables in the formula. They use a pair-based method to construct the words (tuple) and then apply the FastText n-gram embedding model to get the vector representation for each word (tuple). Two weighting algorithms are applied: 1) level weight, which considers a weighting for the influence at different levels of the tree, and 2) frequency weight, using smooth inverse frequency (SIF) [6] weighting process.

4.2.4 Contextual Formula Search

In related work, we have so far reviewed three categories of formula search methods. Our focus was on how similarity is decided in existing search models. Some of the models we discussed used both formula and their related context. For using context of formula, we consider three categories of researches: first, the research for extracting descriptions related to identifiers and symbols, second, the models for extracting descriptions and related text for formulas, and third, models using both formula and context for search.

In the first category, the goal is to find definitions and related descriptions for math symbols or identifiers. These approaches commonly rely on a set of rules based on clustering the related descriptions. Grigore et al. [51], did some of the earliest research in this area, aiming to disambiguate

symbols using the natural language text within which the symbols are, to resolve their semantics, thus enabling the disambiguation process. For this, they used a list of operators in OpenMath content dictionaries as concepts and used term clusters created on operator descriptions, to model their semantics. From each operator description, a bag of nouns is extracted and then manually enriched using terms from online lexical resources. The cluster that maximizes the similarity (based on Pointwise Mutual Information (PMI) and DICE) between nouns in the cluster and the local context of a target symbol is taken to represent its meaning. In a different work, Stathopoulos et al. [150], defined a new task, variable typing, where the goal is to assign a mathematical type (phrases from the technical terminology of the mathematical discourse) to variables using surrounding text in the same sentence. Using the example from their paper, for the sentence "Let P be a parabolic subgroup of ...", the variable P is assigned to "parabolic subgroup". In a similar work, Alexeeva et al. [5], introduced MathAlign, a rule-based approach built on the Odin information extraction framework [145]. Math identifiers and their descriptions are extracted based on a set of rules. Then, images of math formulas are converted to LaTeX strings, which are then segmented. The component identifiers in each segment are their extracted textual descriptions. Schubotz et al. [142], refereed to the context related to identifiers as namespace. A namespace is a topic cluster built on using K-means clustering of documents. Clusters beyond a certain purity threshold are selected and converted into namespaces by extracting phrases that assign meaning to identifiers in the selected clusters.

While the previous category focuses on math symbols, the second category aims to find relevant context for a formula. Kristianto et al. [74] aimed to find natural language descriptions for formulas by training an SVM model to classify whether a description candidate is associated with a formula or not. The SVM model performed better compared two simple baseline systems, one using the nearest noun and the other using sentence–patterns for context extraction. To extract context for formulas, Pathak et al. [125] considered the noun phrases of the sentence containing the formula as its context. Therefore, their proposed model is based on extracting candidate noun phrases and designing an algorithm to select the most proper one. To extract noun phrases, the Stanford Shift-Reduce Constituency Parser [181] is used to parse the sentences, and then using certain heuristics each noun phrase is assigned an initial weight which is later tuned using a development set containing gold contexts for target sentences. In a recent work, Yuan et al. [168] introduced MathDec, a hybird model for constructing descriptions for formulas. The model has two main modules: Selector and Summarizer. In the Selector, a Topic Relation Graph (TRG) obtains the relevant documents which contain the information related to the formulas. The TRG is a graph built according to the citations between formulas. In the Summarizer, the Integer Linear Programming (ILP) framework is used to summarize the descriptions. This module constructs the final description

using a timeline that is extracted from the TRG.

For contextual formula search, models can use both context and formulas. For example, the work by Krstovski and Blei [77] generates embeddings considering both words and equations using different context windows for each. While this work uses unified representation of text and formulas, models such as the one proposed by Ng et al. [116] separates formula and text search problems, doing search for each individually and then combining the search results linearly.

4.3 Isolated Formula Search

This section reviews the models we have introduced for isolated formula search. Our first model is based on an n-gram embedding model. As this model provides vector representation for the formulas, the similarity matching is less strict. Therefore, we also introduce a full-tree matching model that re-ranks the embedding models' results based on tree-edit distance. Finally, we describe our learning-to-rank model that leverages sub-tree and full-tree similarity features, along with vector similarity from our embedding model, for isolated formula search.

4.3.1 Tangent-CFT

Our first proposed model [105] for formula retrieval makes use of both SLT and OPT representations, as previous approaches such as Tangent-S [35] have shown considering both representations can help provide better search results. Tangent-CFT is the first embedding model that uses both representations. In addition to the full representations (which include both the type and the value of a node), this model also employs unification by ignoring the node values and just considering their type. Those representations are called SLT-Type and OPT-Type. For each of the representations, the model would then convert the tree representation to a vector and then, using vector representations, the formulas are retrieved in two different ways.

Here are the steps to train the Tangent-CFT model and do the retrieval:

1. **Tuple Sequence Generation.** Presentation MathML and Content MathML representations are used as a basis from which to generate internal SLT and OPT formula representations. Built using depth-first traversals with the Tangent-S [35] system, these internal tree representations consist of a sequence of tuples. Tangent-s generates tuples for pairs of symbols and their relative positions using tuples in form of (parent, child, path, path-from-root

Table 4.2: SLT and OPT tuples for formula $x - y^2 = 0$

SLT tuples	**OPT tuples (FRP omitted)**
(**V!** x, -, n, -)	(**U!** eq,**O!** minus,0)
(-, **V!** y, n, n)	(**O!** minus, **V!** x, 0)
(**V!** y, **N!** 2, a, nn)	(**V!** x, eob, 0)
(**N!** 2, eob, n, nna)	(**O!** minus, **O!** SUP, 1)
(**V!** y, =, n, nn)	(**O!** SUP, **V!** y, 0)
(=, **N!** 0, n, nnn)	(**V!** y, eob, 0)
(**N!** 0, eob, n, nnnn)	(**O!** SUP, **N!** 2, 0)
	(**N!** 2, eob, 0)
	(**U!** eq, **N!** 0, 1)
	(**N!** 0, eob, 0)

[PFR]) while traversing the tree depth-first. Parent and child are the nodes, path shows the edge label sequence when moving from the parent to the child node, and the PFR is the edge labels seen when traversing from the root of the tree to the parent node. The only difference for Tangent-CFT is that the path-from-root element produced by Tangent-S is ignored. Considering formula $x - y^2 = 0$, Table 4.2 shows the created tuples for this formula using SLT and OPT representations. In Tangent-S, the parameter window size, controls the maximum path length between the symbols. For Tangent-CFT we used a window size of 2. The End-of-Baseline (EOB) are the dummy pairs generated when the last symbol on each baseline is visited.

2. **Tuple Tokenization.** After converting the formula tree representation to a tuple sequence, each tuple can be considered as a word. To use an n-gram embedding model, 'words' and 'characters' should be defined. In our representation, each character (token) is encoded using a unique identifier. The identifiers used for edge labels are different from those used for the node values. Tokenizing the tuple elements can provide good insight into the formula structure. For instance, separating node type and value provides more details on formula structure, allowing two formulas sharing the same structure but different variables or constants to obtain a higher similarity score than they would with untokenized tuples (i.e. if tuples are treated as characters).

3. **Training Embedding Models with fastText.** After linearizing formulas, the fastText [17] n-gram embedding model is applied to embed the formula. Each encoded tuple is considered

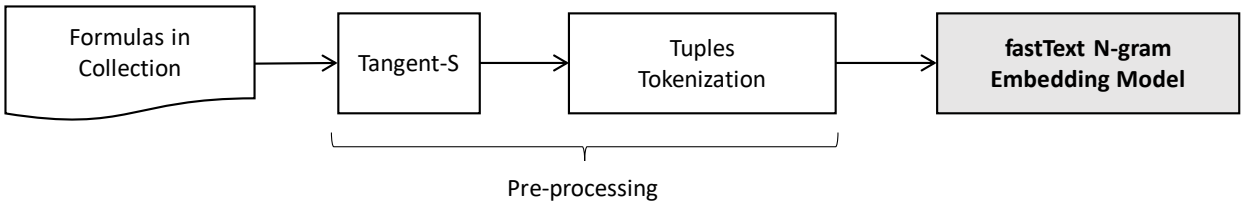


Figure 4.4: Training Tangent-CFT model. Formulas in the collection are linearized with Tangent-S system (with depth first traverse) and then tokenized. Each tuple is considered as a word and the tuple elements are its characters.

as a word, and the context window for a tuple *'T'* is defined by nearby tuples in the linearized tuple sequence. The context window size is a hyperparameter in the model, which is set to the default value of fastText model, 5. After the model is trained, each tuple is assigned a d-dimensional vector (which is defined before training). The vector representation for formula F with set of n tuples, T_F, is defined by mean pooling as:

$$formulaVec\,(F) = \frac{1}{n} \sum_{t \in T_F} tupleVec(t)$$

4. **Retrieval.** To compute the similarity of two formulas, the cosine similarity of their vector representations is used. First, the vector representation for each formula in the collection is created and then, for a given formula query, based on the trained model, the vector representation is obtained. Then, the similarity between the formula query q with vector V_q and a formula f in the collection with vector V_f is measured as follows:

$$sim\,(q, f) = \frac{V_f \cdot V_q}{|V_f|\,|V_q|}.$$

A higher cosine similarity results in a better retrieval rank. Figure 4.5 shows how the cosine similarity is calculated given a formula query and the candidate formula.

We have proposed two different ways to do the formula search with the pipeline above. In our first model (the original Tangent-CFT model) three representations are considered for each formula: SLT, OPT, and SLT-TYPE. The vector representation for each formula is obtained by considering the unwieghted element-wise summation of these three vectors.

In the second version of the system, Tangent-CFT2, the retrieval is done separately with four representations: SLT, OPT, SLT-Type, and OPT-Type. First, the top-k results are retrieved. This is done by computing the cosine similarity between each query vector representation

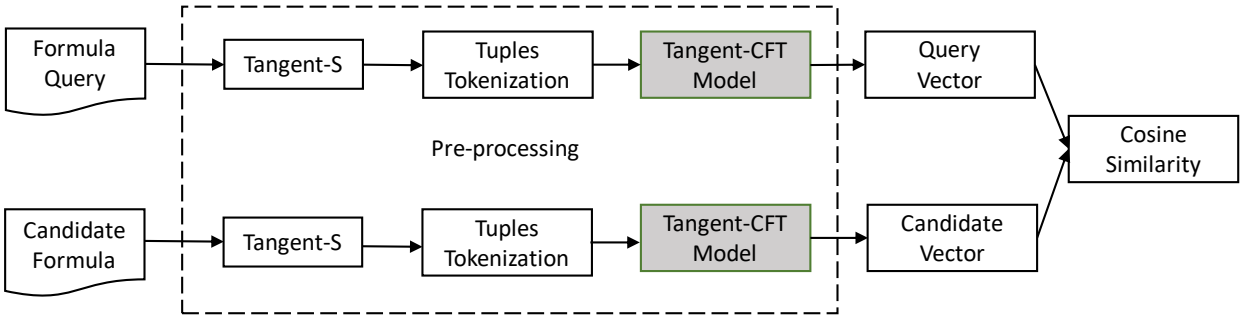


Figure 4.5: Retrieval with Tangent-CFT model. Query and candidate formulas are passed to a same pre-processing pipeline to extract their vector representations. Cosine similarity between these two vectors is the similarity score.

and corresponding the vector representation of formulas in the collection. The top-k (in our experiments, k=1000) results from each representation are computed as the cosine similarity between the vectors. Then the four results are combined using modified Reciprocal Rank Fusion (RRF) [31] with the following formula:

$$RRFscore(f \in F) = \sum_{m \in M} \frac{s_m(f)}{60 + r_m(f)} \tag{4.1}$$

where f is a set of formulas to be ranked, M is a set of models, and s_m and r_m are the scores and the rank, respectively, of the retrieved formula by model m. Note that in the original RRF, only the rank is considered and the numerator of equation (4.2) is 1. Our study showed using the modified RRF provides better rankings than original RRF. Experiment results showed Tangent-CFT2 has better effectiveness than the first version of the system [103].

$$RRF(f \in F) = \frac{S_{Original}(f)}{60 + R_{Original}(f)} + \frac{S_{MathAMR}(f)}{60 + R_{MathAMR}(f)} \tag{4.2}$$

While Tangent-CFT model uses exhaustive search over all the collection to find similar formulas, we have also introduced, **MathFIRE** (Math Formula Indexing and Retrieval for ElasticSearch), a framework for indexing and retrieving formulas extracted using the MathSeer Extraction pipeline. MathFIRE uses OpenSearch [29] to index and retrieve formulas in these different representations. In this system, formula vector representations are extracted with Tangent-CFT, then loaded in OpenSearch [5] where dense vector retrieval was performed by approximate k-NN search using nmslib and Faiss [63]. We used the default parameters.

[5] `https://opensearch.org/`

4.3.2 Tangent-CFTED

The experiments on the first proposed model, Tangent-CFT, showed that this system does better on partial matching compared to full matching. It focuses on common n-grams and therefore for queries such as $O(mn\log m)$, gives higher rank to formula $O(mn)$, compared to $O(KN\log N)$. To address that issue, we extended Tangent-CFT by re-ranking the retrieval results based on tree-edit distance, introducing Tangent-CFTED [102]. Tree Edit Distance (TED) is defined as the minimum-cost sequence of node, or node and edge, edit operations to transform one tree into another. In this study, three kinds of editing operations are considered:

- **Insertion:** inserting a node/edge. If a node is inserted between a parent and its children, the new node will become the parent of the child node.
- **Deletion:** deleting a node/edge. If a deleted node has children, they will be connected to the deleted node's parent.
- **Substitution (i.e., relabeling):** a single operation where a deletion and an insertion happen simultaneously on the same element (i.e., on the same node or the same edge).

Note that if edges are considered, then insertion/deletion of a node also requires insertion/deletion of an edge. The unweighted TED between trees T_1 and T_2 is defined by:

$$TED(T_1, T_2) = \underset{s \in S(T_1, T_2)}{\operatorname{argmin}} |s| \tag{4.3}$$

where $S(T_1, T_2)$ is the set of all edit operation sequences that transform T_1 into T_2, and $|\cdot|$ gives the sequence length.

An alternative approach, the weighted TED, allows non-uniform costs for editing operations. Kamali and Tompa [65] designed such weights by hand; by contrast, in our work, those weights are learned using cross-validation. The weights were learned using grid search for each edit operation, 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 [170] there are 20 formula queries without wildcards. Operation weights were learned for each query independently and then averaged over the 20 queries. We used the values [0,1] with step size of 0.1 for each edit operation. The goal was to maximize the harmonic mean bpref [23] score, which balances full and partial bpref scores. Note that each assessed hit in NTCIR-12 had a relevance score between 0 to 4, 1 and 2 indicating degrees of partial relevance, and 3 and 4 full relevance.

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

1. Retrieve the top-1000 results with the Tangent-CFT model.
2. Re-rank the results with tree-edit distance using full SLT and OPT representations (not type representations). For SLTs, the edit operation costs 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. The ranking score for each retrieved formula is calculated as:
$$sim(T_1, T_2) = \frac{1}{TED(T_1, T_2) + 1}. \tag{4.4}$$
3. Combine the re-ranked results from OPT and SLT representations. For this we propose two methods: first using a linear combination, learning the weight in a similar way to the way the weight for edit operations were learned. Therefore, the retrieval results would be combined as:
$$Score_q(f) = \alpha \cdot SLT\text{-}Score_q(f) + (1 - \alpha) \cdot OPT\text{-}Score_q(f) \tag{4.5}$$
with $\alpha = 0.95$. In an alternative approach (Tangent-CFT2TED) [103], we combine the results with modified Reciprocal Rank Fusion as in equation (4.2). Tangent-CFT2TED, re-ranks the results from Tangent-CFT2 system.

The tree-edit distance values are calculated using a publicly available Python package for the APTED algorithm [126].

4.3.3 Learning-to-rank

In the Tangent-CFTED model, we re-ranked the results from our embedding model with a full-tree similarity score, tree-edit distance. ARQMath-1 results [173] showed Tangent-S, that uses sub-tree similarity features, has better effectiveness than our two previous models. Therefore, we proposed a learning-to-rank [107] model that learns weights for each of the possible similarity features we can have for the formulas. Our model uses SVM-rank [62] with a linear kernel because of its effectiveness with small samples of training queries, and its speed for combining the results (e.g., using a map-reduce framework). Because we use a linear kernel SVM, the weights of the features can also be interpreted. SVM-rank is trained on the 29 queries with relevance judgments for 3,909 formula instances that are provided in the ARQMath-1 (explained in the next section) [173] test collection. The penalty for misclassification during training (C) was 0.01, and the tolerance (epsilon) was 0.001 for termination criterion.

The following feature sets were used in training (names in parentheses are used to refer to that feature set):

- 4 Tuple matching scores (Tuple): harmonic mean of path match percentage on query and candidate formula on full and type SLT and OPT representations.
- 4 Node Matching scores (Node Matching): harmonic mean of symbol match percentage on query and candidate formula on full and type SLT and OPT representations.
- 2 Maximum Sub-tree Similarity (MSS) scores: after greedy alignment of query on the candidate, harmonic mean of matched unified symbols and edges.
- 4 Unweighted tree edit distance scores (UTED): minimum cost for converting one tree to the other with three unweighted edit operations of deletion, insertion and substitution.
- 4 Weighted tree edit distance scores (WTED): minimum cost for converting one tree to the other with three weighted edit operations of deletion, insertion and substitution.
- 4 Cosine similarity scores from the Tangent-CFT model (CFT): cosine similarity between query and candidate formula vectors from the Tangent-CFT model.
- 7 Other tree and symbol features (OTS)

Here we review the features:

Sub-tree matching features. To do sub-tree matching, we used the initial candidate selection method from the Tangent-S system. Using the tree representation of a formula, a list of tuples is created from a tree using a depth-first traversal. After the tuples are generated, the harmonic mean of the ratio of query tuples matched (recall) and the ratio of candidate tuples matching the query (precision) is used for ranking.

Another feature that we use to compare sub-trees is the Maximum Sub-tree Similarity (MSS) score introduced in the Tangent-3 system [172]. MSS is computed from the largest connected match between a query and candidate formula obtained using a greedy algorithm, evaluating pairwise alignments between trees using unified node values. Finally, we consider the harmonic mean of symbol match percentage on query and candidate formula on SLT and OPT representations.

Full-tree matching features. Another approach computes similarity using full formula trees, making the matching criterion more strict. We use tree-edit distance as a full-tree matching fea-

ture. The same approach as described in the Tangent-CFTED system is used by considering both unweighted and weighted tree-edit distance.

Embedding features. While the features described above use exact tree representations, embedding features provide dense vector representations for each formula that can provide better matching scores for partially relevant formulas [105]. For the embedding features, we trained the n-gram embedding model used in the Tangent-CFT system. We considered the cosine similarity score as a feature.

Other tree and symbol features (OTS). This is an additional set of simple features to compare formulas directly. These include:

1. Ratios of matching variables, numbers, and operators between the query and candidate formula.
2. Difference between the number of nodes in query and candidate in SLT and OPT.
3. Difference between OPT depth.
4. Difference between SLT variation from the geometric baseline, defined as the maximum number of edge labels visited when moving from a node to the root of the tree which do not have 'next' as their edge label. For instance, for the example in Figure 4.4, this value is one.

For feature sets other than the 'OTS' and 'MSS', one feature is calculated for each of the four tree representations (SLT, OPT, SLT TYPE, OPT TYPE). As 'MSS' features use unification, the full and type representations would have the same scores, so we have only two features in that set. All feature values are MinMax normalized to be in the range [0,1]. For the sub-tree features, the default Tangent-S parameters were used. The learned weights for deletion, substitution and insertion operations in the weighted TED are respectively: SLT (0.43, 0.20, 0.41), OPT (0.31, 0.21, 0.29), SLT TYPE (0.37, 0.29, 0.33), and OPT TYPE (0.34, 0.25, 0.32). We learnt these weights using additional 29 topics from ARQMath-1 formula search task.

4.3.4 Evaluation of Models

For evaluation, we rely on the ARQMath-1, -2 and -3 test collections.[6] Table 4.3 shows the effectiveness of our models. For our learning-to-rank model, we used the 29 additional topics in

[6]Tangent-CFT results are reported on NTCIR-12 in the related paper [105]

Table 4.3: Results for Contextual Formula Search Task on ARQMath-1 (45), ARQMath-2 (58), and ARQMath-3 (76) topics. Tangent-S is the baseline system. Note that for LtR model for each test collection we used different training set.

	Evaluation Measures								
	ARQMath-1			ARQMath-2			ARQMath-3		
	nDCG′	MAP′	P′@10	nDCG′	MAP′	P′@10	nDCG′	MAP′	P′@10
Tangent-S	0.691	0.446	0.453	0.492	0.272	0.419	0.540	0.336	0.511
Tangent-CFT2TED	0.648	0.480	0.502	**0.580**	**0.381**	**0.545**	**0.694**	**0.480**	**0.611**
Tangent-CFT2	0.607	0.437	0.480	0.565	0.364	0.516	0.641	0.412	0.534
LtR	**0.736**	**0.522**	**0.520**	0.548	0.342	0.539	0.575	0.377	0.566

ARQMath-1 to get results for ARQMath-1 test topics, we used all ARQMath-1 topics to get results for ARQMath-2, and finally used all topics for ARQMath-1 and -2 to get results for ARQMath-3. As shown in this table, the models work better for ARQMath-1 compared to ARQMath-2 where there was more attention to the context, however for ARQMath-3 our systems had better effectiveness. All these models, only consider formulas in isolation to find similar ones.

Looking at ARQMath-1 results, our learning-to-rank framework has better effectiveness compared to the Tangent-S system (baseline), in particular for P′@10 which is higher for all of our systems. The differences between none of our systems were significant in terms of P′@10 and MAP′. However, with nDCG′, the differences between Tangent-CFT2 and the learning-to-rank model are significant for ARQMath-1 ($p < 0.05$, two-tailed paired t-test with Bonferroni correction). Tangent-CFT2, Tangent-CFT2TED, and Tangent-S take different approaches for formula retrieval using embedding, full-tree, and sub-tree matching. Our learning-to-rank model uses SVM-rank to linearly combine similarity measures from each of these models, to overcome their limitations.

Comparing three systems from the Tangent family, Tangent-CFT2, an n-gram embedding model, focuses on retrieving formulas that share common n-grams with the input query. This can be beneficial for formulas such as $\sum_{n=0}^{N} nx^n$, which are less complex, so perhaps finding formulas that share the same n-grams such as the same variables or numbers with the query can be effective. Both Tangent-S and Tangent-CFT2TED return not-relevant formulas such as $\sum_{n=0}^{N}(-1)^n x^n$ in their top-10 results, which have SLTs and OPTs that are similar to the query, but are judged as not relevant. For this query, the P′@10 for Tangent-CFT2 was 0.9, it was 0.4 for Tangent-CFT2TED, and 0.6 for Tangent-S. Tangent-CFT2TED uses tree-edit distance as a full-tree matching score.

Looking at full trees provides better results for formulas such as:

$$\emptyset, \{1\}, \{2\}, \{1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{4\}, \ldots$$

where partial matches are unlikely to provide useful information. The P′@10 values for this query for Tangent [-CFT2TED,-CFT2,-S] are 0.6, 0.4 and 0.3, respectively. Finally, Tangent-S is a system using sub-tree matching, and for complex formulas such as:

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

finding sub-tree matches can be useful, returning highly relevant formulas such as:

$$\iint\limits_{D_{x,y}} f(x,y)dxdy = \iint\limits_{D_{u,v}} f\left(T(u,v)\right)|J(u,v)|dudv,$$

that the other two models did not return in their top-1000 results. Tangent-S has P′@5 of 0.5 for this formula, whereas P′@5 is 0.3 for both Tangent-CFT2 and Tangent-CFT2TED.

From these examples, we can see that each of these models have their strengths and limitations. With our learning-to-rank model, we re-rank Tangent-S results using similarity scores from multiple retrieval models. For example, for the query $\text{lcm}(n_1,n_2) = \frac{n_1 n_2}{\text{gcd}(n_1,n_2)}$ Tangent-S ranks not-relevant formulas such as $L = \text{lcm}(n_1,n_2,\ldots,n_k)$, that share sub-trees with the query in its top-10 results. Using our proposed learning-to-rank model, relevant formulas such as $\text{lcm}(a,b) = \frac{a \cdot b}{\text{gcd}(a,b)}$ are pushed to the top-10 results. As can be seen, the first formula can be converted to the second with a pair of substitutions (a for n_1, b for n_2) and removing the multiplication dot in the second formula.

The learning-to-rank models use only formula tree similarity features. However, some formulas need more features and processing. For instance, for the query:

$$Empty(x) \iff \nexists y(y \in x)$$

there are relevant formulas such as $\vdash \exists x \forall y(y \notin x)$ that may not necessarily share SLT or OPT structure with the query. Perhaps using canonicalization methods [120] can improve the effectiveness for these queries to convert them to a unified form. While we focused on structural similarity features, there are still formulas for which the effectiveness is low. Textual features are another missing part of our current model. There are queries such as $\frac{df}{dx} = f(x+1)$, appearing in a question related to differential equations, for which returning a structurally similar formula such as $\frac{dy}{dx} = f(x)$ is considered not-relevant due to its appearance in a different context (a post on another topic). This motivates using context for formula search, which we will explain in the next section.

Looking at ARQMath-2 and -3 results, all our proposed models significantly outperform the baseline system in all three effectiveness measures ($p < 0.05$, two-tailed paired t-test with Bonferroni correction). Tangent-CFT2ED outperformed our other proposed models. In ARQMath-2 for 5

topics out of 58 and for 2 topics out of 76 in ARQMath-3, P′@10 for Tangent-CFT2ED was 0. On average, these topics had 5.1 formulas assessed as high or medium relevant among the assessed formulas. 4 of these topics were low-complexity formulas, and Tangent-CFT2ED gave higher a rank to formulas that had small differences in the symbols compared to formula query. For example, for query $xRy \vee yRx$ (B.362 in ARQMath-3) there are formulas such as $xRy \wedge yRx$, $xRy \Leftrightarrow yRx$, $xRy \not\Rightarrow yRx$, and $xRy \iff yRx$ retrieved as top results, but none are considered as relevant. Another observation is that Tangent-CFT2ED is an isolated formula search model that ignores context. There are queries such as $x^n + y^n + z^n$ (B.289 in ARQMath-2), for which our model finds exact matches, but as the contexts are different (such as constraints on the variables), they are assessed as not-relevant. This indicates that models should also make use of context, and in our next section we address this problem.

Another observation from ARQMath-2 and -3 is that the effectiveness of our learning-to-rank model drops compared to ARQMath-1. Compared to Tangent-CFT2ED, which only uses tree-edit distance similarity scores, there are topics for which learning-to-rank can provide better retrieval results. For query $[T_X^{0,1}, T_X^{0,1}] \subset T_X^{0,1}$ (B.304 in ARQMath-3), the P′@10 increases from 0.1 to 0.8 when learning-to-rank model is used instead of Tangent-CFT2ED. There are relevant formulas such as $T_U^{0,1}$ and $T^{0,1}X$ that learning-to-rank can retrieve in the top-10 results, where Tangent-CFT2ED did not retrieve in the top-1000. Instead, Tangent-CFT2ED retrieved not-relevant formulas, such as $E_2^{0,1} = E_\infty^{0,1} = F^0H^1$ in the top-10 results. In contrast, in cases that the learning-to-rank model did not perform better than Tangent-CFT2ED, using features such as sub-tree matching score leads to less effective results. For query

$$\lim_{x \to c} f'(x) = L = \lim_{x \to c^+} f'(x) = \lim_{x \to c^-} f'(x)$$

(Topic B.398 in ARQMath-3), the P′@10 drops from 0.9 using Tangent-CFT2ED to 0.4 when the learning-to-rank model is used. Not-relevant formulas such as "$\lim_{x \to a} f_1(x) = \lim_{x \to a} f_2(x) = ... = \lim_{x \to a} f_n(x) = N$" that share common sub-trees with the query are retrieved in the top-10 results of the learning-to-rank model.

4.4 Contextual Formula Search

We consider the problem of *contextual* formula search, where a formula selected from a document is used as a query, and formulas in documents are returned. For example, a formula might be selected

from a question post on a Community Question Answering (CQA) site, and then issued as a query to find posts providing information about the formula and/or the formula's associated question. In contextual formula search, the relevance of retrieved formulas is defined by both formula similarity *and* the interpretation of formulas where they appear.

For retrieval, using context can be helpful in two ways. First, context features can improve recall through characterizing a formula's meaning, or a setting where it might be employed. Second, context can improve precision through including constraints, such as on the range, domain, or types of variables [108]. Given this, we expect that formula search systems exploiting both a formula and its context would yield better results than systems that ignore formula context.

Mathematical notation is hierarchical, and so is represented well by trees. Text by contrast is linear, but the latent meaning of text is complex, and so graph structures have been devised to encode aspects of that meaning. We consider one semantic encoding for text, the Abstract Meaning Representation (AMR) graph. For formula representation, we use Operator Trees (OPTs). We unify AMR and OPT representations to capture the meaning of a formula in its context, which we call *MathAMR*.

4.4.1 Abstract Meaning Representation

Semantic parsing aims to construct semantic representations from language. Abstract Meaning Representation (AMR) was introduced in 1998 by Langlade and Knight in their Nitrogen system [81] to map meanings onto word lattices. Banarescu et al. [11] used PropBank [18] notation and defined AMR annotations as rooted Directed Acyclic Graphs (DAGs). DAG nodes represent core concepts in a sentence, either as words (typically, adjectives, stemmed nouns, or adverbs) or frames extracted from PropBank. Directed labeled edges represent semantic relationships between nodes using more than 100 relations, including frameset argument index (:ARG0, :ARG1), semantic relations (:location, :name), and relations for date-entities, relations for ordered listing (:op1, :op2, :op3). To generate AMRs from text, different parsers have been proposed. We consider three categories of AMR parser:

1. **Graph-based AMR parser.** These parsers build the AMR graph by treating AMR parsing as a procedure for searching for Maximum Spanning Connected Subgraphs (MSCGs) from an edge-labeled directed graph of all possible relations. The first AMR parser was of this type (JAMR [43], in 2014).

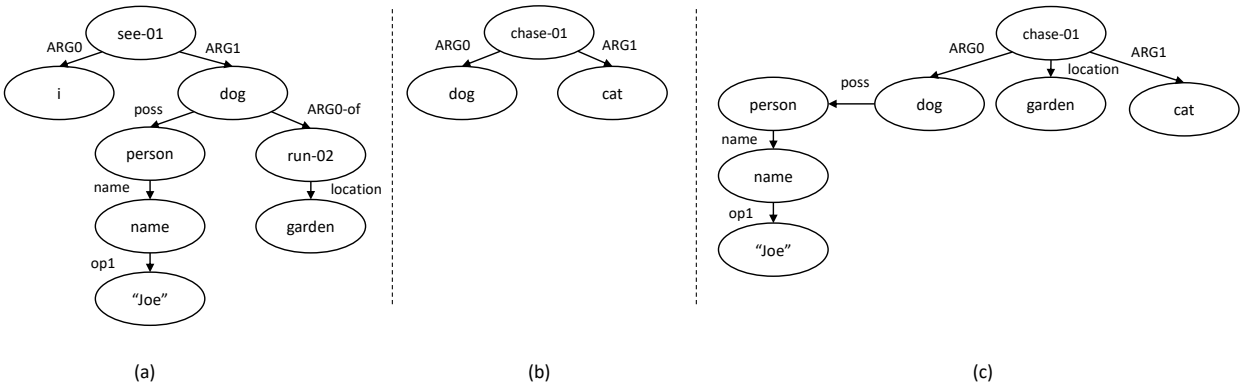


Figure 4.6: AMR summarization example adapted from Liu et al. [89]. (a) AMR for sentence 'I saw Joe's dog, which was running in the garden.' (b) AMR for a following sentence, 'The dog was chasing a cat.' (c) Summary AMR generated from the sentence AMRs shown in (a) and (b).

2. **Dependency-based AMR parser.** Parsers such as CAMR [159] first generate a dependency parse of a sentence, and then transform it to an AMR graph using transition rules.

3. **Neural AMR parser.** Viewing the problem as a machine translation task, neural models directly convert raw text to linearized AMR representations. For example, the SPRING parser [13] uses Depth-first linearization of AMR, and treats the problem of AMR generation as a text-to-AMR translation problem. SPRING leverages a BART transformer model [84] by modifying its tokenizer to handle AMR tokens.

AMRs are commonly used in tasks such as summarization [86, 89], question answering [19, 68, 164], and information extraction [49,175]. For example, Liu et al. [89] summarized text by first generating AMRs for individual sentences in a document, and then merging them by collapsing named and date entities. Next, a summary sub-graph was generated using integer linear programming, and finally summary text was generated from that sub-graph using JAMR [43]. Figure 4.6 shows an example summary of two sentences in their AMR representations [89]:

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

(b) The dog was chasing a cat.

Figure 4.6(c) shows a summary AMR generated for sentences (a) and (b).

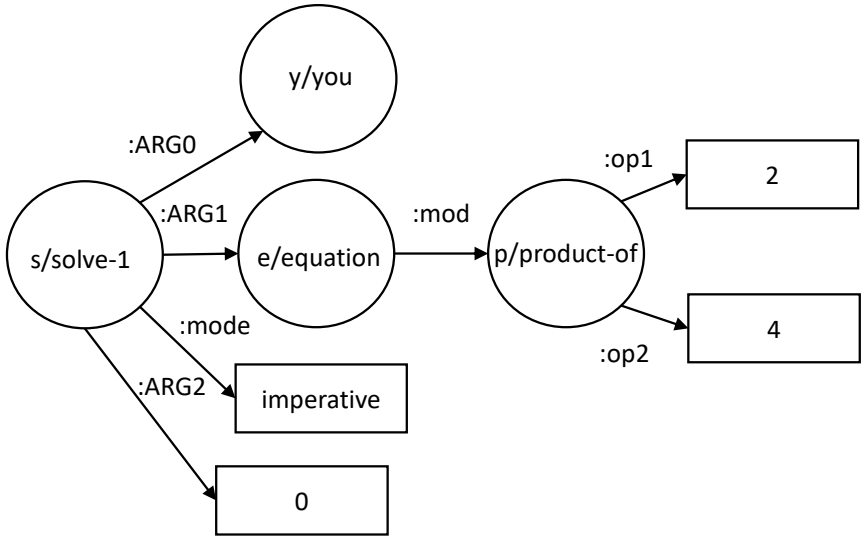


Figure 4.7: AMR with incorrect formula representation for "Solve the equation $x^2 - 4 = 0$".

For question answering, recent work by Bonial et al. [19], introduced InfoForager that uses AMR representation for answering medical research questions, focusing on COVID-19 topics. This system simply converts the question and possible answers to AMR and uses Smatch [25] as the similarity measure for ranking the answers. The Smatch score of two AMR graphs is defined in terms of their matching triples (edges) by finding a variable (node) mapping that maximizes the count of matching triples. The Smatch score is the harmonic mean of precision and recall for matching triples (F_1 score).

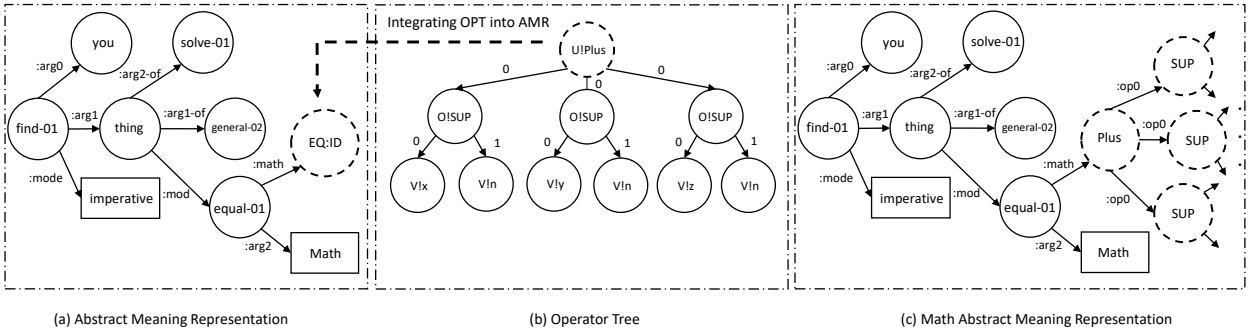
Existing AMR parsers fail to correctly parse math formulas. For example, for the expression: "Solve the equation $x^2 - 4 = 0$", using a BART-based AMR parser,[7] the generated AMR is shown in Figure 4.7. To address this, we introduce MathAMR, in which math formulas are represented using their operator tree representation.

4.4.2 MathAMR

By looking at an operator tree, one can see which type of mathematical operation is being applied to which subexpressions (operands). This is very similar to the representation of text with AMRs, which can roughly be understood as representing "who is doing what to whom".[8] AMR graphs

[7] We used amrlib python library and its xmf-bart model

[8] `https://github.com/amrisi/amr-guidelines/blob/master/amr.md#part-i-introduction`



(a) Abstract Meaning Representation (b) Operator Tree (c) Math Abstract Meaning Representation

Figure 4.8: Generating MathAMR for "Find $x^n + y^n + z^n$ general solution" (ARQMath-2 topic B.289). (a) AMR tree with formula replaced by formula id. (b) OPT formula representation. (c) OPT root replaces formula id. Part of the OPT in (c) not shown.

and operator trees also share similar edge labelings: in AMR the edge labels 'opx' indicate node ordering, where 'x' is an integer enumerator.

As current AMR parsers are not able to parse formulas correctly, we introduce MathAMR. We demonstrate the process of generating MathAMR using topic B.289 from ARQMath-2. This formula appeared in a question post title, as: "Find $x^n + y^n + z^n$ general solution". Figure 4.8 illustrates the steps used to generate MathAMR, which we describe below.

1. **Selecting a context window.** The first step is to decide a formula's context. As AMR was designed for sentences, we use Spacy[9] to split paragraphs into sentences and choose the sentence the formula appears in. Formulas are often delimited in LaTeX by '$'. It is common to see sentence punctuation inside formula delimiters, so before Spacy we move any punctuation (. , ! ?) from the end of formula regions to after the final delimiter. For example, "$a+b=c.$" becomes "$a+b=c$.".

2. **Replacing formulas with an identifier token.** To avoid AMR parsing problems, we replace formulas with a single identifier. Currently, we simply enumerate formulas. For the example, the formula $x^n + y^n + z^n$ is replaced with EQ766, with 766 being the integer formula id.

3. **Generating the AMR.** A neural AMR parser (BART-based [84]) is then used to generate the sentence AMR graph (e.g., as shown in Figure 4.8 (a)). We introduce a new edge label 'math' to connect a formula's placeholder node to its parent.

[9] `https://spacy.io/`

4. **Integrating the OPT representation.** The formula OPT shown in Figure 4.8 (b) is inserted into the AMR by replacing the formula id node with the OPT root, creating the *MathAMR* graph. This is shown in Figure 4.8 (c). To follow AMR conventions, we rename OPT edge labels from numbers to 'opX', where 'X' is the integer from the original OPT.

This is our first attempt at a unified representation of text and formulas in AMR graphs, so we have kept our model simple. We use only OPT formula representations, whereas our isolated formula search models have shown that also using SLT representations can sometimes be helpful [105, 107]. For some domains (e.g., biomedical), there are specialized pre-trained AMR parsers [109], but we used an AMR parser trained on general text, not specifically on mathematical text.

4.4.3 Using MathAMR for Formula Search

After generating MathAMR for the query formula and candidate formulas, we use depth-first traversal to linearize the MathAMR. For simplicity, we ignored MathAMR edge labels. For Figure 4.8 (c) (with the full OPT in Figure 4.8 (b)), the linearized MathAMR string is:

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

MathAMR strings are embedded as vectors using Sentence-BERT [133], and we perform retrieval in that embedding space using Sentence-BERT's cosine similarity. Sentence-BERT is a modified version of the pre-trained BERT network that generates contextual sentence embeddings that may be compared using the cosine-similarity measure. To train Sentence-BERT models, we used `all-distilroberta-v1` as a pre-trained model. This model is trained on one billion sentence pairs from different resources such as Yahoo Answers [174] and MS MARCO [117]. It uses Byte-Pair Encoding (BPE) [144] for tokenization. BPE first uses a pre-tokenization step and then generates a primary vocabulary consisting of all the symbols in the unique words. Using a recursive approach, vocabularies are merged based on a set of learned rules until a certain vocabulary size is obtained. For a sentence such as "find value of Math sqrt a SUP 4", the generated tokens are: {find, Ġvalue, Ġof, ĠMath, Ġsq, rt, Ġa, ĠSUP, Ġ4}, where 'Ġ' indicates the end of a token.

Relevance judgement in ARQMath are graded as high, medium, low, or not-relevant. For fine-tuning, we assigned a relevance score of 1 for high and medium, 0.5 for low, and 0 for not-relevant.

Training data comprised triplets of the form: `(query formula, candidate formula, relevance score)`.

For training, we adopted a multi-task learning framework provided by Sentence-BERT [10] that was used previously for detecting duplicate Quora questions with the *distilbert-base-nli-stsb-quora-ranking* model. The framework combines two loss functions. First, a contrastive loss [28] function minimizes the distance between positive pairs and maximizes the distance between negative pairs, making it suitable for classification tasks. The second loss function is multiple negatives ranking loss [54], which considers only positive pairs, minimizing the distance between positive pairs out of a large set of possible candidates, making it well-suited to ranking tasks.

We expected using these two loss functions would result in a system that distinguishes well between relevant and not-relevant candidates, and learns to rank relevant candidates. In each epoch, we compute the Spearman correlation between the embedding cosine similarity and the relevance score on the validation set. After training the model with a fixed number of epochs, the model with the lowest validation loss is considered as the final model used for retrieval.

In addition to using Sentence-BERT, we also use the Smatch score for retrieval. We show how a triple matching similarity score on MathAMR works for formula retrieval without using a trained model on ARQMath test collections.

4.4.4 Evaluation of Models

This section provides the experiment result on the MathAMR. First, we explain our training process and the parameters used for Sentence-BERT. Then, in our first study, we investigate using Sentence-BERT and compare it with Smatch score using assessed hit in the ARQMath. We then explore the effect of data used for fine-tuning Sentence-BERT, comparing ARQMath-1 and -2 results. After that, we study one of the important design choices for MathAMR, context window. Finally, we study how MathAMR can be used for re-ranking results from other formula search systems.

For fine-tuning we use relevance judgments for all 74 assessed ARQMath-1 topics, plus the 12 training topics from ARQMath-2, for a total of 21,411 training triples. To report our results on ARQMath-3, we used all the 155 topics from ARQMath-1 and -2 for training. Our results are reported on the 58 ARQMath-2 test topics and the 76 test topics from ARQMath-3. We report nDCG′ [139] and P′ at cutoffs of 5 and 10. These measures are nDCG@k and P@k for k={5,10}

[10] `https://www.sbert.net`

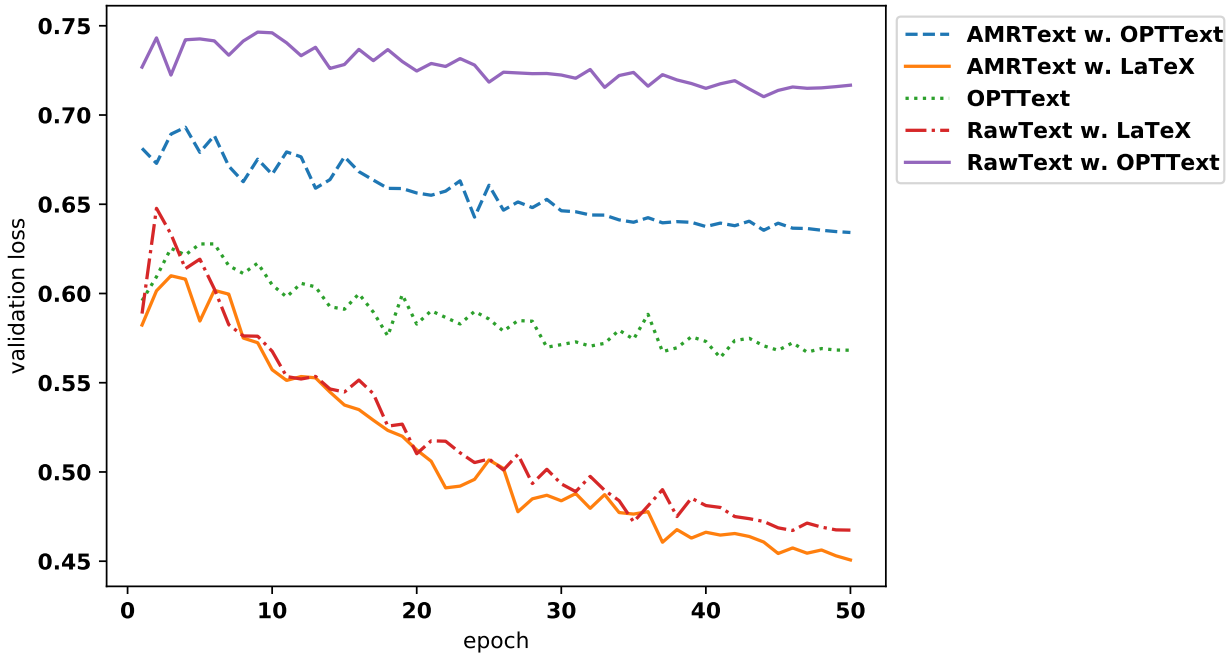


Figure 4.9: Learning curves for AMRText w. OPTText, AMRText w. LaTeX , OPTText, RawText w. LaTeX, and RawText w. OPTText.

after removing unjudged results from the ranked lists. To calculate P′@k, we treat only high and medium relevance as relevant, per the ARQMath evaluation protocol. Following the ARQMath protocol, all measures are calculated on visually distinct formulas.

Sentence-BERT Parameters. To fine-tune Sentence-BERT we used at most 50 epochs, we chose the best model by training loss on a validation set. We divided the training set into 10 folds, consistently using the same fold for validation. Figure 4.9 shows the loss validation for five representations with 50 epochs for ARQMath-2. Even though the learning curve has a low fluctuation for all the representations, the validation loss is higher for MathAMR compared to other representations.

For candidate formulas, the average length in linearized MathAMR strings in tokens was 53.2 (σ = 52.8) and for RawText with OPT it was 96.3 (σ = 141.6). We used batch size 16 and maximum sequence length 128 tokens for all representations other than RawText with OPT, for which we used a maximum of 256 tokens.

Preliminary MathAMR Experiments Using Assessed Hits

As our preliminary experiments, we study the effect of using MathAMR for ranking assessed hits in the ARQMath test collections, reviewing the effectiveness of the Sentence-BERT model and the Smatch score.

1. Context Representation Results (Table 4.4). To study the effectiveness of the MathAMR representation, we consider four other representations similar to MathAMR. For simplicity, we compared these models using only assessed formula hits; this may produce a bias compared to full ranking of the collection, as all relevant formulas are in the set being ranked. Our baseline uses only linearized OPTs (*'OPTText'*), ignoring surrounding text. *'AMRText w. OPTText'* corresponds to Figure 4.8 (c). For the model *'AMRText w. LaTeX'*, we produce AMRs as shown in Figure 4.8 (a), and replace the formula placeholder node with the original formula LaTeX. The final two representations use raw text with the original LaTeX or linearized OPTs. Note that the tokenizer for RawText can correctly handle formula operators (e.g., '$a+b=c$' is tokenized as: $[a, +, b, =, c]$). The same Sentence-BERT setting (maximum sequence length, batch size, and number of epochs) is used for all conditions other than *'RawText w. OPTText'*.

Table 4.4 compares effectiveness measures for MathAMR representations with effectiveness for other context representations. MathAMR (i.e., *AMRText w. OPTText*) does best by both measures, with both rank cutoffs, on average over the 58 ARQMath-2 test topics. The closest competitor is the *RawText w. OPTText* condition. Except for that *RawText w. OPTText* condition, MathAMR results were significantly better than other representations by all four measures ($p < 0.05$, two-tailed paired t-test with Bonferroni correction).

Stratifying our analysis using high/medium/low complexity topic labels distributed with the test collection, we see that both the *RawText w. OPTText* and *AMRText w. OPTText* conditions have identical P′@10 for low complexity topics, and that MathAMR's superior results come entirely from a better average P′@10 on medium and high complexity topics. For several topics, MathAMR achieves better results due to its selective focus on the text in the sentence containing the formula, rather than the whole input text. For example, for the formula query in the sentence, "How to show $(a_1 a_2 \dots a_n)^{\frac{1}{n}} \leq \frac{\sum_{i=1}^{n} a_i}{n}$", P′@10 increases from 0.1 for *RawText w. OPTText* to 0.9 for MathAMR, perhaps because many candidates occur in sentences with uninformative words that AMR ignores, but that raw text includes in the Sentence-BERT input.

2. Smatch Score Results (Table 4.5). As explained in Section 4.4.1, Smatch [25] is a similarity measure to compare MathAMRs by matching triples (edges) with a variable (node) mapping that

Table 4.4: ARQMath-2 Contextual Formula Search results for single sentence embeddings (via Sentence-BERT, assessed formula hits). *w. OPTText*: formulas included as linearized OPTs.

Context Representation	nDCG′@5	nDCG′@10	P′@5	P′@10
AMRText w. OPTText	**0.58**	**0.56**	**0.51**	**0.47**
AMRText w. LaTeX	0.38	0.39	0.35	0.34
RawText w. OPTText	0.54	0.51	0.48	0.43
RawText w. LaTeX	0.43	0.42	0.40	0.37
OPTText	0.35	0.32	0.30	0.24

Table 4.5: Formula Search results re-ranking assessed hits using Smatch score with MathAMR.

Test-Collection	nDCG′@5	nDCG′@10	P′@5	P′@10
ARQMath-1	0.50	0.47	0.47	0.39
ARQMath-2	0.54	0.53	0.49	0.45
ARQMath-3	0.47	0.46	0.45	0.42

maximizes the count of matching triples. Table 4.5 shows formula search results on ARQMath-1, -2, and -3, ranking just the assessed hits, using the Smatch score.

As shown in this table, using Smatch score has lower effectiveness compared to Sentence-BERT in Table 4.4. Looking at ARQMath-3 results, there were 15 topics with P′@10 of 0. There are instances for which the surrounding context is less informative or there is no context, and in such cases formulas are essentially matched in isolation. For example, for topic B.318, with the target formula highlighted:

How do you prove $e^x \geq \left(1 + \frac{x}{n}\right)^n$ for $n \geq 1$

using Smatch score gives higher rank to formulas such as "Prove that $\sum_{k=0}^{n} \frac{1}{k!} \geq \left(1 + \frac{1}{n}\right)^n$" because in the AMRs nodes for 'Prove' and 'you' are connected with same edge labels, and the formulas share similar OPT representations.

While we have only reported ranking with the specific variant of the Smatch score (described in Section 4.1.1), we have also studied other variations of the Smatch score, such as ignoring edge labels, or ignoring PropBank frameset. Focusing on ARQMath-3 test collections, there could be a

Table 4.6: ARQMath-1 and -2 Contextual Formula Search results using MathAMR with single sentence embeddings (via Sentence-BERT), ranking the assessed hits. For each test condition, the other test condition was used for fine-tuning Sentence-BERT.

Test	Train	nDCG′@5	nDCG′@10	P′@5	P′@10
ARQMath-1	ARQMath-2	0.50	0.50	0.45	0.42
ARQMath-2	ARQMath-1	0.54	0.53	0.50	0.45

slight improvement, but not a significant one (by a 2-tailed paired t-test). For example, ignoring the ProbBank frameset can increase P′@10 to 0.03.

Fine-Tuning on Different ARQMath Test Collections

As mentioned in Chapter 3 (Section 3.2.1), despite the same relevance definition, there was a difference in formula search task assessor training, which could lead to different relevance decisions between ARQMath-1 and -2. Therefore, we studied using MathAMR with the Sentence-BERT model, with the same parameters but different training data; fine-tuning on one test collection and testing on the other. Table 4.6 shows the results on ARQMath-1 and -2 test queries. As seen, in this table, the results show stable behavior across training and test sets. All the scores on the ARQMath-1 test set are slightly lower, but such differences are typical in information retrieval evaluation.

Using Different Context Windows

Another parameter to study is what we consider the context window. In our experiments, we selected the sentence in which the formula appeared. However, it is possible that related information to the target formula appears in other contexts than that one sentence. Therefore, we also did the experiment of including a sentence before and after the sentence in which the formula appeared (summing up to 3 sentences) for both queries and for candidates. We use the same process as generating MathAMR with one sentence. However, it should be noted that when generating MathAMR for larger text, the target formula is sometimes dropped as the AMR generator aims to extract a meaning representation. In such cases, we use the MathAMR as it is for search.

For our experiment, we used the same fine-tuning process. ARQMath-1 and additional topics from ARQMath-2 were used to fine-tune Sentence-BERT to get results for ARQMath-2 test topics. For

Table 4.7: Contextual Formula Search results re-ranking assessed hits with MathAMR using context window sizes of 1 and 3.

Test-Collection	**Context Window**	nDCG′@5	nDCG′@10	P′@5	P′@10
ARQMath-2	1 Sentence	**0.58**	**0.56**	**0.51**	**0.47**
ARQMath-2	3 Sentences	0.38	0.38	0.34	0.32
ARQMath-3	1 Sentence	0.43	0.43	0.42	0.39
ARQMath-3	3 Sentences	0.42	0.43	0.40	0.38

ARQMath-3, we used all the topics from ARQMath-1 and -2. We re-ranked the assessed formulas (in the Qrel files) for both test collections. Table 4.7 shows the results. While in ARQMath-3 there is no significant difference between the results, in ARQMath-2 the results were significantly better when using a one-sentence context window. This significant difference in ARQMath-2 may also be an effect of training on fewer samples when we do retrieval on ARQMath-2 test queries.

Considering ARQMath-3, Figure 4.10 plots the differences between P′@10 when using one sentence as context against using three sentences as context. As can be seen, for nearly half the topics in ARQMath-3 using one sentence provided better results, while for the other half, using three sentences, were better. For those that one-sentence context had better effectiveness, the obvious pattern was that one sentence was enough to capture the context of the formula – the related information for the formula has appeared in one sentence in both the query (specifically, query formulas in question titles) and the candidate. For example, topic B.386, for which P′@10 is 0.9 higher when using one-sentence context, is in the question title as: "Prove that $\sum_{n=-\infty}^{\infty} e^{-n^2\pi x} = \frac{1}{\sqrt{x}} \sum_{n=-\infty}^{\infty} e^{-\frac{n^2\pi}{x}}$".

The opposite reasoning can explain topics that have better effectiveness when using 3-sentences as context; related information is found in a larger window. The last pattern that we observed is that when we have a context window of 3 sentences, more formulas can appear in the context (than the formula query itself), and Sentence-BERT can then find formulas similar to those appearing in the context window that are less related to the query.

Re-ranking ARQMath Runs

To further evaluate MathAMR's utility for contextual formula search, we re-rank the best ARQMath-2 and -3 run submitted by each team to the ARQMath-2021 [108] and -2022 [100] shared-task

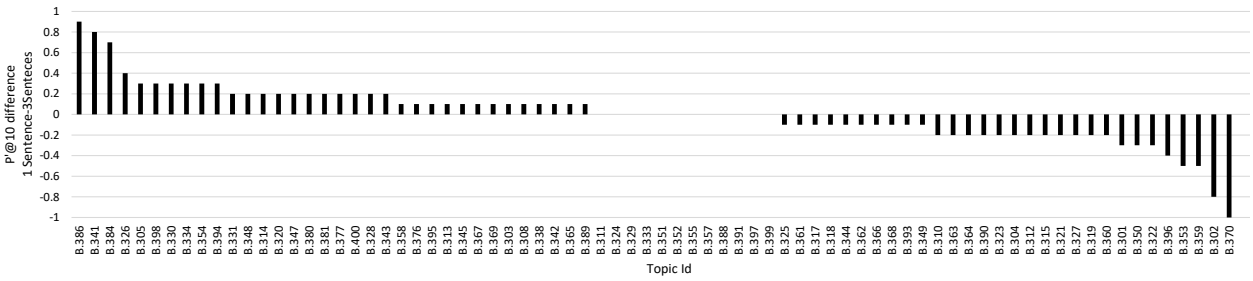


Figure 4.10: Difference between P′@10 using one against three sentences as context for ARQMath-3 topics (P′@10 for one - P′@10 for three sentences).

evaluations, respectively. Table 4.8 shows P′@10 and nDCG′@10 for original runs, those same runs re-ranked using the cosine similarity of MathAMR embeddings *(AMR w. OPTText)*, and combined original ranking scores with MathAMR ranking using our modified Reciprocal Rank Fusion, equation (5.3).

ARQMath-2 runs are from the Approach0 [179], MathDowsers [116], TU-DBS [135], DPRL [103],[11] XY-Phoc [9], and NLP_NITS [33] teams. For ARQMath-3, runs from the Approach0 [178], DPRL [104], MathDowsers [67], XY-Phoc [82], and JU-NIST [140] teams are shown. Only Approach0 was a "manual" run that included human intervention, and only MathDowers used both text and formulas (the others used only formulas).

In ARQMath-2, using linearized MathAMR embeddings to re-rank candidates using cosine similarity sometimes decreases P′@10, partly because there can be several formulas in one sentence, all of which share a single MathAMR similarity score. For this reason, the modified RRF of the original ranking and MathAMR ranking results perform better than re-ranking alone. NDCG′@10 is significantly improved over the original rankings for all systems other than DPRL ($p<0.05$, two-tailed paired t-test with Bonferroni correction). The one case where RRF did not help (DPRL) likely results from the use of tree-edit distance for final matching in that system, since the top results are thus often very similar to the query. However, if we consider all retrieved instances (at $k = 1000$), nDCG′@1000 increases from 0.57 to 0.76 when using RRF to combine DPRL and MathAMR; so even here there is a benefit, it is just seen lower in the ranking.

In ARQMath-3 (similarly to ARQMath-2), using RRF showed better retrieval results compared to re-ranking with cosine similarity using MathAMR, except for the JU-NIST system. That system on its own has a much lower effectiveness than the MathAMR-re-ranked results, and combining

[11] DPRL was our Tangent-CFT2ED run.

Table 4.8: Results for original ARQMath-2 and -3 runs, runs re-ranked by MathAMR, and RRF of Original & Re-rank. *Approach0 is a manual run. †Only MathDowers used text with formulas. In both test collections, DPRL is our Tangent-CFT2ED system.

	nDCG′@10			P′@10		
Model	Original	Re-rank	RRF	Original	Re-rank	RRF
ARQMath-2						
*Approach0	0.59	0.57	**0.63**	0.49	0.48	**0.54**
DPRL	**0.60**	**0.59**	0.62	**0.54**	**0.51**	0.52
†MathDowsers	0.54	0.55	0.58	0.45	0.44	0.48
XY-Phoc	0.51	0.54	0.55	0.43	0.45	0.45
NLP_NIST	0.26	0.33	0.46	0.20	0.26	0.40
TU-DBS	0.25	0.28	0.41	0.22	0.23	0.35
ARQMath-3						
*Approach0	**0.75**	0.56	**0.72**	**0.69**	0.51	**0.67**
DPRL	0.69	**0.59**	0.68	0.61	**0.55**	0.66
†MathDowsers	0.61	0.50	0.62	0.55	0.45	0.57
XY-Phoc	0.64	0.51	0.64	0.56	0.45	0.59
JU_NIST	0.16	0.23	0.22	0.13	0.29	0.17

using RRF does not help. However, the nDCG′@10 had a significant improvement when the results were re-ranked with RRF only for this system ($p<0.05$, two-tailed paired t-test with Bonferroni correction). While for MathDowsers there was a slight improvement in both nDCG′@10 and P′@10 on ARQMath-3, for XY-PHOC, only P′@10 improved (because MathAMR in some topics provided higher rankings for medium and highly relevant formulas).

Looking at the Tangent-CFT2ED (DPRL) model (which was the best automatic run in ARQMath-3), Tangent-CFT2ED has better effectiveness on its own. However, there were cases where combining this model with MathAMR yielded improvements. One obvious pattern is that using MathAMR can help with topics for which variables are important. For example, for topic $F = P \oplus T$. (Topic B.326), P is a projective module and F is a free module. There are instances retrieved in the top-10 results by Tangent-CFT2ED such as $V = A \oplus B$ where variables are referring to different concepts; in the query formula k-dimensional subspaces. With MathAMR, formulas such as $P \oplus Q = F$ appearing in a post that specifically says: "If P is projective, then $P \oplus Q = F$ for some module P and some free module F." (similar text as the topic) are ranked in the top-10 results.

For cases in which Tangent-CFT2ED has better effectiveness, two patterns are observed. First, the formula is specific and variables do not have specifications. Second, the context is not helpful (i.e., no providing any useful information) for retrieval. For instance, in topic B.334, "logarithm proof for $a^{log_a(b)} = b$" the formula on its own is informative enough. Low relevant formulas appearing in a context, such as "When I tried out the proof, the final answer I ended up with was $a^{log_b n}$", are ranked in the top-10 results because of having the word proof and part of the formula.

Combining the results of Tangent-CFT2ED and MathAMR with our modified RRF provided better P′@10 than one of the individual system results for only 10% of the topics. For the topic (B.338) appearing in the title of a question as "Find all integer solutions of equation $y = \frac{a+bx}{b-x}$", both Tangent-CFT2ED and MathAMR had P′@10 of 0.6. However, combining the results with modified RRF increases the P′@10 value to 0.9. Table 4.9 shows the top-10 results for Tangent-CFT2ED+MathAMR, along with the original ranked lists for the Tangent-CFT2ED and MathAMR systems. As can be seen, there are relevant formulas that one of the Tangent-CFT2ED or MathAMR models gave worse ranks to, for which the other system provided a better ranking and combining the systems with our modified RRF improved the results.

Table 4.9: Top-10 Formulas Retrieved by RRF (Tangent-CFT2ED+MathAMR) along with their ranks (smaller is better) in original Tangent-CFT2ED and MathAMR runs for topic (B.338), appearing in a question post title as "Find all integer solutions of equation $y = \frac{a+bx}{b-x}$". For space, the sentences associated with formula hits (which are used by MathAMR) are omitted.

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

4.4.5 Additional Experiments for Sentence-BERT Training and Configuration

After testing our primary MathAMR system design, we investigated our MathAMR model with different designs. Here we explain our additional experiments by using all the topics from ARQMath-1 and -2 for fine-tuning Sentence-BERT and testing our model on ARQMath-3, re-ranking results from the Tangent-CFT2ED system (searched over the whole collection).

MathBERT as a Pre-trained Model. In the experiments reported above, we fine-tuned *all-distilroberta-v1* model that had been pretrained for general text, not specific to math. There could be vocabulary items that are common in math but not in general text. Therefore, we studied using model with math vocabulary. Instead of RoBERTa we used MathBERT (Shen et al., 2021) [146], which uses WordPiece tokenizer [163] with vocabulary size of 30,522. Looking back at our example for tokenization for the sentence "find value of Math sqrt a SUP 4", using the RoBERTa BPE tokenizer the generated tokens are: find, Ġvalue, Ġof, ĠMath, Ġsq, rt, Ġa, ĠSUP, Ġ4, whereas with MathBERT the tokens are: find, value, of, math, sq, ##rt, a, su, ##p, 4, where ## shows that the token is a suffix, following some other subword.

Our experiment results with fine-tuning MathBERT showed a small improvement (0.01) in both

nDCG′@10 and P′@10 when re-ranking Tangent-CFT2ED with cosine similarity of MathAMR embeddings from Sentence-BERT. Note that we kept all the other parameters the same. Looking at the results, only for 29 topics, both models have the same P′@10 values. There are topics such as "... that has the density function $f_X(x) = \frac{2x}{\theta^2}$ for ..." (topic B.396) or "calling the 'Double Basel problem' for the past couple of hours $\sum_{n=1}^{\infty}\sum_{m=1}^{\infty}\frac{1}{n^2+m^2}$." (topic B.324) that have phrases such as "destiny function" and "Double Basel problem" that can appears in different contexts when general text collections are used for pre-training. Perhaps when using a math collection for pre-training, these words would appear in similar contexts more often, leading to a better embedding representation of math terms. For both topics, P′@10 was higher by 0.3 when MathBERT was used.

Special Token for Math. Another model, also named MathBERT (Peng et al., 2021) [127] fine-tunes vanilla BERT without extending the vocabulary. Instead, this model includes a special token (Separator) to distinguish between formula and text. Adopting this idea, we also included two special tokens '$' and '$' at the beginning and end of each formula, respectively. Similar to our previous experiment, we kept the Sentence-BERT parameters the same. Re-ranking Tangent-CFT2ED results with MathAMR Sentence-BERT embeddings (using cosine similarity), the nDCG′ and P′@10 dropped to 0.53 (from 0.59) and 0.49 (from 0.55), respectively.

Extending Vocabulary. Adapting BERT-based models for a new domain can be challenging. There could be specific words that have a different meaning in a specific domain. Also, it is possible that there are some words not existing in the vocabulary set of the tokenizer. As we use both AMR and OPT in our MathAMR, introducing a new vocabulary and fine-tuning a pre-trained model can be helpful. For this, we adopted the idea introduced in exBERT [152] for including new tokens while keeping the existing token embeddings of the original BERT-based model fixed. To find the new tokens that are specific for this domain, exBERT uses the WordPiece algorithm [163]. This algorithm is similar to BPE, except in the merging step. In BPE the most frequent tokens are added to the vocabulary list whereas in WordPiece, the ones that maximize the likelihood of the training data are included. In exBERT model, if a token already exists in the BERT vocabulary, its embedding is initialized randomly and previous embedding is ignored. Embedding for new token are randomly initialized, and optimized during fine-tuning on downstream tasks. ExBERT uses 17.7K new tokens, and their experiments showed that more new tokens gives very little further improvement on downstream tasks.

We have adopted a similar approach. We considered ARQMath collection as our collection, and use MathAMRs generated on the first 128 tokens from both questions and answers to select the most

frequent tokens. As this is a primary attempt, we simply tokenized using white space to find new tokens from MathAMR and then chose the 15K most frequent tokens. Using a similar approach to ExBERT we included these tokens in our vocabulary with random initial embeddings and fine-tuned an `all-distilroberta-v1` model on math question-answer similarity task (ARQMath Task 1). To do this, we considered a question and its selected answer as a positive pair and chose a random answer that was not given to that question to form a negative pair. We have 384K pairs of positive pairs and the same number of negative pairs for fine-tuning. We used batch size 64 with maximum sequence length of 128, fine-tuning for 10 epochs. After fine-tuning with the new set of tokens, we then use this fine-tuned model and did the same fine-tuning we did for the formula search task using linearized MathAMR of topics from ARQMath-1 and -2. We did not change any model parameters, except for the pre-trained model that we used.

Our ranking results showed that on average, there was 0.02 decrease in both nDCG′@10 and P′@10 values when re-ranking with our extended vocabulary model compared to the original RoBERTa. As with MathBERT, we can observe that for certain topics there was improvement in ranking from model fine-tuned with extended vocabulary, whereas for some other topics the ranking results were less effective. One topic for which using original vocabulary was better than the new tokens was topic "... $\int_0^1 \frac{\sin^{-1}(x)}{x}$ ..." (topic B.320 in ARQMath-3) where P′@10 dropped from 0.6 to 0.1. Using extended vocabularies, in the top-10 results for this topic, 8 retrieved formulas have a low relevance. From the results, we see that after fine-tuning on the ARQMath collection, many of the highly ranked formulas contain symbols such as 'sin', '$\int$', and 'π'. The symbol 'π' is not in the topic, although, the formula query is equal to $\frac{\pi}{2} \ln 2$.

For topics on which we saw improvements when using the extended vocabulary, one obvious pattern is that (as with the MathBERT model), there are math-specific phrases for which embeddings pre-trained on general text might not be a good approach. There are phrases such as "cauchy schwartz" (topic B.301), "vector bundle" and "Newlander-Nirenberg theorem" (topic B.304), and "Hilbert space" (Topic B.314) that models with an extended vocabulary probably have provided a better embeddings for. Also, note that with the extended vocabulary model, fine-tuning is done over a math specific collection (ARQMath) in a mono-modal space where both formula and text are represented in the same space and embeddings capture the relation between formula and text.

4.5 Summary

In this chapter, we first introduced the representations used for mathematical formulas. The two common representations that are used in formula search systems are (1) Presentation MathML, capturing the appearance of a formula in a Symbol Layout Tree (SLT), and (2) Content MathML, showing operations in the formula using an Operator Tree (OPT).

Then we reviewed some of the existing formula search models. These models can be categorized into three main groups: text-based, tree-based (full and partial tree matching), and embedding models. Experiment results have shown that text-based models are less effective, as they do not capture the structure of the formula. Tree-based models can find similar formulas by using sub-tree matching, comparing paths in the trees, or by using full-tree matching scores such as tree-edit distance. In embedding models, each formula is represented as a vector. To find similar formulas, vector similarity scores such as cosine similarity used.

After reviewing previously existing systems, we described two categories of formula search systems that we have developed. The first category of systems is the isolated formula search models that consider only formulas for comparison. We developed three systems for isolated formula search. Tangent-CFT is the first embedding model for mathematical formulas that uses both SLT and OPT representations. Since experiment results showed that Tangent-CFT emphasized finding partial matches, we introduced Tangent-CFTED, which re-ranks Tangent-CFT results with a more strict similarity score, tree-edit distance. Finally, we developed a learning-to-rank framework for mathematical formulas that leverages multiple similarity features, including sub-tree, full-tree, and embedding similarity scores.

The second category of systems is contextual formula search models that consider the context of the formula in addition to formula similarity. We have introduced MathAMR, a unified semantic representation for formulas and text. MathAMR integrates Abstract Meaning Representation graphs for text with Operator Tree formula representations. We have made the first study of using MathAMR for formula search, studying the effectiveness of using Sentence-BERT to embed linearized MathAMR trees for single sentences. Compared to representations using raw text, MathAMR achieved better results when ranking all documents for which relevance judgements are available, and using MathAMR embeddings for re-ranking results from other formula search techniques also yielded improvements.

Overall, among the search models we introduced for formula search, our tree-edit distance model, Tangent-CFTED, which re-ranks retrieval results from our embedding model, Tangent-CFT, pro-

vided better effectiveness compared to the other models. This model is the state-of-the-art automatic system on the latest formula search test collection, ARQMath-3. However, our study shows that using context in MathAMR can improve effectiveness for certain topics. Perhaps, analyzing the properties of topics for which MathAMR helped with effectiveness can provide a better means of combining isolated and contextual formula search results.

Chapter 5

Formula+Text Search

Users might use both text and formulas to search for math. They can issue ad-hoc queries with a few keywords and formulas. Another option is using a math question, which was shown to be common in our previous research [106]. Users can specify what information they are looking for related to a formula. For example, they can add terms such as "examples", "solving", and "name of" to a formula to get different information about it. Also, with the growth of community question-answering websites, users might post a math question on these websites, to get an answer to their question.

The attempts at formula+text search include combining individual search results from text and formula searches and searching for both text and formula together using traditional information retrieval or deep neural network models. These types of information retrieval systems are called multi-modal, meaning that both users' queries and the collection can have multi-type data including text, audio, video, pen strokes, and images. While multi-modal information is one related problem to the formula+text search task, another related problem to formula+text search can be image captioning. The goal of this task is to generate a related caption for an image. We can simply map this problem to ours by considering the formula as an image, aiming to generate the text description for the formula. Having a text description of formulas can be beneficial for text-based formula retrieval models.

This chapter introduces the formula+text search task. First, we show how formula+text search can be related to other similar information retrieval and natural language processing tasks. Then, we review some of the existing work on formula+text search and answer retrieval for math questions. Finally, we explain our proposed models for the formula+text search task, focusing on the answer

retrieval task for math questions.

5.1 Related Problems

Multi-modal information retrieval. With the growth of digital content, multi-modal information retrieval systems enable users to search for different modalities such as text, images, audio, and video. As stated in [21], the main challenge in multi-modal (multimedia) retrieval is the gap between the input queries and the collection. This is not an issue in a text search engine as the collection and queries have the same data type, text. Multi-modal search systems consider two main approaches. In the first approach, the retrieval results from individual search systems for each data representation are combined. In the second approach, retrieval is done in a unified framework, with all different types of data being represented in a unified space. These are known as mono-modal search systems, where search is done over a single modality.

An example of the first approach is the multi-modal search method used for semantic patent image retrieval by Pustu-Iren et al. [131]. This work focused on using patent images, which are valuable information sources to compare patents. The previous patent retrieval models before this work only made use of textual information. In this work, two indices are created using textual and image features. For textual information, first a text data is extracted from the image, then a sentence transformer model, RoBERTa [92], is used to extract textual features as a vector. To extract image features, Contrastive Language-Image Pre-training (CLIP) [30] is used, and for each image, a vector representation is created. For a given query, the retrieval is done based on the similarity of textual features, image features, or both. The individual retrieval results are based on the cosine similarity of vector representations. To use both of the features, the individual retrieval results are combined using average or maximum scores.

To provide an example for the second category, in the unified framework introduced by Rafailidis et al. [132], each multimedia document is represented as a vector. A multimedia document contains different modalities such as 3D, images, and audio. First, for each document D_i its x_i^m mono-modal descriptors are extracted, m designating the modality, and a matrix representation is generated based on the number of modalities per document in the collection. Then, with a weighting scheme that captures the importance of each media type. Finally, with Laplacian Eigenmaps using the heat kernel approach, each document is mapped in a vector space, with similar documents being closer to each other.

Formula+Text search is a multi-modal information retrieval task in the sense that formulas can be represented as graphs, images, audio, or penstrokes. The search systems should have the ability to search in different modalities and return relevant documents. Using the first approach, formula and text searches are done separately, and then the retrieval results are combined. The major benefit of this approach is that mature text and formula search can be applied for individual retrieval, and the retrieval results are combined. The main challenge in this approach is to decide how to efficiently combine the search results from text and formula. Specifically, in cases where the input query has multiple formulas in it, a weighting scheme is needed to efficiently combine retrieval results for each formula in the query.

In the second approach, the challenge is defining a unified semantic space for both formulas and text. This might need a new data representation model or a model to map formulas and text to a same space (e.g., using a neural network architecture). In our proposed models, our main focus is leveraging the second approach, with a mono-modal model for search.

Image captioning. Image captioning aims to provide a brief description of an image in a natural language. We choose image captioning over machine translation as a related problem, as in image captioning the modalities of data are different; all the data is not textual. Current methods for tackling this problem combine computer vision and natural language processing techniques [91]. Image captioning approaches can be divided into two groups [55]:

1. **Mono-modal Space.** In these approaches, image and language features are represented differently. Commonly, convolutional neural networks (CNNs) are used to extract image features in the encoder part of the model, and then recurrent neural networks (RNNs) are used in the decoder to generate the caption. The Neural Image Caption (NIC) Generator model [157], for example, uses a CNN for image representations and a Long short-term memory (LSTM) for generating image captions. The NIC uses a novel method for batch normalization, and the output of the last hidden layer is used as an input to the LSTM decoder. The LSTM can track objects that have already been described using text. NIC is trained based on maximum likelihood estimation of generating sequence of words that describe an image.

2. **Multi-modal Space.** In those approaches, there are still two encoders for image and language used as feature extraction pipelines. However, after the encoding, the outputs are mapped to a multi-modal space. The output in the multi-modal space is then fed to the language decoder to produce captions.

   For example, Kiros et al. [71] introduced a model that learns a joint image-sentence embedding. The sentence features are extracted with LSTM and images are represented as the top

layer of a CNN trained for the ImageNet [76] classification task. Then these features are joined in a multi-modal space that maximizes the similarity between image and text embeddings of related pairs (image and its caption), and minimizes similarity to other captions. The decoder is a structure-content neural language model derived from a multiplicative neural language model [72] that models the distribution of a new word for the given context from the previous words and a multi-modal vector space. Also, the decoder uses structural information (in this work, part of speech) to improve generated captions, helping the model to avoid generating grammatical nonsense.

There are several possible ways of using the technique for image captioning for formula+text search. The simplest scenario is to generate the names of the formulas. For example, given the formula $a^2 + b^2 = c^2$, the generated output of the model should be the Pythagorean theorem. However, this can be challenging as there are many formulas that are not linked to a specific math concept. For example, it would be difficult to generate a text for $a + b + c$. Also, there could be different formula representations of the same concept, which may demand a canonicalization method. In an alternative approach, for a given formula as the input, the model can generate its related text. The related text can be a definition, an example, or an application. This can be challenging as there might not be enough data available to train such models.

5.2 Related Work

Bringing the focus back to formula+text search, we review two types of related work. We first review some of the existing search models for ad-hoc queries, and then investigate some of the approaches to answer retrieval for math questions.

5.2.1 Ad-hoc Search Models

The first line of work has the same approach as the ones used for contextual formula search. In models for ad-hoc search, the input query contains both text and formula(s). As with the approaches seen in multi-modal information retrieval, many of the proposed systems combine retrieval results from text and formula search systems. To combine the results, different techniques are used, from a simple averaging of the results to learning to rank models.

1. WikiMirs [48] retrieves mathematical information based on keywords and the structure and

importance of formulas in a document. Using separate tokenizers for formulas and text, the formula tokenizer does normalization and extraction. In the formula index files, the importance of each formula in a document is also recorded. For ranking, this system trained RankBoost [46] using three categories of features: formula-based, relevance-based, and document-based. There are three features used in the formula-based set that capture semantic and structural information about the query. The relevance-based features include five similarity features between the query and a retrieved document. In the document-based features, four features are used from the candidate document.

2. MCAT [75] uses a path-based technique using SLT and OPT representations, as described in the previous chapter (section 4.3.2). Several types of textual information are indexed at three levels of granularity for each formula. At the first level, words in a context window of size 10, descriptions, and noun phrases in the same sentence as the formula are considered. At the second level, all the words in the paragraph in which the formula has appeared are considered. At the third level, the title and abstract of the document, keywords extracted from the document, descriptions of all the formulas, noun phrases, and all the words in the document are considered. (Note that the third level information is extracted based on the nature of the collection. For example, the paragraph-level information is only used for the arXiv collection.)

   A dependency graph of formulas is built, with nodes representing important formulas found in the document. A directed edge between nodes A and B indicates that the string representation of formula A contains expression B. This graph is used to extract additional textual information (from the first level) from similar formulas. For retrieval, all the scores are normalized, and for each field, a cold-start weight is estimated. The final weights for formula and text are learned based on multiple linear regression.

3. MIas [138] uses structural and operator unification in formula search and considers query expansion methods (only for multi-keyword queries) for text search. The formulas are converted within five steps using canonicalization, unification variables, constants, operators, and structure. Weights are assigned to formulas based on the number of unification, with fewer replacements of unification symbols are weighted higher. The authors reported that while unification helped with recall, it hurt precision.

4. Tangent-3 formula search system is similar to Tangent-S [35], using only an SLT representation. Tangent-3 [36] uses the TF-IDF implementation of the Solr text search engine to find relevant documents. For documents with title and body, the score for the title field is doubled, and then the maximum of the title and body scores is considered as the final text

search score. The final relevance score is obtained by averaging similarity scores from text and formula retrieval results.

5. The proposed model by Thanda et al. [154] uses Latent Dirichlet Allocation (LDA) [16] for text search, doc2vec [83] for formula search (as discussed in Chapter 4, section 4.3.2.), and it uses ElasticSearch scoring to find similar patterns. These patterns are based on dependencies and part of speech tags, mapping formulas to their definitions. The retrieval results from the three systems are combined with Borda count-based [7].

5.2.2 Answer Retrieval Models for Math Questions

Math questions are commonly used in math searches. With the growth in community question-answering websites, finding relevant answers to a math question has become more vital. Based on the task introduced in the ARQMath labs, several systems aim to find relevant answers to math questions. These approaches can be categorized into two groups: 1) traditional information retrieval models and 2) embedding models. Overall, knowledge about using embedding models for formula+text is limited. Traditional information retrieval still provides better retrieval results for answer retrieval tasks (based on results in ARQMath-1, -2, and -3). Here, we summarize approaches developed for finding relevant answers to math questions. Note that these systems were developed for ARQMath Task 1, where for a given new math question (a question not in the collection and having similar or duplicate question(s) in the collection), the goal is to find relevant answers from those given to the previous questions. Also note that the traditional information retrieval methods can also be applied for ad-hoc searches, but as these systems were originally designed for answer retrieval task, we describe them here.

Traditional Information Retrieval Methods.

1. Approach0 system [179] is a path-based formula search system that uses OPT representation. For the answer retrieval task, this model is combined with the Anserini toolkit [165]. To improve retrieval results, before combination, two query expansion techniques are used. First, some of the LaTeX representations of formulas are converted to text using a manual table. For example, instead of "π", the term "pi" is used. Also, the proposed approach utilizes the context information from an initial set of retrieved answers using RM3 [1] query expansion. After retrieval, two learning to rank frameworks were introduced using linear regression, and LandaMART [162]. There are three features considered for this: 1) the number of up-votes

for the answer, 2) the similarity of the answer to the topic question,[1] and 3) the number of overlapping tags between the parent question and the topic question.

2. The proposed model from the MathDowsers team [115,116] in ARQMath-1 and ARQMath-2 combines text and formula search results. It achieved the highest effectiveness in answer retrieval task in both years. For formula search, the proposed model relies on the Tangent-L [45] system that is built on the Lucene [14] platform. Tangent-L produces a set of math tuples based on SLT and then each tuple is treated as a single token and BM25$^+$ [93] ranking is used for formula search. The same ranking model is used for the text. However, as the inputs are math questions, there is a conversion step to generate a set of well-formulated query that has formulas and keywords. This tokenization is done by defining a set of rules. For example, if a word contains a hyphen such as "Euler-Totient" it is considered as a single keyword. The retrieval results from text and formula search systems are linearly combined. Then a re-ranking step is applied based on the metadata available for the posts. This includes overlapping tags between the topic question and the question associated with the answer, the answer score (the difference between the positive and negative votes on Math Stack Exchange), and the reputation score of the answer's author.

Embedding-Based Methods.

1. **RoBERTa-based model.** Rohatgi et al. [136], introduced a two-step approach by first selecting candidates with BM25 scoring and TF-IDF with cosine similarity, and then re-ranking the results with RoBERTa [92]. For each answer in the collection, its associated question is also concatenated to it to form a retrieval unit. In the first stage, the top-1000 retrieval results from BM25 and TF-IDF were combined using Reciprocal Rank Fusion [31]. In the second stage, the RoBERTa model fine-tuned on Math Stack Exchange data for a mask prediction task. This system was further extended [137], with formulas being tokenized using byte pair encoding [143]. Also, text was summarized with BERT [37], and noun phrases were extracted as the keywords.

2. **ALBERT-based model.** Reusch et al. [134] made use of ALBERT [80] to find relevant answers for math questions. To use ALBERT, formulas were encoded using byte-pair-encoding [143]. Two approaches were used for text tokenization. In the first approach, posts were split into sentences, whereas in the second approach, chunks of text and formulas were considered, and if the sentence is short and contains a formula, the sentences is concatenated to the formula before it. To fine-tune the model, 1.9M examples were used from

[1] In this research we refer to the questions used in ARQMath answer retrieval task as topic questions.

Math Stack Exchange, with 90% of examples for training and 10% for validation. For each question, one of the answers was randomly chosen as correct, and incorrect answers were chosen randomly from answers given to the other questions that shared at least one tag with the original question.

3. **ColBERT-based model.** In a similar approach to that used for ALBERT, Reusch et al. [134] made use of a ColBERT [69] model to find relevant answers for math questions. For each question, 10 correct and 10 incorrect answers were used to train the model. Each correct answer was paired with all the incorrect ones, generating 100 pairs for each question. If for a question, there are fewer correct answers, the same number of incorrect answers were chosen.

4. **BERT-based model.** CompuBERT model [122] fine-tuned a BERT model using Math Stack Exchange data. For each question, its asker-selected answer is chosen to form a positive pair and a random accepted answer to another question is used to form a negative pair. Formulas are represented as LaTeX strings.

5. **SCM.** The soft cosine measure (SCM) system [121], used a fastText model [17] to provide a vector representation for each post. In this representation, formulas were upper-cased and all the other text tokens were lower-cased to distinguish between them. Then punctuation and numbers were removed. Each answer post was a concatenation of its body, its parent question (both title and body), and parent question tags.

Some systems combine the retrieval results of embedding and traditional retrieval models. For example, Novotny et al. [122], experimented using ensemble models from different embedding and traditional retrieval models (from previous ARQMath runs) using different methods such as unweighted sum, weighted sum, and regression, and with regression they achieved higher nDCG′ than other fusion techniques.[2]

5.3 Answer Retrieval for Math Questions

This section describes our approach to the answer retrieval task. We aim to answer two research questions: (1) is finding similar questions for the input question and then ranking the answers given to those to similar questions effective? and (2) is a mono-modal search model effective for this task?

[2]At the time of writing this dissertation, papers related to ARQMath-2022 systems were not publicly available and are not included in related work

Our first research question explores whether automating the oracle baseline system introduced in ARQMath, Linked Math Stack Exchange Post, is effective. In this baseline, for each test question, the duplicate questions in the ARQMath collection were first listed using links data that were not included in the collection, and the answers given to them were sorted based on the vote scores given by Math Stack Exchange users (describe in Chapter 3, Section 3.2.2). This model had the highest P′@10 value compared to the other runs in the answer retrieval task of ARQMath-2020 and 2021, but not in 2022.

All our proposed models with Sentence-BERT use a two-step retrieval approach [103, 104]. First, similar questions are retrieved. Then, all answers given to similar questions are ranked using different ranking models.[3] We first investigate the use of raw text with formulas represented as LaTeX strings for the mono-modal search model. We then study if MathAMR can be more effective compared to raw text.

5.3.1 Raw Text for Answer Retrieval

Here we introduce our two-step retrieval model using raw text, with formulas represented with LaTeX strings.

STEP 1: Finding Similar Questions

Math Stack Exchange provides links to related and duplicate questions. The related questions have a similar topic,[4] but they are not exactly the same question. The duplicate questions tagged by Math Stack Exchange moderators are defined as a newly posted question that has been asked before on Math Stack Exchange.

In our retrieval models, to first identify similar questions to a topic question, we used a Sentence-BERT with a model pre-trained on the Quora question pairs dataset.[5] The model was trained on over 500,000 sentences, with over 400,000 pairwise annotations indicating whether two questions are duplicates or not. Using this model, we did two-step fine-tuning. First, we fine-tuned the model on both duplicate and related questions. Then another fine-tuning was done, using only the duplicate questions. For our training, we used the posts provided in the ARQMath collection (from 2010 to 2018). In the first fine-tuning, 358,306 pairs were used, and in the second, 57,670 pairs

[3] We also had earlier attempts at the answer retrieval task using FastText-based models, which were less effective. Further information is provided in [102]

[4] Here by topic we mean subject and not topic as defined in information retrieval

[5] `https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs`

were used. In both cases, half of the pairs were positive samples, and the other half were negative samples chosen randomly from the collection.

For training, we used multi-task learning, considering two loss functions: constrastive loss [28] and multiple negatives ranking loss [54]. For query q, with relevant document $d+$ and a set of N non-relevant documents, the constrastive loss function is defined as:

$$L(q, d+) = \frac{exp\left(score\left(q, d+/\tau\right)\right.}{\sum_{i=1}^{N} exp\left(score\left(q, N_i/\tau\right)\right.}$$

where τ is a temperature parameter (in our work the default value $\tau = 1$ is used). The constrastive loss function minimizes the distance between positive pairs and maximizes the distance between negative pairs, making it suitable for classification tasks. The multiple negatives ranking loss function, however, considers only positive pairs, minimizing the distance between positive pairs out of a large set of possible candidates. For the answer retrieval task, this function is defined as:

$$L(q, a, \theta) = -\frac{1}{B}\sum_{i=1}^{B}[S(q_i, a_i) - \log\sum_{j=1}^{B} e^{S(q_i, a_i)}]$$

where θ is the word embedding function with parameters S, B is the batch size, q_i is the question i and a_i is the corresponding answer to q_i. Every other answer (corresponding to another question) is considered a negative candidate. We set the batch size to 64 and the number of training epochs to 20. The maximum sequence size was set to 128.

Figure 5.1(a) shows the Cross-Encoder model pre-trained for finding similar questions. In a Cross-Encoder, two sequences are sent to a BERT model in one pass to output a relevance score. In the first fine-tuning, the question title and body are concatenated, with a maximum sequence length of 265 tokens. In the second fine-tuning, however, we considered the same setting for fine-tuning as the first fine-tuning, with one of three different inputs:

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

To find a similar question, we use each of these three models to separately retrieve the top-1000 most similar questions. The retrieved results are then combined using modified RRF (defined in equation (4.2)). We call this similarity score Question-Question Similarity (**QQSim**).

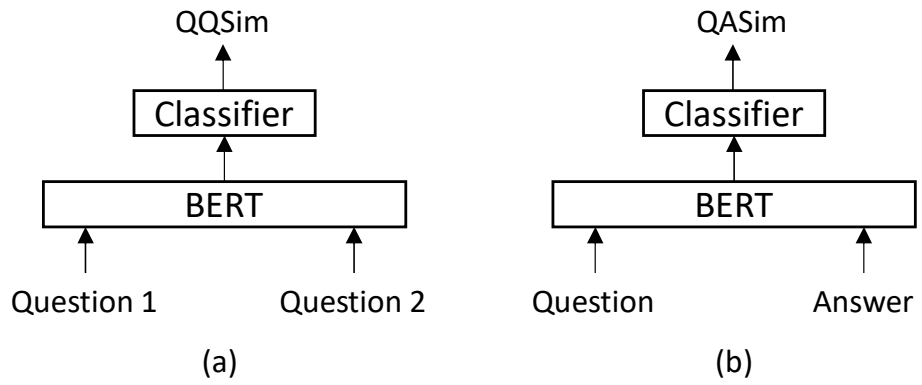


Figure 5.1: (a) Sentence-BERT Cross-Encoder for identifying similar questions and (b) similarity of question and answer. The classifiers give probability of relevance.

STEP 2: Finding Related Answers After the top-1000 similar questions for a topic question are retrieved, all the answers given to similar questions are compiled and then scored with a ranking function. We considered four ranking functions. First, we use the scores given to the answers on Math Stack Exchange for ranking. This means we are using real users' feedback on the relevance of answers to a question. Then, we turn to machines and introduce a second ranking function using a Sentence-BERT model trained to score the relevance of an answer to a question. Our third ranking function combines real-users' and machine scores. Finally, in our last ranking function, we train a rank-SVM model to study the effect of incorporatin additional data from Math Stack Exchange. Here, we describe our ranking functions:

1. **Math Stack Exchange score (MathSE-RawText).** Each post on Math Stack Exchange has a score given by the users. The score is the difference between the positive and negative votes given to that post. In this ranking function, we simply consider this score as an indicator of answer relevance. We used MinMax normalization to map answer Math Stack Exchange scores to values between 0 and 1. Given a set of ranking scores S for a topic question, MinMax normalizes score s to s' as:

$$s' = \frac{s - min(S)}{max(S) - min(S)} \tag{5.1}$$

Note that min and max are computed over the answers to the candidate question. The final similarity score between a question and a candidate answer is calculated as:

$$Relevance(Q_T, A) = QQSim(Q_T, Q_A) \cdot MathSE_{score}(A) \tag{5.2}$$

where the Q_T is the topic question, A is a candidate answer and Q_A is the question to which answer A was given.

Table 5.1: An example of accepted answer for a question similar to ARQMath topic (A.21 in ARQMath-1), assessed as not relevant.

ARQMath Topic Title	Finding the last two digits of $9^{9^{\cdot^{\cdot^{\cdot^{9}}}}}$ (nine 9s)
Similar Question Title	The last two digits of 9^{9^9}
Accepted Answer	At this point, it would seem to me the easiest thing to do is just do 99 mod 100 by hand. The computation should only take a few minutes. In particular, you can compute 93 and then cube that.

2. **Two-step Hierarchical Sentence-BERT (QASim-RawText).** Studies on ARQMath-1 results showed that not all the answers with a high score on Math Stack Exchange are relevant. An example is shown in Table 5.1. In this example, the accepted answer (with the highest score) to a similar question as the topic question in ARQMath-1 is assessed as a non-relevant answer.

   Therefore, we fine-tune a Sentence-BERT model on question and answer pairs from ARQMath answer retrieval task topic questions and their assessed hits. Our pre-trained model is TinyBERT [61] with 6 layers pre-trained on the "MS Marco Passage Reranking" [117] task. The inputs are triplets of (Question, Answer, Relevance), with relevance ranging from 0 to 1. In the ARQMath answer retrieval task evaluation [173], high and medium relevance degrees were considered relevant for precision-based measures. In our training, therefore, we use a relevance score of 1 for answers assessed as high or medium, 0.5 for low, and 0 for non-relevant. This choice was based on a full grid search over [0, 1] with a step size of 0.25 for each relevance degree, and we chose the scores resulting in the highest nDCG′ on the ARQMath-1 topics.

   To train Sentence-BERT we used at most 50 epochs, we chose the best model by training loss on a validation set. We divided the training set into 10 folds, consistently using the same fold for validation. We use a batch size of 64 with a maximum sequence length of 128. We again use multi-task learning with constrastive and multiple negatives ranking loss functions. After training, the cross-encoder outputs the similarity of the question and answer, called **QASim**, as shown in Figure 5.1(b).

   For fine-tuning, the input question is the concatenation of the question title and body. The final ranking score considers two similarity scores; between the topic question and answer, and also between the topic question and the question to which the answer is given (calculated in STEP 1). The ranking function is:

$$Relevance(Q_T, A) = QQSim(Q_T, Q_A) \cdot QASim(Q_T, A) \tag{5.3}$$

where Q_A is the question to which answer A was given.

3. **Combined model with modified Reciprocal Rank Fusion (RRF-MathSE-QASim-RawText).** Our next ranking function combines the similarity scores obtained from the two previous functions using modified Reciprocal Rank Fusion, as given in equation (4.2).

4. **SVM-Rank.** As previous approaches for the answer retrieval task have shown that information such as question tags and votes can be useful in finding relevant answers [116, 179], we aimed to make use of these features and study their effect for retrieval. In this ranking function we considered 6 features: Question-Question similarity (QQSim) score, Question-Answer similarity (QASim) score, number of comments on the answer, answer's score (post score on Math Stack Exchange), a binary field showing if the answer is marked as selected by the asker (to their question), and the percentage of topic question post tags that the question associated with an answer post also contains (which we refer to as question tag *overlap*). Note that we did not apply normalization to feature value ranges. We trained an SVM-rank model [62] using ARQMath-1 and -2 Task 1 topics.[6] After training, we found that QQSim, QASim, and overlap between the tags were the most important features, with weights 0.52, 2.42 and 0.05, respectively, while the weights for other features were less than 0.01.

5.3.2 MathAMR for Answer Retrieval

In our second set of proposed approaches for the answer retrieval task, we use MathAMR as the input for our models. Similar to ranking models with raw text, retrieval with MathAMR is comprised of two stages: identifying candidates from answers to questions similar to a topic question, and then ranking candidate answers by comparing them with the topic question. Here we explain our two steps for retrieval:

STEP 1: Finding Similar Questions In our first step, we find similar questions to a given topic question. For this, we only focus on the question title. Our intuition is that AMR was designed to capture the meaning of a sentence. As the question titles are usually just a sentence, we assume that similar questions can be found by comparing AMR representations of their titles. For each question, we generated MathAMR for its title. Then the MathAMRs are linearized using

[6] For evaluation on ARQMath-2 we only used ARQMath-1 topics.

a depth-first traversal. As a pre-trained model, we used the Sentence-BERT model that we fine-tuned on raw text of the questions' titles. We used the known duplicates from the ARQMath collection (Math Stack Exchange posts from 2010–2018) to train our model on the linearized AMR of questions, using a similar process as for raw text.

STEP 2: Finding Related Answers We considered two ranking functions that similar to our first two proposed approaches with raw text:

1. **Math Stack Exchange score (MathSE-MathAMR).** Using the questions' titles represented as MathAMR, we find the top-1000 similar questions for each topic. Starting from the most similar question and moving down the list, we compile the answers given to the similar questions. The answers to each similar question are ranked based on their Math Stack Exchange score. We used a similar ranking function as in equation (5.2).
2. **Two-step Hierarchical Sentence-BERT with MathAMR (QASim-MathAMR).** This ranking model is similar to QASim with raw text, except instead of using raw text we use linearized AMR. For similarity of questions, we only use the question titles, while for similarity of a question and an answer, we use the first 128 tokens of the linearized MathAMR from the post bodies of the question and the answer. We fine-tuned a Sentence-BERT model, and did retrieval, similar to our QAsim model. We used the ranking function in equation (5.3).

We have also considered another model combining raw text and MathAMR rankings. We combined the results from our SVM-Rank model (from raw text approaches) and QASim-MathAMR (from MathAMR approaches) using modified reciprocal rank fusion (equation (4.2)). We call this model RRF-AMR-SVM.

5.4 Experiment Results

Here we evaluate our proposed models. We first present our results on ARQMath-2 and then explore the ARQMath-3 results. For both test collections, all our model search over the whole collection (and not just the assessed hits). A set of candidates are selected in the first step of our models, and then they are ranked with different functions in the second step. For comparison, we include the best run (with the highest nDCG′) on the ARQMath-2 and ARQMath-3 answer retrieval task from each participating team in ARQMath-2021 and ARQMath-2022, respectively. Note that throughout our analysis we compute P′@10 and mAP′ relevance binarization, considering

high and medium as relevant, and low and non-relevant as answers that are not relevant.

5.4.1 ARQMath-2 Results

Table 5.2 shows the retrieval results on the ARQMath-2 answer retrieval task. The RRF-AMR-SVM model that combines results from QASim-MathAMR and SVM-Rank (using raw text) achieved the highest nDCG′, mAP′ and P′@10. Note that our SVM-Rank model was trained only on ARQMath-1 topics when doing retrieval on ARQMath-2 topics. Except for nDCG′, RRF-AMR-SVM model significantly outperformed the runs from all participating teams in ARQMath-2021 ($p < 0.05$, two-tailed paired t-test with Bonferroni correction). The nDCG′ difference was significant for all systems except MathDowers. Among our proposed models, RRF-AMR-SVM has a significantly better effectiveness than all of our other models except for SVM-Rank considering all three measures. Our MathAMR models are using the Sentence-BERT model that are pre-trained on raw text not AMR token sequence, and this might explain why using raw text provides better results compared to MathAMR in isolation.

Comparing Answer Retrieval Models by Topic Attributes. ARQMath provides different types of topic questions. They are categorized based on their difficulty into hard, easy and medium. Another grouping is based on whether the question is expected to be dependent on the text, formulas, or both. The last category divides the questions based on their subject into concept, computation or proof. We separate topic questions based on these categories and calculate P′@10 for each group. The results are shown in Table 5.3.

As shown in this table, SVM-Rank and RRF-AMR-SVM provided better P′@10 for all the categories. Looking at the difficulty levels, for hard topics, combining results from MathAMR and raw text (RRF-AMR-SVM) provides better retrieval results. For topics A.255 and A.258 in ARQMath-2, that are both categorized as hard, P′@10 increases by 0.4 when SVM-Rank results are combined with QASim-MathAMR (RRF-AMR-SVM). It should be noted that both topics are dependent on formulas. Looking at the dependencies, as expected combining raw text with MathAMR that considers OPT representation of formulas, provided better retrieval results for the topics dependent on formulas (P′@10 for RRF-AMR-SVM on formula-dependent category is 0.648). Similarly, for the topics dependent on text, using a raw text representation had a better P′@10, with a value 0.1 higher on average. For topic A.210, the P′@10 increases from 0 to 0.5 when SVM-Rank is combined with QASim-MathAMR. The title of this question is 'what's an elegant way to show that $x(1-x) \leq \frac{1}{4}$?' and when re-ranked with RRF-AMR-SVM, relevant answers such as 'Why don't you phrase the question by asking why $x(1-x) \leq \frac{1}{4}$ ...' are in the top-10 results.

Table 5.2: Answer retrieval results on ARQMath-2 (71 test topics). Our models are trained on ARQMath-1 answer retrieval task topics. Approach0 is a manual run.

Team Name (Run Name)	nDCG′	MAP′	P′@10
Baseline (Linked MSE Posts)	0.203	0.120	0.282
MathDowsers (primary)	0.434	0.169	0.211
TU_DBS (TU_DBS_P)	0.377	0.158	0.227
Approach0 (B60)	0.351	0.137	0.189
MIRMU (WIBC)	0.332	0.087	0.106
MSM (MG)	0.278	0.077	0.127
PSU (PSU)	0.242	0.065	0.110
GoogolFuel (2020S41R71)	0.203	0.050	0.092
BetterThanG (Combiner1vs1)	0.157	0.031	0.051
RawText			
SVM-Rank	0.433	0.342	0.504
QASim-RawText	0.388	0.147	0.193
RRF-MathSE-QASim-RawText	0.347	0.101	0.132
MathSE-RawText	0.323	0.083	0.078
MathAMR			
RRF-AMR-SVM	**0.473**	**0.348**	**0.507**
MathSE-MathAMR	0.200	0.069	0.122
QASim-MathAMR	0.144	0.050	0.110

Table 5.3: P′@10 values for our models on different categories of topic questions in ARQMath-2 answer retrieval task with 71 topics.

	Difficulty			Dependence			Subject		
	Hard	Medium	Easy	Formula	Text	Both	Computation	Concept	Proof
Topic Count (ARQMath-2):	19	20	32	21	10	40	25	19	27
RawText									
MathSE-RawText	0.053	0.050	0.109	0.076	0.120	0.068	0.056	0.084	0.093
QASim-RawText	0.184	0.115	0.247	0.300	0.130	0.153	0.204	0.116	0.237
RRF-MathSE-QASim-RawText	0.105	0.065	0.191	0.181	0.140	0.105	0.132	0.100	0.156
SVM-Rank	0.347	**0.495**	**0.603**	0.586	**0.620**	**0.433**	**0.548**	**0.426**	0.519
MathAMR									
MathSE-MathAMR	0.150	0.053	0.148	0.171	0.050	0.114	0.200	0.053	0.100
QASim-MathAMR	0.119	0.067	0.135	0.153	0.089	0.091	0.124	0.044	0.152
RRF-AMR-SVM	**0.390**	0.480	0.594	**0.648**	0.520	0.430	**0.548**	0.411	**0.537**

Comparing our MathSE-RawText and MathSE-MathAMR models, for text-dependent topics, MathSE-

Table 5.4: P′@10 for ARQMath-2 Answer Retrieval task topics that are dependent on formulas, for which using *MathSE* with MathAMR had better effectiveness than raw text.

Id	Title	MathAMR	RawText
A.209	Evaluate the definite integral: $\int_0^\infty e^{-hx^2}\,\mathrm{d}x$	0.9	0.0
A.255	Integral of $e^{-\frac{u^2}{2}}$	0.6	0.1
A.205	How can we find x for $x^n = n^x$	0.5	0.0
A.210	What's an elegant way to show that $x(1-x) \leq \frac{1}{4}$?	0.3	0.0
A.279	If $\lim_{x\to 0}\left(f(x) + \frac{1}{f(x)}\right) = 2$, show that $\lim_{x\to 0} f(x) = 1$	0.2	0.0

RawText had a higher P′@10, whereas for formula-dependent topics using MathSE-MathAMR provided better results. Table 5.4 shows the P′@10 values for 5 topics (with their titles) that the MathSE-MathAMR model had better effectiveness compared to MathSE-RawText. As can be seen, these topics have a direct question in the title, so perhaps our choice of focusing on the title for MathAMR when finding similar questions was a reasonable approach. Also, using MathAMR did better for hard and computation topics, but our analysis also shows that topics that MathSE-MathAMR provided better results for topics dependent formulas than MathSE-RawText.

Looking at our QASim model, in contrast to MathSE, QASim-RawText model works better compared to QASim-MathAMR, for all categories of topics. Specially, for the formula-dependent category, the P′@10 is nearly twice as large when raw text representation is used compared to MathAMR, which contrasts to our observation with the MathSE models. One issue that can be observed with our current design for QASim-MathAMR is that we just use the question body and ignore the title. Our idea was to use text with similar field types (post bodies) for comparison. However, there are topics for which the main question is asked in the title and the body is providing more explanation. Note that in our QASim-RawText model, we used concatenation of the title and body of the topic for search. One example that QASim-RawText had a better P′@10 was topic A.209. The body of this topic is:

"where $h > 0$. Could someone explain to me how to solve it? I searched the internet and I found the result is $\frac{\sqrt{\pi}}{2\sqrt{h}}$ but I couldn't undersand Gauss error function - that is involved in solving."

which continues providing more information about the main question, which is written in the title as "Evaluate the definite integral: $\int_0^\infty e^{-hx^2}\,\mathrm{d}x$".

Table 5.5: Answer retrieval results on ARQMath-3 (78 topics). Our models are trained on ARQMath-1 and -2 answer retrieval task topics. Approach0 is a manual run.

Team Name (Run Name)	nDCG′	MAP′	P′@10
Baseline (TF-IDF (Terrier))	0.272	0.064	0.124
Baseline (Linked MSE Posts)	0.106	0.051	0.168
approach0 (fusion_alpha05)	**0.508**	**0.216**	**0.345**
MSM (Ensemble_RRF)	0.504	0.157	0.241
MIRMU (MiniML+Roberta)	0.498	0.184	0.267
MathDowsers (L8_a018)	0.474	0.164	0.247
TU_DBS (math_10)	0.436	0.158	0.263
SCM (interpolated_text+positional_word)	0.257	0.060	0.119
RawText			
SVM-Rank	0.324	0.112	0.222
MathSE-RawText	0.313	0.087	0.147
RRF-MathSE-QASim-RawText	0.277	0.090	0.195
QASim-RawText	0.255	0.083	0.194
MathAMR			
RRF-AMR-SVM	0.318	0.097	0.168
MathSE-MathAMR	0.203	0.046	0.118
QASim-MathAMR	0.187	0.041	0.103

5.4.2 ARQMath-3 Results

For ARQMath-3, all our model are the same as ARQMath-2, except for the training data used. We trained our SVM-Rank and QAsim models for both raw text and MathAMR using all the assessed topics in ARQMath-1 and -2. Table 5.5 shows the results of our models on ARQMath-3, along with the best runs from each participating team in ARQMath-2022. As can be seen, our proposed models are less effective on ARQMath-3 compared to other runs that participated in ARQMath-2022. As explained, our models are developed based on the idea of automating the baseline system, Linked MSE posts. Comparing the ARQMath-2 and -3 results for the Linked MSE posts baseline system, almost in all the three measures, the effectiveness drops. This observation might help explain why our models' effectiveness also drop in ARQMath-3 compared to ARQMath-2. However, we also observe our SVM-Rank model using only raw text, achieves the highest nDCG′ and mAP′ and P′@10. In ARQMath-2, RRF-AMR-SVM was the best model with all three effectiveness measures.

There are 10 topics in ARQMath-3, with an average of 20.1 answers assessed as high or medium for which none of our models can retrieve any high or medium relevant answers in the top-10

results (compared to 37.7 on average for the 78 assessed topics), leading to P′@10 of 0 on those 10 topics. Looking at the results for those topics, one obvious reason for low effectiveness is the candidate answer selection process, where answers to questions that might not be relevant to the topic are selected as candidates. For example, for topic A.305 with title 'What will be the value of floor function of $\lim_{N\to\infty}\left\lfloor\sum_{r=1}^{N}\frac{1}{2^r}\right\rfloor$' question with the title 'Show that $\lim_{n\to\infty}\frac{1}{n}\sum_{k=1}^{\infty}\left\lfloor\frac{n}{3^k}\right\rfloor=\frac{1}{2}$' is detected as similar, but answers given to that question not provide useful information for the original topic. Another example is topic A.314, with title 'Closed span of a sequence in Hilbert spaces.' where the most similar question detected for this topic is the question with title 'Closure of the span in a Banach space'. Our model failed to detect the importance of *Hilbert* vs *Banach* space. Answers given to this question are assessed as non-relevant. There are also topics such as A.308 with title 'Riemann's definition of the zeta function' for which our models detect the question with title 'What is the Riemann-Zeta function?' as a similar one, but answers given to that question are assessed as non-relevant in ARQMath-3. Finally, there are topics such as A.330, with the less informative title 'Shilov's Linear Algebra[7] - Chapter 1, Problem 9'. All our models make use of topic-question similarity to find similar questions, and questions with titles such a 'Shilov Linear Algebra, Coordinate transformations' and 'Georgi E Shilov Linear Algebra P44' are selected in the first-stage of retrieval, where the original questions are different from the topic. This becomes concerning for our MathAMR models that in the first stage (finding similar questions) only use question titles. In addition, in ARQMath-3, a new category was introduced for topics that determine whether a topic has a multiple questions inside it. 7 out of the 78 assessed topics had multi-parts and for 6 these parts are explained inside the question's body and not the title.

As in ARQMath-2, our raw text models in isolation provide better retrieval results than using MathAMR in isolation. All our models rely on finding similar questions in the first stage, but for raw text we consider three parts of the questions: the title, the beginning and the ending, while for MathAMR only the title is considered. This more limited input information for MathAMR decreases effectiveness in topics such as A.382, where the title is 'Set of functions from $SS = \{A_1, A_2, A_3, ...\}$ to $\{\mathbf{T}, \mathbf{F}\}$ is countable?'. For this topic, P′@10 is 0.5 for MathSE-RawText and 0 for MathSE-MathAMR. With MathAMR, questions having titles that include terms such as 'function', 'set' and 'countable' (or 'counting') were considered as questions with high similarity, but answers given to them are assessed as low or non-relevant in ARQMath-3. For example, answers given to the question with title "Counting functions between two sets" are assessed as non-relevant, as the question body has a different content compared to question title.

Comparing Answer Retrieval Models by Topic Attributes. Looking at different categories

[7]This is the name of a linear algebra book

Table 5.6: P′@10 values for our models on different categories of topic questions in ARQMath-3 answer retrieval task with 78 topics.

	Difficulty			Dependence			Subject		
	Hard	Medium	Easy	Formula	Text	Both	Computation	Concept	Proof
Topic Count (ARQMath-3):	18	43	17	22	10	46	21	16	41
RawText									
MathSE-RawText	0.094	0.147	0.206	0.182	0.120	0.137	0.157	0.156	0.139
QASim-RawText	0.139	0.174	0.300	0.246	**0.240**	0.159	0.229	0.225	0.163
RRF-MathSE-QASim-RawText	**0.144**	0.165	0.324	0.255	0.200	0.165	**0.243**	0.219	0.161
SVM-Rank	**0.144**	**0.200**	**0.359**	**0.314**	0.170	**0.189**	0.195	**0.294**	**0.207**
MathAMR									
MathSE-MathAMR	0.106	0.105	0.165	0.109	0.140	0.117	0.119	0.138	0.110
QASim-MathAMR	0.111	0.086	0.135	0.109	0.090	0.102	0.105	0.144	0.085
RRF-AMR-SVM	0.128	0.158	0.235	0.232	0.150	0.141	0.133	0.244	0.156

of topics, Table 5.6 show the P′@10 values of our models for each category of ARQMath-3. Except text-dependent and computation-related topics, the SVM-Rank model has the highest P′@10. MathSE and QASim have better P′@10 when raw text representation is used instead of MathAMR. Only for hard topics, and topics dependent on text, the MathAMR-MathSE model has an even slightly higher P′@10 compared to raw text.

There are cases where MathAMR can be more effective due to its incorporating OPT representations. For example, for topic A.328, with the title:

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

Table 5.7 shows the titles of the top-5 similar questions detected in the first stage of our models using MathAMR and raw text. As seen in this table, MathAMR representations retrieved two similar questions (at ranks 3 and 4) that have similar formulas, whereas raw text failed to retrieve those formulas in its top-5 results. The P′@10 for the QASim-MathAMR was 0.5, whereas with QASim-RawText it was 0.1.

5.5 Summary

This chapter explored techniques for formula+text search. This problem is similar in some ways to multi-modal retrieval and image captioning. The techniques used for both problems can be divided into two groups: mono-modal and multi-modal approaches. In mono-modal approaches, each content type is represented in its own feature space, whereas in multi-modal approaches,

Table 5.7: Titles of the top-5 most similar questions found with MathAMR and raw text, for the topic question with title *"Proving* $\sum_{k=1}^{n} \cos \frac{2\pi k}{n} = 0$*"*.

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

different types of content are encoded in one unified feature space.

For formula+text search, two tasks are defined in previous test collections. One is ad-hoc search with queries containing both keywords and formula(s), and the other is answer retrieval for math questions. The current approaches for ad-hoc search make use of formula search systems and traditional information retrieval models such as TF-IDF or BM25, and they combine the search results. The combination is done in different ways, from simple averaging to learning to rank. For the answer retrieval task, traditional information retrieval models used for ad-hoc search provide better results compared to the deep neural network approaches.

We introduced two new sets of approaches for the answer retrieval task. Both sets of approaches are similar in that they both first find similar questions and then use different ranking approaches. In our first set of proposed approaches, we used raw text with Sentence-BERT embedding. We trained Sentence-BERT for question-question and question-answer similarity. We also used SVM-Rank, considering additional features such as answer scores for ranking. Our second set of approaches also uses Sentence-BERT, but on linearized MathAMRs. MathAMR is a mono-model representation that uses a single unified representation for formulas and text. Our research showed that our approaches can achieve the state-of-the-art results in ARQMath-2, but are less effective compared to state-of-the-art systems in ARQMath-3, analysis of those results is still ongoing. However, MathAMR and raw text representations can be effective for different types of questions. For example, for questions depending on formula, such as 'Evaluate the definite integral: $\int_0^\infty e^{-hx^2}\, \mathrm{d}x$', our models achieved a high effectiveness (P′@10 of 0.9). Therefore, combining results based on topic categories might be promising. We have also shown how using OPT in our MathAMR models can be beneficial for answer retrieval task. As our models with raw text had a better effectiveness, perhaps incorporating OPT representation with raw text can be beneficial.

Chapter 6

Conclusion

This dissertation has addressed three broad aspects of a math-aware search research.

- **Query Log Analysis**: We started our work with analyzing users' search behavior that helped with design decisions of both search models and test collections. We have seen how users' behavior can be different when it comes to a specific domain such as math compared to general searches. Perhaps future work on specific domains would benefit from doing such analysis before designing systems and test collections.
- **Test collections**: We have introduced the ARQMath test collections for math-aware search. In addition to its direct support for evaluation of math-aware search, the CLEF ARQMath lab has brought attention to math information retrieval tasks, and brought a community of researchers together.
- **Search Models**: We have introduced models that consider isolated formulas represented in tree format. Models such as Tangent-CFT have brought attention to use of embedding models for formula search. For example, MathBERT [127] uses a similar linearization approach as Tangent-CFT. For contextual formula search, we have introduced a new unified single representation for text and formulas that we believe has potential to be used for both math information retrieval and natural language processing tasks such as summarization and math entity linking.

This dissertation aims to provide analysis on math searches, test collections for math-aware search, and leverages formula and text context for math information retrieval. Using formulas and text for

search can be challenging, and we address this challenge in this dissertation. We have addressed these research questions in this dissertation:

- **Question:** How do users utilize existing general-purpose search engines for math-aware search, and what types of information needs can we observe in their query logs?

  Answer: Our study on general search engine query logs revealed several properties of these searches. For example, nearly 20% of math queries are questions, much higher than the 2% in general searches. This points to the importance of question answering systems for math. (Chapter 2)

- **Question:** How should we (1) define relevance for contextual formula search task and (2) evaluate contextual formula search systems?

  Answer: Developing the ARQMath test collections, we defined contextual formula search as the task of finding formulas that are associated with the information helping searchers achieve their search goal. In the ARQMath, we consider this goal as understanding the math question and answering it. We introduced an improved evaluation protocol based on visual clustering of formulas by considering only one instance of retrieved distinct formulas. (Chapter 3)

- **Question:** Can an embedding model (any model to map formulas to points in space) be used beneficially for the formula search task?

  Answer: Our first model for formula search, Tangent-CFT, is an n-gram embedding model based on appearance and syntax representations of formulas. Experiments showed this model can be good for approximate matching, but some full matches were ranked lower. Therefore, we introduced Tangent-CFTED, which re-ranks Tangent-CFT results with tree-edit distance (a stricter matching criterion). Tangent-CFTED is the current state-of-the-art for the contextual formula search task. Finally, we introduced a learning to rank framework that leverages different similarity scores, including sub-tree, full-tree and embedding similarities. (Chapter 4)

- **Question:** Can a unified single representation of text and formulas be beneficial for contextual formula retrieval?

  Answer: We have introduced MathAMR to represent both text and formulas in one monomodal space. We then used the Sentence-BERT model on linearized MathAMR for formula search. Our experiment results showed combining isolated, and contextual formula search models can achieve promising results for the contextual formula search task. (Chapter 4)

- **Question:** Can first finding similar questions and then ranking answers given to them be effective for the answer retrieval task?

  Answer: We used a Sentence-BERT model for finding similar questions and then ranking answers given to them. For re-ranking, we fine-tuned Sentence-BERT on tuples of (question, relevant answer and not-relevant answer). While our current models are effective for some math questions, results averaged over a broad range of math questions still have room for improvement. (Chapter 5)

- **Question:** Can answer retrieval be performed effectively using a unified representation for text and formulas?

  Answer: We used both a raw text and MathAMR representations as two unified single representation of formulas and text. By unified representation in raw text, we mean both text and formulas are represented as a sequence of tokens as the input for a search model. In MathAMR, both text and formulas are unified in tree representation. We used fine-tuned Sentence-BERT models with two-step retrieval model. Our approach for some categories of math questions can provide effective results, whereas for other math questions the results are less effective.

6.1 Limitations

This section identifies some of the key limitations of our research.

1. **Limited Training Data.** In this dissertation, we have introduced the ARQMath test collections, with a far larger number of topics than any existing math information retrieval test collections. These test collections are more than adequate for evaluation, but as training data, there are far smaller than the MS-MARCO [117] test collection that has been used to train text retrieval systems.

2. **AMR Parser for Math.** One of the limitations of our current work is not having an AMR parser trained specifically for math. We used a parser that was trained on general text. With an AMR parser trained on math text, it is possible, that as has been seen in the biomedical domain [49], we could have a better representation of the text. That might not be the case in real-world applications.

3. **Test Collection Design.** With our current design of ARQMath, we cannot tell how well systems can help users determine there is an answer in the collection for their question.

In addition to this, systems are studied based on the relevance of the answers, not their correctness. Finally, Task 1 answer retrieval systems do not get rewarded for diversity in retrieved answers. Same information can be repeated in different answers and for each such information a system gets rewarded. In a real-world scenario, providing diversity in search results could be important. For example, when a user issues a question, it could be beneficial to see different information related to definitions, applications, and examples.

4. **Answer Retrieval Models.** Our answer retrieval models are designed for the answer retrieval task in ARQMath, where each topic question had at least one duplicate (nearly exact) question in the collection. However, this might not be the case in a real-world problem. In Math Stack Exchange, there are a lot of questions that have no duplicate question. We have not explored our models' effectiveness for those questions.

5. **Collections Used for Search.** To study our models, we used the ARQMath test collections, where formulas are easy to find. Formulas are located in specific HTML tags with a unique identifier. Also, formulas are easily parsed, with different representations such as Presentation and Content MathML available for each formula. In our work, we also did not study error propagation from parsing and detecting formulas in a collection to be searched.

6. **Efficiency.** In real searches, system efficiency can be as important as effectiveness. Our limited focus on both measuring efficiency and on optimizing for it is a limitation of our work.

6.2 Future work

This research has opened several potential lines of future work, including:

1. **Understanding of Math Information Needs.** We believe there is still a lot left to understand about how users express their mathematical information needs, and search models can benefit from understanding their behavior. As we write this dissertation, there is yet no research analyzing users' behavior on a math-aware search engine.

2. **Answer Retrieval Models.** We explored both isolated and contextual formula search models in this dissertation. Our proposed model, with MathAMR, considers all the formulas in the context equally. For future work, we would like to explore more with models that leverage both isolated and contextual formula search together.

3. **Intrinsic Study of MathAMR.** In our work, we studied MathAMR for two search tasks: contextual formula search and math answer retrieval. As a future work, intrinsic evaluation of MathAMR embeddings can provide better insight about their effectiveness.

4. **BERT-based Models Trained on Math.** As we were reaching the end of this dissertation, there were a few attempts to train BERT-based models from scratch on math, such as MathBERT [127]. Leveraging those models can might show better results. Also, our current MathAMR models use a Sentence-BERT pre-trained on natural language. Re-training BERT-based models on MathAMR strings might improve search results.

5. **MathAMR for Math-aware Search.** In future work, we would like to explore using MathAMR with models such Poincaré embeddings [118] that are designed for hierarchical data, avoiding the need for our current linearization.

   Another potential direction is using AMR parsers that support parsing multiple sentences. The models that we have used are designed for handling one sentence. Parsers that are designed for summarization [89] could be useful for parsing passages, for example.

   Our experiment results on contextual formula search show that different context windows can affect effectiveness differently for different topics. As a future work, an automatic method might be used to decide on a context window size for each formula.

   Finally, we have only considered one representation of formulas in our MathAMR design, operator trees. Previous research on math-aware search has shown that combining different formula representations, such as operator trees and symbol layout trees, can provide better effectiveness [35, 107]. Integrating symbol layout trees in MathAMR may be worth investigating.

6. **MathAMR for Math Entity-Linking** Our MathAMR representation could also be useful for tasks other than search. For example, we believe linking formulas to their concept name is a task that MathAMR could be useful for, since in many cases a formula and their concept are connected to the same parent node in our MathAMR representation as words used to name the concept that the formula represents.

7. **AMR Search Models for other Domains.** We have shown how using AMR can sometimes be effective for integrating formulas and text together as a unified representation for search. For future work, other domains that also deal with graphs and text might adopt a similar approach. For instance, in chemistry a chemical compound can be represented as a graph and the text and graph representations might be unified into a useful representation as a basis for searching documents containing chemical notations.

8. **Open Domain Question Answering.** We introduced the ARQMath test collections with two main tasks, answer retrieval for math questions, and formula search. In ARQMath-3 we introduced a pilot task, open-domain question answering, in which answers might be automatically generated rather than found. Work on this task should surely continue.

9. **ARQMath MSE Collection.** The ARQMath Math Stack Exchange (MSE) collection itself is a rich collection with a large number of annotated formulas. We believe this collection can also be used for other tasks, such as summarization of mathematical texts and text/formula co-reference [110].

10. **Test Collections for Other Domains.** With some of the code we have created for developing this test collection, we believe other domains can also adopt our design for generating task-specific test collections. Stack Exchange includes focused CQA site for a wide variety of topics such as: Law, Physics, Sports, History and Chemistry.

6.3 Broader Impact

Our research had two main elements: math and search. Almost everyone uses math. We calculate and measure many things in our daily activities, such as planning a path and spending money. Several jobs make extreme use of with math; engineers, mechanics, scientists, and artists all rely on aspects of math. Searching is also a part of our daily routine. We turn to search systems to get simple information, such as weather conditions, or very technical information. Putting these two facts together supports the potential of math-aware search.

We started by studying how users search for math, and have ended by studying systems that can better support those search activities. Several questions have been answered in this research, and several new questions have been identified. That said, it is clear that the state of the art has now reached a point where practical systems could now be deployed in operational settings. With all our work publicly available, we look forward to what more will be learned about our proposed techniques through actual practice.

We focused on developing math-aware search systems, but search is not the end goal. Engineers will want to use our techniques to improve access to engineering literature. Teachers and students could use our methods to develop a better learning environment. In learning activities, students can use our methods to find the references for assignments, to help them solve problems, and clarify concepts. Our goal was to enable new capabilities, and we look forward to seeing the broad rage

of human accomplishment that our advances in the state of the art will help to foster.

Bibliography

[1] Nasreen Abdul-Jaleel, James Allan, W Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark D Smucker, and Courtney Wade. UMass at TREC 2004: Novelty and HARD. *Computer Science Department Faculty Publication Series*, 2004.

[2] Saleem Ahmed, Kenny Davila, Srirangaraj Setlur, and Venu Govindaraju. Equation Attention Relationship Network (EARN) : A Geometric Deep Metric Framework for Learning Similar Math Expression Embedding. In *25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021.

[3] Akiko Aizawa, Michael Kohlhase, and Iadh Ounis. NTCIR-10 Math Pilot Task Overview. In *NTCIR*, 2013.

[4] Akiko Aizawa, Michael Kohlhase, and Iadh Ounis. NTCIR-11 Math-2 Task Overview. In *NTCIR*, 2014.

[5] Maria Alexeeva, Rebecca Sharp, Marco A Valenzuela-Escárcega, Jennifer Kadowaki, Adarsh Pyarelal, and Clayton Morrison. MathAlign: Linking Formula Identifiers to their Contextual Natural Language Descriptions. In *Proceedings of The 12th Language Resources and Evaluation Conference*, 2020.

[6] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A Simple but Tough-to-Beat Baseline for Sentence Embeddings. In *5th International Conference on Learning Representations*, 2017.

[7] Javed A Aslam and Mark Montague. Models for Metasearch. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2001.

[8] Ron Ausbrooks. Mathematical Markup Language (MathML) Version 2.0. 1998.

[9] Robin Avenoso, Behrooz Mansouri, and Richard Zanibbi. XY-PHOC Symbol Location Embeddings for Math Formula Retrieval and Autocompletion. In *CLEF (Working Notes)*, 2021.

[10] Peter Bailey, Ryen W White, Han Liu, and Giridhar Kumaran. Mining Historic Query Trails to Label Long and Rare Search Engine Queries. In *ACM Transactions on the Web (TWEB)*, 2010.

[11] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract Meaning Representation for Sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability With Discourse*, 2013.

[12] Michael Bendersky and W Bruce Croft. Analysis of Long Queries in a Large Scale Search Log. In *Proceedings of the Workshop on Web Search Click Data*, 2009.

[13] Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. One SPRING to Rule Them Both: Symmetric AMR Semantic Parsing and Generation Without a Complex Pipeline. In *Proceedings of AAAI*, 2021.

[14] Andrzej Białecki, Robert Muir, Grant Ingersoll, and Lucid Imagination. Apache Lucene 4. In *SIGIR Workshop on Open Source Information Retrieval*, 2012.

[15] Barry Biletch, Kathleen Kay, and Hongji Yu. *An Analysis of Mathematical Notations: For Better or For Worse*. Worchester Polytechnic Institute, 2015.

[16] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation. *The Journal of Machine Learning Research*, 2003.

[17] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 2017.

[18] Claire Bonial, Julia Bonn, Kathryn Conger, Jena D Hwang, and Martha Palmer. Propbank: Semantics of New Predicate Types. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, 2014.

[19] Claire Bonial, Stephanie Lukin, David Doughty, Steven Hill, and Clare Voss. InfoForager: Leveraging Semantic Search with AMR for COVID-19 Research. In *Proceedings of the Second International Workshop on Designing Meaning Representations*, 2020.

[20] Pia Borlund. The IIR Evaluation Model: a Framework for Evaluation of Interactive Information Retrieval Systems. *Information Research*, 2003.

[21] Alessandro Bozzon and Piero Fraternali. Multimedia and Multimodal Information Retrieval. In *Search Computing*. Springer, 2010.

[22] Andrei Broder. A Taxonomy of Web Search. In *ACM SIGIR Forum*, 2002.

[23] Chris Buckley and Ellen M Voorhees. Retrieval Evaluation with Incomplete Information. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2004.

[24] Stephen Buswell, Olga Caprotti, David P Carlisle, Michael C Dewar, Marc Gaetano, and Michael Kohlhase. The OpenMath Standard. Technical report, Version 2.0. Technical report, The Open Math Society, 2004.

[25] Shu Cai and Kevin Knight. Smatch: An Evaluation Metric for Semantic Feature Structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013.

[26] Florian Cajori. *A History of Mathematical Notations*. Dover Publications, 1993.

[27] Jaime G Carbonell and Jade Goldstein. The Use of MMR and Diversity-Based Reranking for Reodering Documents and Producing Summaries.(1998). 1998.

[28] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a Similarity Metric Discriminatively, with Application to Face Verification. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. IEEE, 2005.

[29] DeWitt Clinton. OpenSearch 1.1 Specification (draft 4). *Opensearch.org*, 2007.

[30] Marcos V Conde and Kerem Turgutlu. CLIP-Art: Contrastive Pre-Training for Fine-Grained Art Classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[31] Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. Reciprocal Rank Fusion outperforms Condorcet and Individual Rank Learning Methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2009.

[32] Pankaj Dadure, Partha Pakray, and Sivaji Bandyopadhyay. An Empirical Analysis on Retrieval of Math Information from the Scientific Documents. In *International Conference on Communication and Intelligent Systems*. Springer, 2019.

[33] Pankaj Dadure, Partha Pakray, and Sivaji Bandyopadhyay. BERT-Based Embedding Model for Formula Retrieval. In *Working Notes of CLEF*, 2021.

[34] Yifan Dai, Liangyu Chen, and Zihan Zhang. An N-ary Tree-based Model for Similarity Evaluation on Mathematical Formulae. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020.

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

[36] Kenny Davila, Richard Zanibbi, Andrew Kane, and Frank Wm Tompa. Tangent-3 at the NTCIR-12 MathIR Task. In *NTCIR*, 2016.

[37] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

[38] Emanuele Di Buccio, Massimo Melucci, and Federica Moro. Detecting Verbose Queries and Improving Information Retrieval. *Information Processing and Management*, 2014.

[39] Yancarlos Diaz, Gavin Nishizawa, Behrooz Mansouri, Kenny Davila, and Richard Zanibbi. The MathDeck Formula Editor: Interactive Formula Entry Combining LaTeX , Structure Editing, and Search. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, 2021.

[40] Abishai Dmello. Representing Mathematical Concepts Associated With Formulas Using Math Entity Cards. [Master's Thesis] Rochester Institute of Technology, 2019.

[41] Zhicheng Dou, Ruihua Song, and Ji-Rong Wen. A Large-scale Evaluation and Analysis of Personalized Search Strategies. In *Proceedings of the 16th International Conference on World Wide Web*, 2007.

[42] Richard Fateman. A Critique of OpenMath and Thoughts on Encoding Mathematics, January, 2001. *University of California, Berkeley*, 2001.

[43] Jeffrey Flanigan, Sam Thomson, Jaime G Carbonell, Chris Dyer, and Noah A Smith. A Discriminative Graph-Based Parser for the Abstract Meaning Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014.

[44] Joseph L Fleiss. Measuring Nominal Scale Agreement Among Many Raters. *Psychological Bulletin*, 1971.

[45] Dallas Fraser, Andrew Kane, and Frank Wm. Tompa. Choosing Math Features for BM25 Ranking with Tangent-L. In *Proceedings of the ACM Symposium on Document Engineering*, 2018.

[46] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. An Efficient Boosting Algorithm for Combining Preferences. *Journal of Machine Learning Research*, 2003.

[47] Liangcai Gao, Zhuoren Jiang, Yue Yin, Ke Yuan, Zuoyu Yan, and Zhi Tang. Preliminary Exploration of Formula Embedding for Mathematical Information Retrieval. *arXiv:1707.05154*, 2017.

[48] Liangcai Gao and Yuehan Wang. ICST Math Retrieval System for NTCIR-11 Math-2 Task. In *NTCIR*. 2014.

[49] Sahil Garg, Aram Galstyan, Ulf Hermjakob, and Daniel Marcu. Extracting Biomolecular Interactions using Semantic Parsing of Biomedical Text. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[50] Deyan Ginev, Heinrich Stamerjohanns, Bruce R. Miller, and Michael Kohlhase. The LaTeXML Daemon: Editable Math on the Collaborative Web. In *Proceedings of the 18th Calculemus and 10th International Conference on Intelligent Computer Mathematics*. Springer-Verlag, 2011.

[51] Mihai Grigore, Magdalena Wolska, and Michael Kohlhase. Towards Context-based Disambiguation of Mathematical Expressions. In *the Joint Conference of ASCM*, 2009.

[52] Dongyi Guan, Sicong Zhang, and Hui Yang. Utilizing Query Change for Session Search. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2013.

[53] Ferruccio Guidi and Claudio Sacerdoti Coen. A Survey on Retrieval of Mathematical Knowledge. *Mathematics in Computer Science*, 2016.

[54] Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. Efficient Natural Language Response Suggestion for Smart Reply. *arXiv preprint*, 2017.

[55] MD Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga. A Comprehensive Survey of Deep Learning for Image Captioning. *ACM Computing Surveys (CsUR)*, 2019.

[56] Xuan Hu, Liangcai Gao, Xiaoyan Lin, Zhi Tang, Xiaofan Lin, and Josef B Baker. Wikimirs: a Mathematical Information Retrieval System for Wikipedia. In *Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital libraries*, 2013.

[57] Sharaf Hussain, Samita Bai, and Shakeel Khoja. Content MathML(CMML) conversion using LATEX Math Grammar (LMG). In *7th International Conference on Smart Computing and Communications (ICSCC)*. IEEE, 2019.

[58] Patrick Ion, Robert Miner, Stephen Buswell, and A Devitt. *Mathematical Markup Language (MathML) 1.0 Specification*. World Wide Web Consortium (W3C), 1998.

[59] Bernard J Jansen, Danielle L Booth, and Amanda Spink. Determining the User Intent of Web Search Engine Queries. In *Proceedings of the 16th International Conference on World Wide Web*, 2007.

[60] Jyun-Yu Jiang, Yen-Yu Ke, Pao-Yu Chien, and Pu-Jen Cheng. Learning User Reformulation Behavior for Query Auto-Completion. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2014.

[61] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for Natural Language Understanding. *arXiv preprint arXiv:1909.10351*, 2019.

[62] Thorsten Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '06*. ACM, 2006.

[63] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data*, 2019.

[64] Shahab Kamali and Frank Wm Tompa. Improving Mathematics Retrieval. *Towards a Digital Mathematics Library*, 2009.

[65] Shahab Kamali and Frank Wm. Tompa. Retrieving Documents With Mathematical Content. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2013.

[66] Chris Kamphuis, Arjen P de Vries, Leonid Boytsov, and Jimmy Lin. Which BM25 Do You Mean? A Large-Scale Reproducibility Study of Scoring Variants. *Advances in Information Retrieval*, 2020.

[67] Andrew Kane, Yin Ki Ng, and Frank Wm Tompa. Dowsing for Answers to Math Questions: Doing Better with Less. In *CLEF (Working Notes)*, 2022.

[68] Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Salim Roukos, Alexander Gray, Ramón Fernandez Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue-Nkoutche, et al. Leveraging Abstract Meaning Representation for Knowledge Base Question Answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP*, 2021.

[69] Omar Khattab and Matei Zaharia. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020.

[70] Youngho Kim, Ahmed Hassan, Ryen W White, and Imed Zitouni. Modeling Dwell Time to Predict Click-level Satisfaction. In *Proceedings of the 7th ACM International Conference on Web search and data mining*, 2014.

[71] Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models. *arXiv preprint*, 2014.

[72] Ryan Kiros, Richard Zemel, and Russ R Salakhutdinov. A Multiplicative Model for Learning Distributed Text-Based Attribute Representations. *Advances in Neural Information Processing Systems*, 2014.

[73] Michael Kohlhase and Ioan Sucan. A Search Engine for Mathematical Formulae. In *International Conference on Artificial Intelligence and Symbolic Computation*. Springer, 2006.

[74] Giovanni Yoko Kristianto, Akiko Aizawa, et al. Extracting Textual Descriptions of Mathematical Expressions in Scientific Papers. *D-Lib Magazine*, 2014.

[75] Giovanni Yoko Kristianto, Goran Topic, and Akiko Aizawa. MCAT Math Retrieval System for NTCIR-12 MathIR Task. In *NTCIR*, 2016.

[76] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, 2012.

[77] Kriste Krstovski and David M. Blei. Equation Embeddings. *arXiv:1803.09123*, 2018.

[78] P Pavan Kumar, Arun Agarwal, and Chakravarthy Bhagvati. A Structure Based Approach for Mathematical Expression Retrieval. In *International Workshop on Multi-disciplinary Trends in Artificial Intelligence*. Springer, 2012.

[79] Leslie Lamport. LATEX–A Document Preparation System Addison–Wesley. *Reading, MA*, 1985.

[80] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *arXiv preprint*, 2019.

[81] Irene Langkilde and Kevin Knight. Generation that Exploits Corpus-Based Statistical Knowledge. In *the 17th International Conference on Computational Linguistics*, 1998.

[82] Matt Langsenkamp, Behrooz Mansouri, and Richard Zanibbi. Expanding Spatial Regions and Incorporating IDF for PHOC-Based Math Formula Retrieval at ARQMath-3. In *CLEF (Working Notes)*, 2022.

[83] Quoc Le and Tomas Mikolov. Distributed Representations of Sentences and Documents. In *International Conference on Machine Learning*, 2014.

[84] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[85] Lin Li, Lu Qi, Fang Deng, Shengwu Xiong, and Jingling Yuan. Enhancing Keyword Suggestion of Web Search by Leveraging Microblog Data. *J. Web Eng.*, 2016.

[86] Kexin Liao, Logan Lebanoff, and Fei Liu. Abstract Meaning Representation for Multi-Document Summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, 2018.

[87] Paul Libbrecht and Erica Melis. Methods to Access and Retrieve Mathematical Content in ActiveMath. In *International Congress on Mathematical Software*. Springer, 2006.

[88] Xiaoyan Lin, Liangcai Gao, Xuan Hu, Zhi Tang, Yingnan Xiao, and Xiaozhong Liu. A Mathematics Retrieval System for Formulae in Layout Presentations. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2014.

[89] Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A Smith. Toward Abstractive Summarization Using Semantic Representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

[90] Qiaoling Liu, Eugene Agichtein, Gideon Dror, Yoelle Maarek, and Idan Szpektor. When Web Search Fails, Searchers Become Askers: Understanding the Transition. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2012.

[91] Xiaoxiao Liu, Qingyang Xu, and Ning Wang. A Survey on Deep Neural Network-based Image Captioning. *The Visual Computer*, 2019.

[92] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint*, 2019.

[93] Yuanhua Lv and ChengXiang Zhai. Lower-Bounding Term Frequency Normalization. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, 2011.

[94] Martin Líška, Petr Sojka, Michal Růžička, and Petr Mravec. Web Interface and Collection for Mathematical Retrieval WebMIaS and MREC. *Towards a Digital Mathematics Library*, 2011.

[95] Craig Macdonald and Nicola Tonellotto. Declarative Experimentation in Information Retrieval using Pyterrier. In *Proceedings of the ACM SIGIR on International Conference on Theory of Information Retrieval*, 2020.

[96] Behrooz Mansouri. Embedding Formulae and Text for Improved Math Retrieval. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021.

[97] Behrooz Mansouri, Anurag Agarwal, Douglas Oard, and Richard Zanibbi. Finding Old Answers to New Math Questions: The ARQMath Lab at CLEF 2020. In *European Conference on Information Retrieval*, 2020.

[98] Behrooz Mansouri, Anurag Agarwal, Douglas W Oard, and Richard Zanibbi. Advancing Math-Aware Search: The ARQMath-2 Lab at CLEF 2021. In *European Conference on Information Retrieval*, 2021.

[99] Behrooz Mansouri, Anurag Agarwal, Douglas W Oard, and Richard Zanibbi. Advancing Math-Aware Search: The ARQMath-3 Lab at CLEF 2022. In *European Conference on Information Retrieval*, 2022.

[100] Behrooz Mansouri, Vit Novotný, Anurag Agarwal, Douglas W Oard, and Richard Zanibbi. Overview of ARQMath-3 (2022): Third CLEF Lab on Answer Retrieval for Questions on Math. In *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer, 2022.

[101] Behrooz Mansouri, Douglas W Oard, Anurag Agarwal, and Richard Zanibbi. Effects of Context, Complexity, and Clustering on Evaluation for Math Formula Retrieval. *arXiv preprint*, 2021.

[102] Behrooz Mansouri, Douglas W Oard, and Richard Zanibbi. DPRL Systems in the CLEF 2020 ARQMath Lab. In *CLEF (Working Notes)*, 2020.

[103] Behrooz Mansouri, Douglas W Oard, and Richard Zanibbi. DPRL Systems in the CLEF 2021 ARQMath Lab: Sentence-BERT for Answer Retrieval, Learning-to-Rank for Formula Retrieval. In *CLEF (Working Notes)*, 2021.

[104] Behrooz Mansouri, Douglas W Oard, and Richard Zanibbi. DPRL Systems in the CLEF 2022 ARQMath Lab: Introducing MathAMR for Math-Aware Search. In *CLEF (Working Notes)*, 2022.

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

[106] Behrooz Mansouri, Richard Zanibbi, and Douglas W. Oard. Characterizing Searches for Mathematical Concepts. In *2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, 2019.

[107] Behrooz Mansouri, Richard Zanibbi, and Douglas W. Oard. Learning to Rank for Mathematical Formula Retrieval. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021.

[108] Behrooz Mansouri, Richard Zanibbi, Douglas W Oard, and Anurag Agarwal. Overview of ARQMath-2 (2021): Second CLEF Lab on Answer Retrieval for Questions on Math. In *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer, 2021.

[109] Jonathan May and Jay Priyadarshi. Semeval-2017 Task 9: Abstract Meaning Representation Parsing and Generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, 2017.

[110] Jordan Meadows and Andre Freitas. A Survey in Mathematical Language Processing. *arXiv preprint*, 2022.

[111] Bruce R Miller and Abdou Youssef. Technical Aspects of the Digital Library of Mathematical Functions. *Annals of Mathematics and Artificial Intelligence*, 2003.

[112] Jozef Mišutka and Leo Galamboš. Extending Full Text Search Engine for Mathematical Content. *Towards Digital Mathematics Library*, 2008.

[113] Bhaskar Mitra and Nick Craswell. Query Auto-Completion for Rare Prefixes. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015.

[114] Rajesh Munavalli and Robert Miner. MathFind: A Math-Aware Search Engine. In *Proceedings of the 29th annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2006.

[115] Yin Ki Ng, Dallas J Fraser, Besat Kassaie, George Labahn, Mirette S Marzouk, Frank Wm Tompa, and Kevin Wang. Dowsing for Math Answers with Tangent-L. In *CLEF (Working Notes)*, 2020.

[116] Yin Ki Ng, Dallas J Fraser, Besat Kassaie, and Frank Wm Tompa. Dowsing for Answers to Math Questions: Ongoing Viability of Traditional MathIR. In *CLEF (Working Notes)*, 2021.

[117] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In *CoCo@ NIPS*, 2016.

[118] Maximillian Nickel and Douwe Kiela. Poincaré Embeddings for Learning Hierarchical Representations. *Advances in Neural Information Processing Systems*, 2017.

[119] Gavin Nishizawa, Jennifer Liu, Yancarlos Diaz, Abishai Dmello, Wei Zhong, and Richard Zanibbi. MathSeer: A Math-Aware Search Interface with Intuitive Formula Editing, Reuse, and Lookup. In *Advances in Information Retrieval*. Springer International Publishing, 2020.

[120] Immanuel Normann and Michael Kohlhase. Extended Formula Normalization for ε-Retrieval and Sharing of Mathematical Knowledge. In *Towards Mechanized Mathematical Assistants*. Springer, 2007.

[121] Vít Novotnỳ, Petr Sojka, Michal Štefánik, and Dávid Lupták. Three is Better than One. Ensembling Math Information Retrieval Systems. In *CLEF (Working Notes)*, 2020.

[122] Vít Novotný, Petr Sojka, Michal Štefánik, and Dávid Lupták. Ensembling Math Information Retrieval Systems. In *CLEF (Working Notes)*, 2021.

[123] Association of American Publishers. Markup of Mathematical Formulas. *APP Inc.*, 1986.

[124] Iadh Ounis, Gianni Amati, Vassilis Plachouras, Ben He, Craig Macdonald, and Douglas Johnson. Terrier Information Retrieval Platform. In *European Conference on Information Retrieval*. Springer, 2005.

[125] Amarnath Pathak, Ranjita Das, Partha Pakray, and Alexander Gelbukh. Extracting Context of Math Formulae Contained inside Scientific Documents. *Computacion y Sistemas*, 2019.

[126] Mateusz Pawlik and Nikolaus Augsten. Efficient Computation of the Tree Edit Distance. *ACM Transactions on Database Systems (TODS)*, 2015.

[127] Shuai Peng, Ke Yuan, Liangcai Gao, and Zhi Tang. MathBERT: A Pre-Trained Model for Mathematical Formula Understanding. *arXiv:2105.00377*, 2021.

[128] Lukas Pfahler and Katharina Morik. Semantic Search in Millions of Equations. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020.

[129] John R Pierce. *An Introduction to Information Theory: Symbols, Signals and Noise*. Courier Corporation, 2012.

[130] NAFM Poppelier, E van Herwijnen, and CA Rowley. Standard DTD's and Scientific Publishing. *EPSIG News*, 1992.

[131] Kader Pustu-Iren, Gerrit Bruns, and Ralph Ewerth. A Multimodal Approach for Semantic Patent Image Retrieval. In *Proceedings of the 2nd Workshop on Patent Text Mining and Semantic Technologies (PatentSemTech)*. Aachen, Germany: RWTH Aachen, 2021.

[132] Dimitrios Rafailidis, Stavroula Manolopoulou, and Petros Daras. A Unified Framework for Multimodal Retrieval. *Pattern Recognition*, 2013.

[133] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019.

[134] Anja Reusch, Maik Thiele, and Wolfgang Lehner. An ALBERT-based Similarity Measure for Mathematical Answer Retrieval. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021.

[135] Anja Reusch, Maik Thiele, and Wolfgang Lehner. TU_DBS in the ARQMath Lab 2021. In *Working Notes of CLEF*, 2021.

[136] Shaurya Rohatgi, Jian Wu, and C Lee Giles. PSU at CLEF-2020 ARQMath Track: Unsupervised Re-ranking using Pretraining. 2020.

[137] Shaurya Rohatgi, Jian Wu, and C Lee Giles. Ranked List Fusion and Re-ranking with Pretrained Transformers for ARQMath Lab. 2021.

[138] Michal Ržička, Petr Sojka, Martin Líška, et al. Math Indexer and Searcher under the Hood: Fine-tuning Query Expansion and Unification Strategies. In *Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies*, 2016.

[139] Tetsuya Sakai. Alternatives to Bpref. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '07*, 2007.

[140] Sandip Sarkar, Dipankar Das, Partha Pakray, and David Pinto. Formula Retrieval Using Structural Similarity. In *Working Notes of CLEF*, 2022.

[141] Philipp Scharpf, Moritz Schubotz, and Bela Gipp. Representing Mathematical Formulae in Content MathML using Wikidata. In *BIRNDL@ SIGIR*, 2018.

[142] Moritz Schubotz, Alexey Grigorev, Marcus Leich, Howard S Cohl, Norman Meuschke, Bela Gipp, Abdou S Youssef, and Volker Markl. Semantification of Identifiers in Mathematics for Better Math Information Retrieval. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2016.

[143] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. *arXiv preprint arXiv:1508.07909*, 2015.

[144] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

[145] Rebecca Sharp, Adarsh Pyarelal, Benjamin Gyori, Keith Alcock, Egoitz Laparra, Marco A Valenzuela-Escárcega, Ajay Nagesh, Vikas Yadav, John Bachman, Zheng Tang, et al. Eidos, INDRA, & Delphi: From Free Text to Executable Causal Models. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

[146] Jia Tracy Shen, Michiharu Yamashita, Ethan Prihar, Neil Heffernan, Xintao Wu, Ben Graff, and Dongwon Lee. MathBERT: A Pre-trained Language Model for General NLP Tasks in Mathematics Education. *arXiv preprint*, 2021.

[147] Richard A Silverman. *Essential Calculus with Applications*. Courier Corporation, 1989.

[148] Petr Sojka and Martin Líška. Indexing and Searching Mathematics in Digital Libraries. In James H. Davenport, William M. Farmer, Josef Urban, and Florian Rabe, editors, *Intelligent Computer Mathematics*. Springer, 2011.

[149] Yang Song and Li-wei He. Optimal Rare Query Suggestion With Implicit User Feedback. In *Proceedings of the 19th International Conference on World Wide Web*, 2010.

[150] Yiannos Stathopoulos, Simon Baker, Marek Rei, and Simone Teufel. Variable Typing: Assigning Meaning to Variables in Mathematical Text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018.

[151] Yiannos Stathopoulos and Simone Teufel. Retrieval of Research-level Mathematical Information Needs: A Test Collection and Technical Terminology Experiment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, 2015.

[152] Wen Tai, HT Kung, Xin Luna Dong, Marcus Comiter, and Chang-Fu Kuo. ExBERT: Extending Pre-trained Models with Domain-specific Vocabulary Under Constrained Training Resources. In *Findings of the Association for Computational Linguistics: EMNLP*, 2020.

[153] Yla R Tausczik, Aniket Kittur, and Robert E Kraut. Collaborative Problem Solving: A Study of MathOverflow. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work and Social Computing*, 2014.

[154] Abhinav Thanda, Ankit Agarwal, Kushal Singla, Aditya Prakash, and Abhishek Gupta. A Document Retrieval System for Math Queries. In *NTCIR*, 2016.

[155] George Brinton Thomas, Maurice D Weir, and Joel Hass. *Thomas' Calculus: Single Variable*. Pearson, 2013.

[156] Noriko Tomuro. Question Terminology and Representation for Question Type Classification. *Terminology. International Journal of Theoretical and Applied Issues in Specialized Communication*, 2004.

[157] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and Tell: A Neural Image Caption Generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[158] Björn von Sydow. The Design of the EUROMath System. *Euromath Bulletin*, 1992.

[159] Chuan Wang, Nianwen Xue, and Sameer Pradhan. A Transition-Based Algorithm for AMR Parsing. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

[160] Zichao Wang, Andrew Lan, and Richard Baraniuk. Mathematical Formula Representation via Tree Embeddings. In *CEUR Workshop Proceedings*, 2021.

[161] Ryen W White and Resa A Roth. Exploratory Search: Beyond the Query–Response Paradigm. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 2009.

[162] Qiang Wu, Chris JC Burges, Krysta M Svore, and Jianfeng Gao. Ranking, Boosting, and Model Adaptation. Technical report, 2008.

[163] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's Neural Machine Translation System: Bridging the Gap Between Human and Machine Translation. *arXiv preprint*, 2016.

[164] Weiwen Xu, Huihui Zhang, Deng Cai, and Wai Lam. Dynamic Semantic Graph Construction and Reasoning for Explainable Multi-hop Science Question Answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP*, 2021.

[165] Peilin Yang, Hui Fang, and Jimmy 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.

[166] Michihiro Yasunaga and John D. Lafferty. TopicEq: A Joint Topic and Mathematical Equation Model for Scientific Texts. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[167] Abdou Youssef. Roles of Math Search in Mathematics. In *International Conference on Mathematical Knowledge Management*. Springer, 2006.

[168] Ke Yuan, Zuoyu Yan, Yibo Li, Liangcai Gao, and Zhi Tang. Automatic Description Construction for Math Expression via Topic Relation Graph. *arXiv preprint*, 2021.

[169] Mohammad Sadegh Zahedi, Behrouz Mansouri, Shiva Moradkhani, Mojgan Farhoodi, and Farhad Oroumchian. How Questions are Posed to a Search Engine? An Empiricial Analysis of Question Queries in a Large Scale Persian Search Engine Log. In *the 3th International Conference on Web Research (ICWR)*. IEEE, 2017.

[170] Richard Zanibbi, Akiko Aizawa, Michael Kohlhase, Iadh Ounis, Goran Topic, and Kenny Davila. NTCIR-12 MathIR Task Overview. In *NTCIR*, 2016.

[171] Richard Zanibbi and Dorothea Blostein. Recognition and Retrieval of Mathematical Expressions. *International Journal on Document Analysis and Recognition (IJDAR)*, 2012.

[172] Richard Zanibbi, Kenny Davila, Andrew Kane, and Frank Wm. 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.

[173] Richard Zanibbi, Douglas W Oard, Anurag Agarwal, and Behrooz Mansouri. Overview of ARQMath 2020: CLEF Lab on Answer Retrieval for Questions on Math. In *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer, 2020.

[174] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level Convolutional Networks for Text Classification. *Advances in Neural Information Processing Systems*, 2015.

[175] Zixuan Zhang and Heng Ji. Abstract Meaning Representation Guided Graph Encoding and Decoding for Joint Information Extraction. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021.

[176] Jin Zhao, Min-Yen Kan, and Yin Leng Theng. Math Information Retrieval: User Requirements and Prototype Implementation. In *Proceedings of the 8th ACM/IEEE-CS Joint Conference on Digital libraries - JCDL '08*. ACM, 2008.

[177] Wei Zhong, Shaurya Rohatgi, Jian Wu, C. Lee Giles, and Richard Zanibbi. Accelerating Substructure Similarity Search for Formula Retrieval. In *Advances in Information Retrieval*. Springer, 2020.

[178] Wei Zhong, Yuqing Xie, and Jimmy Lin. Applying Structural and Dense Semantic Matching for the ARQMath Lab 2021, CLEF. In *CLEF (Working Notes)*, 2022.

[179] Wei Zhong, Xinyu Zhang, Ji Xin, Richard Zanibbi, and Jimmy Lin. Approach Zero and Anserini at the CLEF-2021 ARQMath Track: Applying Substructure Search and BM25 on Operator Tree Path Tokens. In *CLEF (Working Notes)*, 2021.

[180] Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. Retrieving and Reading: A Comprehensive Survey on Open-Domain Question Answering. *arXiv preprint*, 2021.

[181] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. Fast and Accurate Shift-Reduce Constituent Parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013.

Appendix

Fixes in ARQMath-3

In our last release of ARQMath collection in 2022, we fixed existing issues we were notified about in the previous years. We have made the following improvements to the collection:

1. **Formula Representations.** We found and corrected 65,681 formulas with incorrect Symbol Layout Tree (SLT) and Operator Tree (OPT) representations. This resulted from incorrect handling of errors generated by the LaTeXML tool that had been used for generating those representations.

2. **Clustering Visually Distinct Formulas.** Correcting SLT representations resulted in a need to adjust the clustering of formula instances. Each cluster of visually identical formulas was assigned a unique 'Visual ID'. Clustering had been performed using SLT where possible, and LaTeX otherwise. To correct the clustering, we split any cluster that now included formulas with different representations. In such cases, the partition with the largest number of instances retained its Visual ID; remaining formulas were assigned to another existing Visual ID (with the same SLT or LaTeX) or, if necessary, to a new Visual ID. To break ties, the partition with the largest cumulative ARQMath-2 relevance score retained its Visual ID or, failing that, choosing the partition with the lowest Formula ID. 29,750 new Visual IDs resulted.

3. **XML Errors.** In the XML files for posts and comments, the LaTeX for each formula is encoded as a `<span>` XML element with the class attribute `math-container`. We found and corrected 108,242 formulas that had not been encoded in that way.

4. **Spurious Formula Identifiers.** The ARQMath collection includes an index file that includes Formula ID, Visual ID, Post ID, SLT, OPT, and LaTeX for each formula instance. However, there were also formulas in the index file that did not actually occur in any post or comment in the collection. This happened because formula extraction was initially done on the Post History file, which also contained some content that had later been removed. We added a new annotation to the formula index file to mark such cases.