Foundations and Trends® in Information Retrieval

Mathematical Information Retrieval: Search and Question Answering

Suggested Citation: Richard Zanibbi, Behrooz Mansouri and Anurag Agarwal (2025), "Mathematical Information Retrieval: Search and Question Answering", Foundations and Trends® in Information Retrieval: Vol. 19, No. 1-2, pp 1–190. DOI: 10.1561/1500000095.

Richard Zanibbi
Rochester Institute of Technology
rxzvcs@rit.edu

Behrooz Mansouri
University of Southern Maine
behrooz.mansouri@maine.edu

Anurag Agarwal
Rochester Institute of Technology
axasma@rit.edu

now
the essence of knowledge
Boston — Delft

Contents

Mathematical Information Retrieval: Search and Question Answering

Richard Zanibbi[1], Behrooz Mansouri[2] and Anurag Agarwal[3]

[1]*Department of Computer Science, Rochester Institute of Technology, USA; rxzvcs@rit.edu*
[2]*Department of Computer Science, University of Southern Maine, USA; behrooz.mansouri@maine.edu*
[3]*School of Mathematics & Statistics, Rochester Institute of Technology, USA; axasma@rit.edu*

ABSTRACT

Mathematical information is essential for technical work, but its creation, interpretation, and search are challenging. To help address these challenges, researchers have developed multimodal search engines and mathematical question answering systems. This monograph begins with a simple framework characterizing the information tasks that people and systems perform as we work to answer math-related questions. The framework is used to organize and relate the other core topics of the monograph, including interactions between people and systems, representing math formulas in sources, and evaluation. We close by addressing some key questions and presenting directions for future work. This monograph is intended for students, instructors, and researchers interested in systems that help us find and use mathematical information.

Richard Zanibbi, Behrooz Mansouri and Anurag Agarwal (2025), "Mathematical Information Retrieval: Search and Question Answering", Foundations and Trends® in Information Retrieval: Vol. 19, No. 1-2, pp 1–190. DOI: 10.1561/1500000095.
©2025 R. Zanibbi *et al.*

Preface

This monograph provides an introduction to the foundations and current developments in *mathematical information retrieval* research, or *math IR*. In this area we focus on systems designed to assist with finding, collecting, and using mathematical information. With the advent of Large Language Models (LLMs), mathematical question answering is of particularly keen interest at the moment. Systems that combine LLMs with logic-based systems have been making news by solving problems from the International Mathematical Olympiad, for example.

The authors have spent more than a decade working on systems and interfaces for math-aware search engines in web pages, PDF documents, and even videos. We have more recently worked with interactive *conversational* systems for math IR, and looked into applications of LLMs. In this monograph we try to summarize what we have learned about systems for searching existing sources, and briefly introduce emerging math question answering systems that automatically generate responses based on usage patterns for text *and* formulas.

Our intended audience includes students, instructors, and researchers of information science, computer science, and mathematics.[1] Because the goal of this series is to introduce a topic from its fundamentals and then build up to the state-of-the-art, we needed to prioritize making

[1]With this said, we have often been ‘unintended’ readers of monographs, and warmly welcome anyone with even the slightest interest in what we have to say here.

the presentation as clear and concrete as possible. This means that we also had to make difficult decisions about what material to include. We have focused on covering what we understand to be core topics and concerns, rather than provide an exhaustive survey of techniques. For those interested in learning more than we could cover here, we refer you to other math IR surveys that are available (Zanibbi and Blostein, 2012; Guidi and Sacerdoti Coen, 2016; Dadure *et al.*, 2024).

Our approach in this work is searcher-centered, *i.e.,* focused on models and systems used directly by people. We acknowledge that there are *vast* bodies of literature concerned with searching and discovering mathematical information with little or no human interaction. Important examples include automated theorem proving, one of the oldest and most influential corners of artificial intelligence research, and work with mathematical knowledge databases in the Mathematical Knowledge Management community (*e.g.,* at the CICM[2] conferences), from which much of the early influential work in math IR also originates from. In Appendix B we present related work from the theorem proving space, but this is only a brief overview, and just the tip of the iceberg.

Structured representations for formulas and computations precede the electronic computer. The tree-based representations for formula structure presented in Section 2 were designed to be simple and capture key structural properties, *i.e.,* writing lines of symbols for formula appearance, and operation hierarchies for mathematical expressions represented by formulas. However, we make no assertion of their novelty; many analagous representations have been used in papers and systems. We've found that Symbol Layout Trees (SLTs) and Operator Trees (OPTs) generalize the graph types used for recognition and retrieval of formulas well.

We close here with a brief summary of the contents of this monograph. The sections of the monograph are summarized visually in Figure 1, and in more detail below.

Section 1 introduces math IR, and presents a framework that unifies information seeking activities performed by people *and* systems in (1) the real world, with (2) 'traditional' retrieval systems, and (3)

[2]https://cicm-conference.org/cicm.php

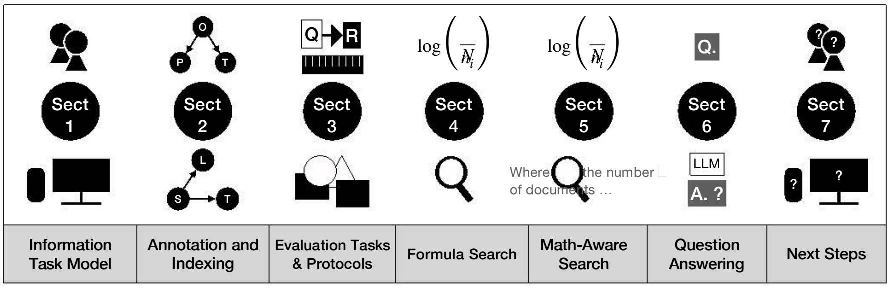


Figure 1: Visual summary of this monograph's contents

question answering systems. The framework is organized around information needs, sources, and tasks, with an informal 'source jar' model for human information seeking, and a structured task graph for systems. The systems-oriented model is used to organize material in Sections 2–6.

Section 2 considers the types of mathematical information present (and missing) in sources, and provides an overview of formula representations. The larger focus is annotating sources with additional information (*e.g.,* formula representations) and representing text and formulas in indexes for 'sparse' (*i.e.,* discrete pattern-based lookup) and 'dense' (*i.e.,* continuous vector space-based lookup) retrieval.

Section 3 presents the math IR tasks addressed in Sections 4–6, along with procedures for creating *test collections* for evaluation, and evaluation metrics used in benchmarks. These are presented together to emphasize similarities and differences between retrieval and question answering tasks.

Sections 4–6 present past and current systems for formula search, text + formula search ('math-aware' search), and question answering. These systems depend upon indexing and evaluation techniques covered in Sections 2 and 3. Each section begins with a summary of test collections for evaluation, followed by a presentation and comparison of methods.

Section 7 provides a closing summary, a discussion of what math IR *cannot* provide, and directions for future work organized around the information task framework from Section 1.

Richard Zanibbi, Rochester, NY, USA
Behrooz Mansouri, Portland, ME, USA
Anurag Agarwal, Rochester, NY, USA
December 2024

1

Sources and Information Tasks

We often pause to search when something that we read, watch, or hear prompts questions that we want answers to. We then go about finding answers using additional sources of information: some already exist, some are created in response to requests (*e.g.,* emails or search results), and some are created to record and organize what we find.

In this way, information sources are the backbone of our information ecosystems. The sources available to us place a hard limit upon which questions we can answer. In addition to the information content in a source, its terminology, notation, writing style, and other factors determine the amount of information one can recover from a source, and how accurately and completely. This is a key reason why math instructors that communicate well are so highly regarded: they help us more easily understand topics by *how* they speak, write, and present exercises. Through course materials, lectures, and conversations, these instructors provide multiple sources tailored to their students' level of understanding and communication style.

Outside the classroom, we still often find ourselves in need of mathematical information. It might be as simple as finding a formula to convert temperatures in Fahrenheit to Celsius, or the formula associated with a name (*e.g.,* inverse document frequency). Or the goal may be more complex, such as understanding a proof of the *sensitivity conjecture*.

As we look for answers, we will in some way annotate and organize the sources we find in order to identify and apply pertinent information, *e.g.,* to find other sources, choose different search terms, execute suggested exercises, and make notes about partial answers to our questions. The effort needed for these tasks depends largely on the content and presentation in the sources that we access. To save time, we often create additional sources of our own (*e.g.,* bookmarking a web page, placing notes in a file, or highlighting a PDF document).

In this work, when we speak about **sources**, we are usually referring to individual documents, recordings (*e.g.,* videos) or other artifacts that contain information. Libraries and other people are of course also information sources, in the sense that they can provide information, but here we use 'sources' to refer to records of specific information.

To help organize our study of mathematical information retrieval, in this section we introduce a framework for information tasks based on sources. The framework is built upon two main ideas:

1. Search begins, progresses, and ends with sources.
2. Tasks other than search are often needed to find information.

The key components of the framework are:

- **information needs** that individuals have,
- **sources of information** that we search, consult, and create,
- information **tasks** performed to address information needs, and
- their roles in **search algorithms and user-interfaces**.

In the next section, we consider how these components interact when we have a mathematical question that we wish to answer.

1.1 When and Where Do We Search?

Some short answers to this question are (1) when we have a question, and (2) wherever is easiest. While not very satisfying, these answers are basically correct. Search is generally performed as part of some larger information task, and not for its own sake.[1] This motivates finding quick paths to answers.

However, technical subjects such as mathematics can be complex. Finding and understanding information on math may require multiple activities, such as web search, reading sources (*e.g.,* Wikipedia pages and textbooks), taking notes, talking to instructors or colleagues, and doing exercises. As a result, when retrieving technical material on math and other specialized topics (*e.g.,* law, chemistry, music history), it is helpful to understand how search interacts with other information tasks.

To illustrate, consider the more general problem of *sensemaking*, which learning about detailed mathematical topics is closely related to.[2] In sensemaking, we construct a conceptual understanding of a topic with many sources, usually along with communicating this understanding. Common examples include writing a school term paper on an unfamiliar topic (*e.g.,* applications of category theory), or summarizing a complex historical event from multiple news reports.

Sensemaking tasks are challenging because information must be found in multiple sources, but also because this information must be analyzed, compared, and integrated. These thinking activities often require most of the effort for sensemaking. To manage these thinking tasks, we record plans, notes, and outlines to organize our work. These working documents may be checked repeatedly as we work, and as we write our final summary. They are themselves important information sources that provide the scaffolding needed to focus and ultimately complete work on a sensemaking task.

To further illustrate information tasks that complement search, imagine taking handwritten notes on eigenvectors as described in a linear algebra textbook. The notes allow us to annotate this source with our own observations, and record them for reference at a later time. The analysis and insights in the notes come from applying information that we know and find. These notes communicate a new information source to a specialized audience: ourselves.

[1] A fact that is both important and humbling for IR researchers.

[2] See Hearst (2009) for an overview of early research on sensemaking.

For our notes to be useful, we organize them. Perhaps this is a purple sticky note that we attach to a monitor to check later in the evening. Or, perhaps we use a paper notebook with separate sections for different subjects, along with other organizational devices (*e.g.,* sticky notes acting as bookmarks). We might instead be using a tablet computer, which also provides handwriting recognition to convert the notes to computer-searchable data (*e.g.,* using `Ctrl-f`).

The information tasks above are distinct from a basic search task where we submit a query, post a question, or send an email to obtain new information sources. However, it turns out that search engines implement variations of the same information tasks described above: they need to *index*, *communicate*, *annotate*, and *apply* information in sources to be effective. For example, we organize sources when we arrange sticky notes by topic and color on a wall, or construct an *inverted index* mapping words or formulas to their document locations: these are both forms of *indexing.* As another example, search engines produce Search Engine Result Pages (SERPs) summarizing documents matching a query, and question answering systems or AI 'bots' produce answers. These retrieval system outputs and our notes are *communications* creating new information sources.

Making notes on a passage requires us to *apply* information to create an *annotation*: additional information associated with the passage. In turn, if those notes were handwritten on a tablet computer, a system converting these to text and LaTeX for math applies information captured in an algorithm, annotating the notes themselves. We end up with a hierarchy of annotations: the notes annotate a passage, while a recognition algorithm annotates the notes.

In our framework we will distinguish different source types, based largely on what information tasks they are primarily used for. More specifically, we distinguish:

1. available sources on a topic including search queries and results,
2. information added to sources (*annotation*), and
3. structures and organizations created for search (*indexing*).

Getting back to our motivating question, when we have identified a mathematical *information need*, we generally start with questions, and hope to end with one or more information sources that we feel address or ideally answer those questions (*i.e., relevant* sources for the information need). Where we search is motivated by the types of sources we expect to find from places online and/or the physical world (*e.g.,* conversations and post-its). Unless we are casually browsing resources on a topic, the places and order in which these sources are found will generally reflect attempts to reduce our time and effort.[3] Relevant sources are often of different types: perhaps a passage in a web page along with a SERP page, an answer from an online AI system, an email from a friend, and a green sticky note on your monitor.

From this perspective, math-aware search engines and question answering systems are important tools, but only one among many resources for finding math information, and only a small part of what happens when we search for mathematical information.

1.2 Information Task Framework

While we focus in this monograph on information retrieval using computers, we wish to address sources in their broadest sense here. Not all sources are text documents, and not all sources are recorded in documents. Consider an informal conversation about Bayesian decision theory in the hallway, or observing that there are no clouds in the sky: often, your only record of important information is your own memory.

In addition to textbooks, technical papers, and web pages, in recent years the types of resources used to locate mathematical information has grown to include substantial amounts of video (Davila *et al.*, 2021) and audio, *e.g.,* for course lectures, tutorials, and technical talks. Community Question Answering sites and direct question answering is provided by resources such as Math Stack Exchange,[4] WolframAlpha, and large language models such as the Generative Pre-trained Transformer (GPT).

It is worth noting that when we have found or produced information we want to share or reuse, we usually produce a source of information ourselves. For example, when we have an answer to a math homework

[3]Information foraging theory (Pirolli and Card, 1999) suggests we evolved to gather and consume information similar to food, governed by cost-benefit analyses.

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

question, we create a physical or digital document, so that this can be checked by ourselves and graded by our instructor. If we found a helpful video while doing the homework, we might share it in a text message, which is itself a form of ‘micro-source.’

Differences in Information Sources. Especially when we include information obtained directly from our environment along with modern computing and communication devices, information sources may come in many forms. Sources vary in the dimensions listed below, among others.

- immediacy, *e.g.,* having a conversation vs. reading a transcription
- authorship, *e.g.,* human, machine generated, environment
- interactivity, *e.g.,* a human/chatbot conversation vs. a document
- audience, *e.g.,* grade school students vs. math professors
- modality, *e.g.,* text, video, audio, or a web page combining these
- purpose, *e.g.,* textbook, search results, or a search index
- structure, *e.g.,* free text in a sticky note, vs. a book with chapters
- length
- formality, *e.g.,* proof vs. text message
- style, *e.g.,* how concepts and examples are communicated
- correctness, *e.g.,* correct vs. incorrect definition or proof
- completeness, *e.g.,* partial vs. full search index
- type of information, *e.g.,* technical, notes, communications

For the *immediacy* of a source, we are referring to whether the source comes directly from observing a person’s environment (*e.g.,* through conversation, experimentation, travel, etc.) or is recorded, as in a document or audiovisual recording. Particularly with the advent of large language models, authorship and correctness are important concerns. In many cases LLMs and other sources may appear credible but are incorrect. Knowing how a source is created can help us determine how trusting we should be of it when we are uncertain about validity (*e.g.,* from the perceived expertise of an author or system).

The intended audience and style of a source are also critical concerns. They determine the prerequisite knowledge needed to decide whether a source is relevant and to interpret and use information in the source.

Information task types. The primary task types we will use for people and systems come from common descriptions of sensemaking and simpler information tasks: retrieving, analyzing, and synthesizing. For this framework, these **tasks are considered at the source level**. For example, *analyzing* a source refers to analysis that produces additional recorded information for a source (*e.g.,* in a note) rather than reading and interpreting a source without producing an observable artifact.

We subdivide each of these into two subtasks based on how sources are created and used, producing six tasks in total, as shown in Figure 1.1. The *apply* task is critical, and used for all other tasks (*e.g.,* creating queries, consulting or creating information sources, or generating annotations and/or indexes). It is distinct because people often apply information without producing observable sources. For example, recognizing what a variable represents does not involve creating an annotation, index, or source outside of our own minds. The *apply* task also identifies an important commonality between thought and computation (*e.g.,* algorithms): both apply information but using differing levels of formality, flexibility, and automation.

Not all information needs require queries. If we have a helpful document describing the inverse document frequency on our laptop, we may simply consult it to review previously highlighted passages. This locating of items in an available source or across available sources through references and links is known as *navigation*, which is distinct from submitting a *query* to a system or person to find new sources. As another example of using navigation to satisfy an information need, in some case we may simply use the contents of available sources directly (*e.g.,* copying-and-pasting into an online form).

The process of analyzing a source and recording a map for use in retrieval is known as *indexing*. Consider Figure 1.2, where a book index provides a map for the book, so that a reader can quickly *navigate* to parts of the book discussing 'Terms,' for example. Contrast this *subject index* with the index used in a traditional term-based search engine,

Retrieve Information

R1 **Query** to request sources of information
R2 **Consult** and interpret available sources, *examining* and *navigating* within and across sources

Analyze Information

A1 **Annotate** sources with additional information, *e.g.*, notes, add formula locations
A2 **Index** sources by organizing them for retrieval

Synthesize Information

S1 **Apply** available information that we know, have in available sources, or is encoded in algorithms, etc.
S2 **Communicate** information by creating new sources

Figure 1.1: Information task taxonomy.

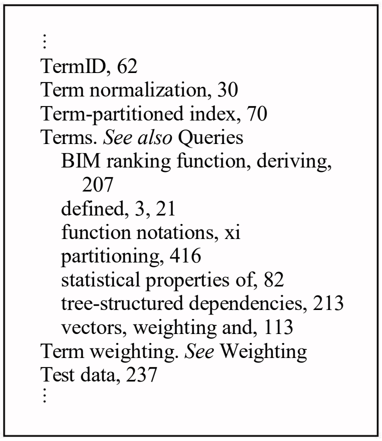

⋮
TermID, 62
Term normalization, 30
Term-partitioned index, 70
Terms. *See also* Queries
  BIM ranking function, deriving, 207
  defined, 3, 21
  function notations, xi
  partitioning, 416
  statistical properties of, 82
  tree-structured dependencies, 213
  vectors, weighting and, 113
Term weighting. *See* Weighting
Test data, 237
⋮

Figure 1.2: Excerpt from the index to "Introduction to Information Retrieval" by Manning *et al.* (2008).

which provide a much simpler map known as a *concordance* recording where specific terms appear in documents (Duncan, 2021). While these different indices are both used for retrieval, they differ in their scales (one document vs. a collection) and intended audiences (human reader vs. search algorithm). Other forms of indexing are less formal, such as collecting and organizing notes on different sticky notes for easier use.

As discussed earlier, we distinguish tasks for analyzing sources in terms of organizing them for use and retrieval (*indexing*), and adding information to sources (*annotation).* Annotations are often used in indexing sources, such as adding formula locations for PDF documents.

Source Jar Framework. To put sources and the tasks used to create them in a more intuitive relationship, Figure 1.3 visualizes our task framework as a jar of sources with a lid. The jar contains immediately available sources as marbles in the jar. Each marble has an identifying color and shape. The source marbles contain information of different types, and may refer to other sources inside and outside of the jar. Sources that are directly available are either with us, or inside the jar.

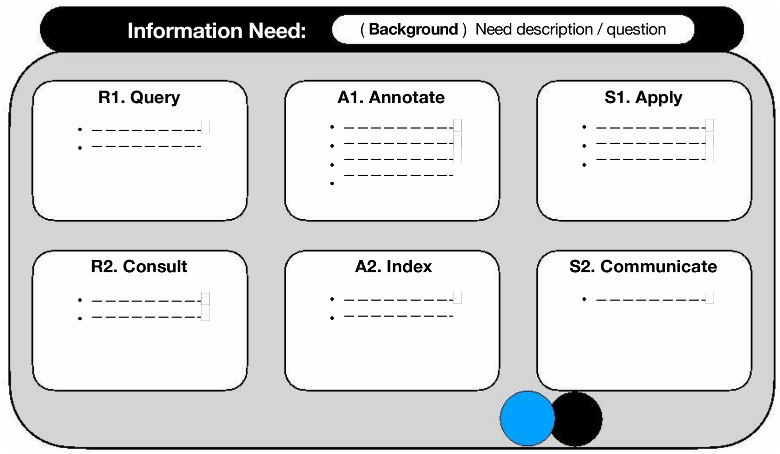


Figure 1.3: Information task framework: the source jar. The jar contains source 'marbles.' As we work we add, create, annotate and organize the sources in the jar, and record completed information tasks on the jar labels.

The jar lid is labeled with the background of the searcher, and the need that information is being retrieved, analyzed, and synthesized for. Stickers on the outside of the jar record information tasks that we perform to address this need. When we find or create a new information source, we add a marble to the jar.

If a new source *annotates* another source, we place it in a container with the source it describes inside the jar (*e.g.,* using a small plastic box). *indexing* produces a marble containing a description of which sources it organizes, and how. We take source marbles and containers out of the jar to use them, and return them to the jar when they are no longer useful. It is also possible to lose sources when the jar is accidentally left open and 'spilled.'[5] When we stop working to find and create sources for our information need, we select any sources we might wish to use, and then close the lid.

We can imagine having a shelf of these jars for different information needs. For a new information need we create a jar, adding any potentially useful initial sources to the jar (possibly from other jars). To reuse or get additional information for a need we worked on previously, we open a jar from our shelf.[6] Just as in the real world, not replacing sources in a jar runs the risk that we lose track of it, and have to pick up an old jar and work to *refind* that source, or find a replacement for the information in the lost source.

This informal jar model is intended to roughly capture how people experience working with information in a simple way. It captures observable sources and observable task actions. We tend to move from source to source, performing tasks of specific types with a goal in mind. We are often unaware of why we performed tasks in a particular order, and so this is not represented explicitly, other than as marbles moving in and out of the jar and notes for which information tasks were executed being written on the outside of the jar.

[5] *e.g.,* 'the dog ate it,' 'my internet is down,' or 'I know it's here...somewhere.'

[6] *e.g.,* 'Wait; I forgot one of the types of category theory applications I wanted to discuss in my paper from my notes...'

1.3 Information Needs and Information Task Strategies

When searching for mathematical information, what we need to find will vary from finding definitions for terminology, math symbols, formulas, operational knowledge such as proof techniques, applications of mathematics (*e.g.,* information retrieval models), resources for instructors, and detailed information on mathematical spaces, theorems, etc.

Example 1.1 illustrates information needs that different audiences may be seeking to address using the same query, along with a list of sources that might be used to address their needs. These needs vary from finding definitions to exploring sophisticated relationships between the generalization of the theorem in different mathematical structures (Hilbert spaces) and applications in other fields (quantum mechanics).

Example 1.1: Differing Information Needs

Query: What does $a^2 + b^2 = c^2$ represent and how is it useful?

Students might use this query to learn the Pythagoras theorem, and perhaps find an example demonstrating the theorem, and a possible proof.

Educators may have similar interests to students, but may seek additional resources on how to teach this result.

Researchers can have very different interests than the other audiences. They may be interested in one or more of the following:

- For a mathematician: Is this true in a general metric space and/or a Hilbert space?
- For a physicist: How is it linked to the probability assignments in quantum mechanics?
- For an IR expert: How is it related to probability assignments in a Hilbert space used in describing interaction for information retrieval?

As a result, the types of sources needed by each audience differ dramatically, but the initial (admittedly vague) query is identical: the *query intent* differs for these audiences. For math information needs, we have found it important to consider information needs both in terms of the desired information, as well as who is searching, and specificaly their mathematical background. Some places where relevant information might be found for these information needs include web-pages (MSE, Brilliant.org etc.), YouTube videos, online lecture slides, text documents and digital books (*e.g.,* OpenStax, LibreTexts), articles, and online notes (*e.g.,* MIT OpenCourseWare).

For a broader sense of the types of mathematical information needs users have online, Table 1.1 illustrates information needs for math organized by Broder's taxonomy of needs/intents behind web search queries (Broder, 2002). While some question the usefulness of the transactional class in Broder's model, for math, the transactional class is a useful distinction. For example, a user may be looking to *refind* a web page they used to enter formulas in LaTeX (i.e., a navigational intent). Or, they may instead be looking to find such a web page for the first time, thereby looking to interact/transact with as-yet unknown websites (i.e., a transactional intent).

Within the *informational* needs class, a distinct subclass of *computational* information needs exist. These include needs to evaluate or simplify a formula, or to produce a proof for a statement using logical operations. It was useful to distinguish questions that were seeking *concepts, proofs,* and *computation* for the ARQMath shared tasks (Mansouri *et al.*, 2022b) that we discuss in Section 3.

In our work we have found it useful to consider math information needs in two dimensions, based on the type of information need as shown in Table 1.1, and the user's mathematical background. More formally, we have a space/set of mathematical information needs N defined by a Cartesian product of possible information needs (T)ypes and user/audience (B)ackgrounds $(N \in T \times B)$. How these types and backgrounds interact is illustrated in Example 1.1.

Information task strategies. For a given information need, it helps to think about *strategies* that might be used to satisfy it. We can sketch these in strategy 'jar' diagrams as seen in the panel labeled Strategy 1. The diagrams identify an information need, initial queries, expected tasks, readily available sources, and a list of where other relevant sources

Table 1.1: Examples of mathematical information needs within Broder's Taxonomy (Broder, 2002). A user's math background is another dimension.

Navigational:	Find a specific source ('known item' retrieval)
	Web page (*e.g.,* for formula entry)
	Document (*e.g.,* Book, Technical Paper)
	YouTube or Khan Academy Video
	Podcast
Transactional:	Find online resources for use/interaction
	Formula entry
	Evaluating and plotting a formula
	Simplification of a formula
	Interactive theorem proving
Informational:	Find information for a topic or question
Sub-categories:	computation, concepts, and proofs
	How to compute an expression (*e.g.,* integral)
	Symbol and operation definitions (*e.g.,* ζ, $\binom{n}{k}$)
	Concept name(s) associated with a formula
	When is a function not differentiable?
	Who was Gauss?
	Proof drafts for P = NP

might be found. We can imagine beginning a new search by writing the information need on the lid, putting already available source 'marbles' in the jar, and then writing planned tasks on the jar labels. For readability, we use informal descriptions for the three main task types along with initial queries.

Let us first consider search strategies that might be used by undergraduate students, for learning how to complete a square, and to change the base of a logarithm (Strategy 1 and Strategy 2). In both examples, two queries that might be used are given, and the *Synthesis* tasks clarify the specific information need: the source they want to produce. In the first example, this involves completing an exercise on paper, and in the second example, obtaining a value from a calculator. Note that for the queries containing formulas, students might find it difficult or be unable to express the formulas in queries using a standard text query box, particularly if they are unfamiliar with LaTeX.

Strategy 1: (Student) Completing the Square

Retrieve:

Query: how to complete the square
OR $ax^2 + bx + c = (\star)^2 + \text{constant}$?

Search using the text query or possibly the symbolic query; $(\star)$ is a wildcard for any subexpression. Identify where the general method can be found, and examine the proof of the result.

Analyze: Mark-up/bookmark sources to identify useful information. Use a notebook to summarize key details found in sources. Save examples for different cases, *e.g.,* $a, b > 0$ and $c \leq 0$.

Synthesize: Solve an integration problem on paper, such as $\int \frac{1}{x^2 + 4x + 3}\, dx$.

Initial Sources: Textbook

Possible Sources: ChatGPT, YouTube, Prof. X?

Strategy 2: (Student) Log Base Change

Retrieve:

Query: log base change *OR* how to convert $\log_b x$ to $\log_c x$?

The student may use the text or symbolic query. Find sources giving the conversion rule with general bases.

Analyze: Markup sources and note down where relevant sources are located in a list (*e.g.,* in a text file). Save some special cases like converting $\log_{10} x$ to $\ln x$.

Synthesize: They use this to compute $\log_4 13$ on a calculator as the $\boxed{\text{log}}$ button on most calculators only represents $\log_{10}(\cdot)$.

Initial Sources: Web page on log conversion (hard to read)

Possible Sources: Somewhere online?

Now let's consider more advanced information needs for researchers. The researchers may be interested in following progress on an old conjecture (*e.g.,* Riemann Hypothesis). Or, they may be interested in learning about a new possible proof of the problem, or perhaps they were unfamiliar with the problem but are curious to know more about it. Strategy 3 seeks information and a proof for a problem that was posed in 1994. It became a major unresolved question in mathematical computer science until 2019, when Hao Huang solved it. Notice that our researcher is aware of *many* places where relevant sources for their

Strategy 3: (Researcher) Sensitivity Conjecture

Retrieve:

Query: What is **Sensitivity Conjecture**? Has it been proven?

Find papers/books defining the conjecture and providing proofs.

Analyze: Since the conjecture is very technical, retrieved material is annotated with sources where terminology in the conjecture can be comprehended. An index (graph) is made capturing the chronological account of progress on the proof.

Synthesize: Results and the methods for proving this conjecture are used for similar problems, and new articles/material are created to disseminate the findings.

Initial Sources: Email from colleague suggesting this might be relevant for my work.

Possible Sources: online encyclopedias (Wikipedia, Wolfram MathWorld), online Q&A sites (MathOverflow.net, AoPS, sciencedirect.com), YouTube videos, online lecture notes, text documents (*e.g.,* digital books, research articles), online science & math magazines (Quanta Magazine), online math databases (Cornell's mathematics library, zbMATH Open, $\mathcal{AMS}$: Math Reviews)

information need may be found in comparison with the undergraduate examples.

As another example, imagine that another researcher encounters a technical statement for the sensitivity conjecture, but which does not name it. They want to know the status of the statement, and if there are associated results they can use in their own work. Here the searcher only wants to learn the conjecture's name, properties, and proofs for later reference. The strategy from Strategy 3 needs to be altered, as reflected in Strategy 4. In this second case, the researcher has a document summarizing the key findings and where sources may be found.

A related challenge is that the interpretation of most mathematical expressions is *context-dependent*, i.e., the same formula may refer to different concepts in different contexts. For example, a student looking to understand the formula $\pi(m+n)$ using search will likely end up with multiple interpretations, which might represent:

- the distributive law: $\pi(m+n) = \pi m + \pi n$, or
- the value of the *prime-counting function* that counts the number of primes less than or equal to $m+n$.

Strategy 4: (Researcher) Unknown Conjecture

Retrieve:

Query: Any set H of $2^{n-1} + 1$ vertices of the $n-$cube contains a vertex with at least $\sqrt{n}$ neighbors in H.

The search is done using a textual query with LaTeX for the formulas. Related papers/books are collected and consulted for theorem definitions and proofs.

Analyze: Retrieved sources are annotated with links to other sources where terminology used can be comprehended. Highlight the name of the statement when it is found.

Synthesize: Create document summarizing the theorem name and key details, with cites/links to key sources found. Include link to a file directory on a laptop where additional notes in text and LaTeX files can be found, if any.

Initial Sources : Research paper with technical statement of interest

Possible Sources: online math databases (Cornell's mathematics library, zbMATH Open, AMS: Math Reviews), . . .

This property of a single object signifying multiple entities is known as *polysemy*, such as the word 'apple' being used to represent both a food and a company, and often poses challenges for both information retrieval and natural language processing.

User studies and use cases. There are a small number of papers examining math retrieval online. We know of just two studies looking at user behaviors in text-based search engines for math. The first was for the DLMF system (Miller, 2013), which supported text and formula search in a standard text box using queries in a LaTeX-based formula syntax. Few users at the time visited the site intending to search using formulas, most likely because of its novelty, and because this capability wasn't prominently featured on the site. What math queries were used were often short, or even single symbols. There also tended to be fewer click-throughs to pages from search results, and more query reformulation for formula queries; whether users were browsing formula search results for interest, had challenges satisfying information needs, or some combination of these is unclear.

The second log study was for a standard text-based search engine (Mansouri *et al.*, 2019b). Query logs from a Persian general-purpose search engine were used. Compared to the general case, search sessions for math topics were typically longer with more query refinements (*i.e.,* changing queries to try and improve results) and were less successful. In contrast to the DLMF study, queries were also longer and more varied more than queries overall. This was partly because many math queries appeared to be questions copy-and-pasted from exercises or homework assignments.

In another interesting study, posts to threads in an online math Community Question Answering (CQA) site were studied (MathOverflow[7]). The authors identified patterns in the collaborative actions they exhibit (*e.g.,* providing information, clarifying a question, revising an answer) and their impact on the final solution quality (Tausczik *et al.*, 2014).

Earlier work considered use cases for math-aware search in a study of mathematics graduate students and faculty (Zhao *et al.*, 2008). Surprisingly the participants did not find formula search was useful overall, perhaps because they generally knew the names of entities they wanted to search on. The study also points out that the type of a source is an important relevance factor (*e.g.,* exercises vs. code). Another analysis of expert use cases is also available (Kohlhase and Kohlhase, 2007), in which formula search was studied using the MathWebSearch tool.

1.4 Retrieval Systems

Figure 1.4 provides an overview of retrieval system interactions with people, and the specific sub-tasks from the 'jar' framework that they perform. Unlike the freely interacting tasks of the 'jar' model, retrieval systems generally perform information tasks in a fixed order, shown by arrows in Figure 1.4. The figure has two main information flows for the collection of sources that a retrieval system uses.

1. **Index construction (offline).** Information passes from the sources at top and flows to the bottom-right, as sources are annotated with additional information, and then used to compile a searchable index of patterns. The collection index is precomputed before the system is used for retrieval.

[7]https://mathoverflow.net

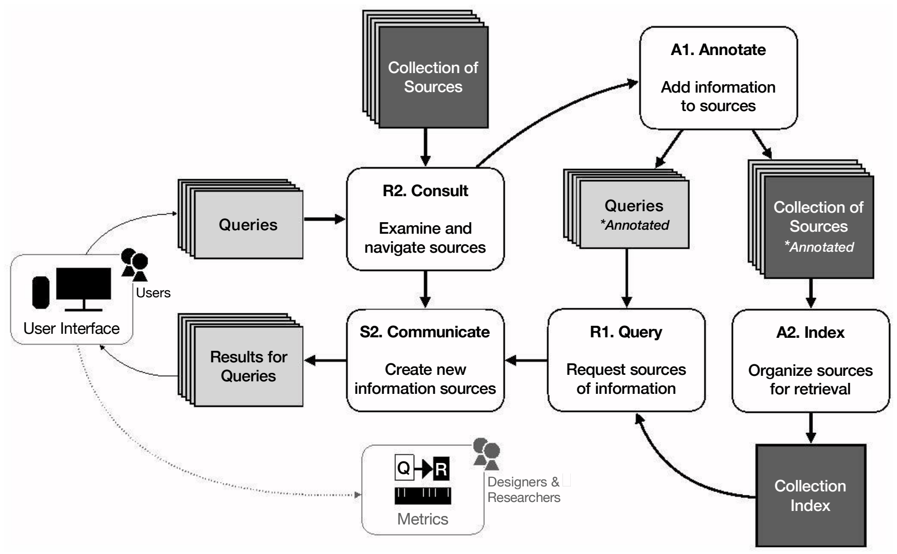


Figure 1.4: Information tasks in retrieval systems (backend). Arrows show the flow of information. All tasks in Figure 1.3 other than *Apply* are shown.

2. **Retrieval (online).** Submitted queries are annotated and then matched against patterns in the index, returning one or more matching sources. The collection is generally consulted for passages, bibliographic data, and other contents when generating the result returned to the user.

Consulting sources. Search engines that match queries to contents in sources are a type of filter. A standard search result is useful precisely because it contains sources with patterns of information shared with the query, omitting all other sources.

The implementation of *consult* tasks that access sources is important for both index construction and retrieval, and is another way that sources are filtered in a retrieval system. Source contents shown in search results directly impact our impression of which returned sources are promising. Source contents used for index construction define the available patterns for matching queries to sources.

For example, omitting high frequency terms from queries and sources that do not signify a topic (*i.e.,* skipping *stop words* such as ‘the’) can greatly reduce index sizes and increase retrieval speed, but at the risk of performing poorly on queries using these terms; a classic example is the phrase ‘to be or not to be’ from Shakespeare’s play Hamlet, which is instantly recognizable but composed entirely of stop words.

For math-aware search, a similar decision would be omitting tokens and strings representing formulas (*e.g.,* in LaTeX source files). Limitations on what can be consulted includes formulas in PDF documents, which are usually not represented explicitly (Shah *et al.*, 2021). This and other missing information can be addressed by annotating sources.

Annotation and indexing. In addition to selecting source contents in the *consult* step, we will also *annotate* sources with additional information. This extra information can be used to add patterns for matching sources in the index, or to add information to retrieval results.

For example, some neural net-based techniques such as SPLADE automatically add words that do not appear in a source to the inverted index (Formal *et al.*, 2021).[8] These additional terms are synonyms and other words appearing in similar contexts within a training collection. For math, a simple example is adding additional representations for formulas in sources, such as generating Content MathML for operator trees corresponding to formulas represented in LaTeX or Presentation MathML, allowing formulas to be searched using both formula appearance and operation structure.

From the information obtained through *consulting* and *annotating* documents, an index of patterns for matching queries is produced. This can take different forms, but is generally one or a combination of:

1. *inverted indexes* that map patterns to sources and source locations (*e.g.,* tokens or paths in graphs for math formulas), and
2. *embedding spaces* mapping patterns to points in a vector space, where entities with more similar contexts across a collection are closer (*e.g.,* words, sentences, and formulas).

[8]This augmentation is also applied to queries. Query annotations are called a *query expansion* when they add tokens or other patterns for matching additional sources in the collection.

Embedding vectors have their own dictionary mapping vectors to specific sources or source locations (*e.g.,* when search is done on text *passages* or individual math formulas). This allows sources matched in vectors to be consulted when communicating results to users.

Retrieval: Querying sources and communicating system results. We query a collection using the collection index and an annotated query containing additional terms and/or an embedding vector. Search using inverted indexes is referred to as *sparse* retrieval, while search using embedding spaces is referred to as *dense retrieval*, based on the underlying vector representations for each. In particular, term vectors representing the presence of words or formula structures in a document are mostly zeros. In general, sparse retrieval models such as BM25 that use tokens or other source contents directly for lookup are faster (Robertson and Zaragoza, 2009), but dense retrieval models such as ColBERT (Khattab and Zaharia, 2020) are more effective (Wang *et al.*, 2023; Giacalone *et al.*, 2024). Some retrieval models use dense models to improve sparse models *e.g.,* SPLADE, mentioned earlier.

The improved effectiveness of dense retrieval models is partly from additional context used in defining patterns, *e.g.,* using the words referring to and surrounding a formula to represent a formula in a pattern vs. the formula alone. The use of a vector space also provides more holistic and flexible pattern matching, *e.g.,* finding source vectors with the most similar angles to a query vector, rather than matching query formula tokens individually to vocabulary entries in an inverted index. These help bridge the *vocabulary problem* discussed in the next subsection.

How the final result of a query is *communicated* (generated) can vary substantially, and often makes use of query and source annotations. In a traditional search engine, specific sources are matched in the index for the *query* task, with the index comprised of some combination of inverted indexes and embedding spaces. Source contents are then used to generate a result in the *communicate* task, using sources and source locations matched in the index.

However, for a generative question answering or retrieval system, the result of the *query* task may be a single vector capturing the similarity of patterns in the query to patterns within sources of a very large collection produced using a neural network. This vector is then used in the *communicate* task as a starting point for generating the response, for example using a second recurrent neural network trained on the collection, possibly along with additional information from the original collection of sources (*e.g.,* with references to specific sources). Some are used to generate a list of ranked sources directly (Zeng *et al.*, 2024), ultimately producing an *extractive* search result summary based on source contents.

Other recent systems such as Google's AI search assistant produce *abstractive* summaries of retrieved sources, which summarize matching sources but without limiting the summary to contents found in the matched sources or their annotations.

System design. System designers and IR researchers are interested in the efficiency and effectiveness of a retrieval system. As seen in Figure 1.4, these are observed in live systems through query and user interaction logs. For experiments, system results are computed using simulated user interactions for a fixed set of queries, and relevance scores for sources, along with a description of the information needs associated with each query. Designers and researchers also make use of additional tools for evaluation, some of which we discuss in Section 3.

1.5 User Interfaces and System Interactions

User interfaces play a very important role in mathematical information retrieval. In addition to executing queries and returning results, how queries are entered, how results are returned, and how other information tasks in Figure 1.3 are supported can help speed up or even limit a person's efforts to find and use information.

We next present a user-centered view of retrieval systems in math information tasks. We then share some key challenges for retrieval system interaction, along with interface designs aiming to address them.

Interfaces-in-the-task-loop. Figure 1.5 illustrates a student working to change the base of a logarithm (*i.e.,* Strategy 2) using multiple retrieval systems. At the bottom-left of Figure 1.5 is a jar holding

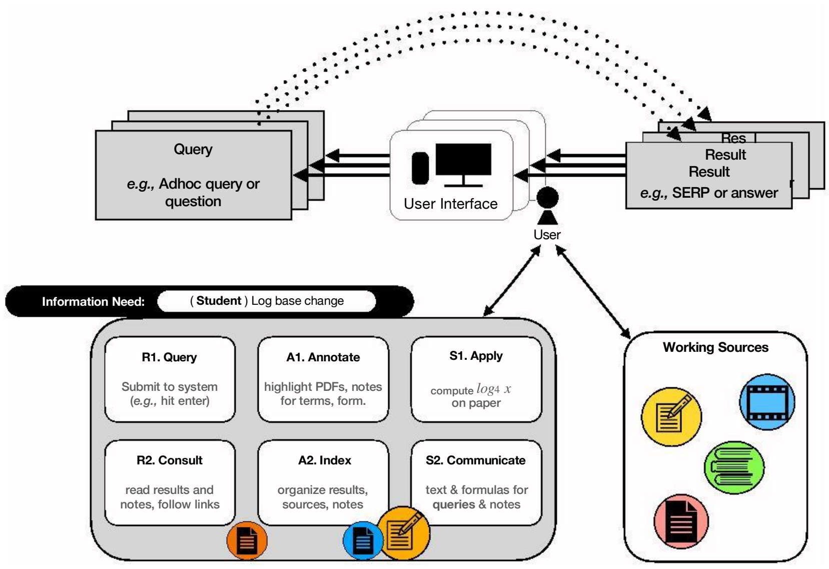

Query
e.g., Adhoc query or question
User Interface
User
Res
Result
Result
e.g., SERP or answer
Information Need:
(Student) Log base change
R1. Query
Submit to system (e.g., hit enter)
A1. Annotate
highlight PDFs, notes for terms, form.
S1. Apply
compute $log_4 x$ on paper
R2. Consult
read results and notes, follow links
A2. Index
organize results, sources, notes
S2. Communicate
text & formulas for queries & notes
Working Sources

Figure 1.5: Interacting with multiple retrieval systems (frontend). Each dotted arrow represents a retrieval system backend (see Figure 1.4). Sources currently used to address the information need are shown in a separate container at bottom right.

sources the student had on hand when they started searching, new sources they find or create, along with other linked sources, *e.g.,* by web link, citation, or mention. In addition to these sources, their queries, results from queries, and handwritten notes (*e.g.,* from converting bases by hand) are also found in the jar. Of these available sources, the ones currently being used are at the bottom-right of Figure 1.5.

Some selected sources being worked with partially or fully answer the student's needs, but others do not, such as sources later deemed not relevant. Other selected sources might exercise knowledge such as shown for the *Apply* task in Figure 1.5, or come from other tasks such as annotating and indexing sources of interest. Some selected sources may be added even after finding answers, perhaps because they provide a different perspective, or have a presentation that is easier to understand.

For more complex tasks, we often see the focus of our selected sources drift. In the *berry picking* model of retrieval (Bates, 1989), people see their queries and information needs change as they search and learn. Particularly for unfamiliar topics, our needs and queries may change dramatically as our understanding does (Belkin, 1980). For example, this is likely to happen when a person explores unfamiliar concepts associated with unfamiliar notation. In our jar model, information need changes involve changing the jar lid label, perhaps using an orange sticky note placed over the original description.[9]

What we have in Figure 1.5 is a person generating, selecting. and using sources for needs that may change as they work. Retrieval systems are a part of this process, but not the focus.

Interaction challenges. All systems embody design decisions and biases. Naturally, no one retrieval system will be ideal for all queries or subjects. However, users often have challenges in search that are more cognitive than system-related. These are important considerations in creating usable systems, particularly for search interfaces (Hearst, 2009; White, 2016; Holmes *et al.*, 2019).

Norman identifies two ‘gulfs’ that limit human task performance (Norman, 1988). Broadly speaking, for retrieval systems the two main categories of interaction challenges are with expressing queries (a *gulf of execution*) and interpreting search results (a *gulf of evaluation*). In both cases, for unfamiliar topics, the user may be unable to formulate an effective query or interpret results reliably precisely because of what they do not know, or because their understanding is incorrect (*i.e.,* their *Anomalous State of Knowledge*, Belkin, 1980).

A common cause of a gulf of execution in query formulation is the *vocabulary problem*, where the terms/patterns a person uses for search differ from those used to index sources. For example, in one study undergraduate students were challenged while trying to define the binomial coefficient ‘$\binom{n}{k}$’ (Wangari *et al.*, 2014). Because of this notation-based vocabulary problem, the students’ were unable to find a definition using standard text search. When allowed to enter the expression by hand with automatic translation to LaTeX, they found definitions using the same text search engines.

People sometimes also encounter a *gulf of evaluation* when trying to identify relevant information in search results. Aside from missing

[9]Sticky notes: a versatile information tool in this section and in life.

relevant items in results due to the vocabulary problem, an important factor here is how retrieval results are presented to a user. For example, Reichenbach *et al.* (2014) report statistically significant differences in the ability of participants to identify relevant sources in SERPs when excerpts present formulas as raw LaTeX vs. rendered formulas. Additional gulfs occur when selected excerpts are not relevant for a need, or a person lacks the math background to understand a result.[10]

Learning new terminology and notation while searching allows a user to extend their patterns used to express queries and identify relevant results, bridging these gulfs of execution and evaluation. Some of these new patterns might be recorded explicitly in a source, *e.g.,* recording an unfamiliar notation for eigenvectors on a blue sticky note.

Query input: math-aware search bars. For the most part, math-aware search bars differ in how they include formulas. Perhaps the simplest design is for users to enter *both* text terms and formulas as text. An early example is the Digital Library of Mathematical Functions[11] which accepts LaTeX commands for formulas along with text terms in queries (Miller and Youssef, 2003). The more recent Approach Zero system[12] system uses MathQuill[13] to render LaTeX formulas as they are typed in the search bar, and allows writing lines and argument positions to be reached with arrow keys rather than LaTeX commands (*e.g.,* for superscripts and fraction denominators).

To avoid remembering many names for operations and symbols, or to avoid unfamiliar LaTeX or other syntax for creating formulas, usually a palette of buttons with images for symbols and operations accompanies the search bar. Buttons add formula elements including operation structures (*e.g.,* fractions, integrals, and radicals) and symbols not found on a keyboard (*e.g.,* greek letters such as ζ (*zeta*)). Query bars with palettes often display formulas in a structured editor like those in document editors (*e.g.,* Word). Early examples of prototypes with symbol/operation palettes include MathWebSearch (Kohlhase and Prodescu, 2013) and MIAS (Sojka *et al.*, 2018).

[10]Impatience, inattention, mental strain, and tiredness are also factors here.
[11]https://dlmf.nist.gov
[12]https://approach0.xyz
[13]http://mathquill.com

As another way to reduce the effort and expertise required for formula entry, some search bars also support *multimodal* formula entry. Multimodal query editors allow formulas to be uploaded from images or entered using handwriting in addition to standard keyboard and mouse-based entry. There are also multimodal tools such as Detexify,[14] which looks up LaTeX commands for symbols drawn using a tablet or mouse (Kirsch, 2010). In addition to search, recognizing math in handwriting and images has been used for interactive computer algebra systems and other applications, and is an active area of research dating back to the 1960's (Zanibbi and Blostein, 2012; Truong *et al.*, 2024).

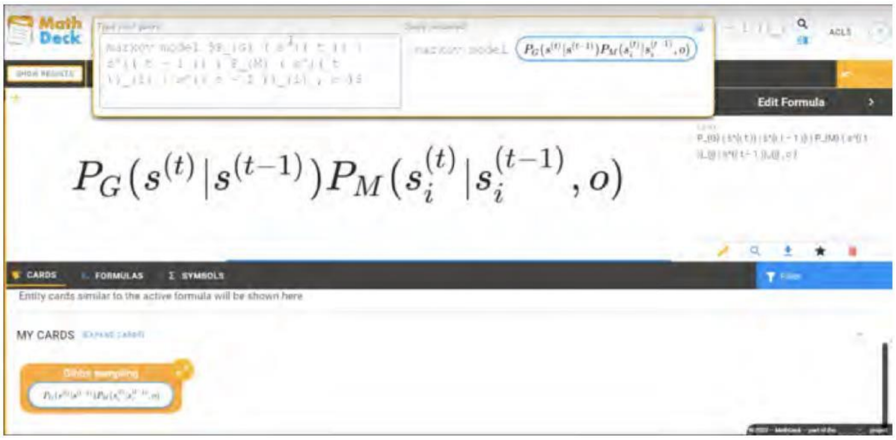
An example of a search bar with multimodal formula entry is the MathDeck system[15] (Diaz *et al.*, 2021). As seen at the top-left in Figure 1.6, a text search box can be used to enter words and LaTeX for formulas. Formulas can also be added from a visual formula editor shown at the center-left in Figure 1.6, and using formula 'chips' with embedded LaTeX (*e.g.,* blue oval at right of the query text box). Like MathQuill and structured formula editors, MathDeck renders a formula as it is entered, but with more flexible subexpression selection and entry. MathDeck's query and formula entry interface is designed to:

1. support text entry; natural for text, and one can easily type 'x + 2', or copy-and-paste LaTeX with small changes (*e.g.,* $a \rightarrow x$)
2. provide symbol palettes to help enter symbols and structures
3. provide handwriting input for those who prefer it, and to avoid searching palettes for symbols & structures
4. support formula reuse in chips; chips can be used in editing, and can be exported/shared as images with LaTeX metadata
5. construct formulas interactively using a structured editor, with larger formulas easily built up from smaller pieces.

Other multimodal query entry interfaces have similar design goals, most commonly to support image and keyboard/mouse input.

Other familiar ways to reduce query and formula entry effort are query suggestions and query autocompletion. Their helpfulness is related

[14]http://detexify.kirelabs.org/classify.html
[15]https://mathdeck.org



$$P_G(s^{(t)}|s^{(t-1)})P_M(s_i^{(t)}|s_i^{(t-1)},o)$$

Figure 1.6: MathDeck query entry, formula chips, and cards (Diaz *et al.*, 2021). Chips can be dragged, edited, and combined. Editing may be done using raw LaTeX, or a combination of operations, chips, handwriting, and LaTeX using the canvas at center. Formula cards (bottom left) contain chips, titles and descriptions. New cards can be created by users, and searched by formula & title (video: https://www.youtube.com/watch?v=XfXQhwIQlbc).

to the principle of *recognition over recall*: it is usually easier to recognize something we see than recreate the same thing from memory (Hearst, 2009). As a simple example, a query autocompletion might include a concept whose name but not formula we can remember, and allow us to quickly select a query containing both.

To illustrate the communication of retrieval results, we will use a system for visual search that uses an inverted index. Figure 1.7 shows handwriting in math lecture videos being queried with a LaTeX-generated formula image. The inverted index uses pairs of symbols (*e.g.*, (I, n), $(=, A)$) as the vocabulary for lookup. Before searching, the query image is annotated with a graph containing nodes for symbols and edges with angles between adjacent symbols.

The inverted index is queried by looking up *all* adjacent query symbol pairs, to find their occurrences in the video collection. Each entry in the *posting list* for a queried symbol pair (*e.g.*, (I, n)) refers to an edge connecting the same symbols drawn in a video. Before indexing, videos are annotated with keyframes of drawn symbols that overlap in

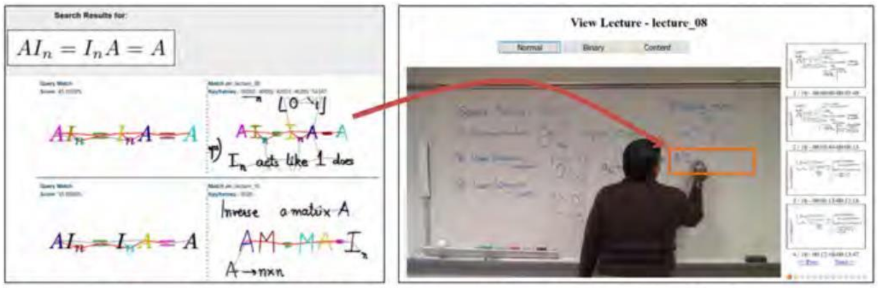


Figure 1.7: Tangent-V Formula Search Results (left) and Video Player Supporting Navigation (right) (Davila and Zanibbi, 2018). A rendered LaTeX formula is used to search handwritten math symbols recognized in a video (Davila and Zanibbi, 2017b). Here the user has clicked on the '=' of a matched formula on the whiteboard, and this advances the video to where it is first drawn (video: https://www.youtube.com/watch?v=gn24qo1MLN0).

time along with an adjacency graph for each keyframe. Each edge in a keyframe graph is added to the posting list for its pair of symbols, as a *posting* containing a unique identifier for the edge, its keyframe graph, and video. In Figure 1.7, zoomed-in video keyframe graphs are shown in the results on the left, and keyframe thumbnails are shown at the far right.

Query response: communicating results. It is actually the drawn symbol keyframes annotated on videos that are searched. Keyframes are scored by the similarity of matched adjacency subgraphs to the query graph, based on the similarity of matched symbols (nodes) and their angles (edges). In the results shown in Figure 1.7, symbols matched in a query/keyframe have the same color, and matched graph edges are red. To avoid missing symbols due to recognition errors, symbol pairs are indexed using all combinations of possible labels for each symbol. This is how n matches M in the second match shown.[16]

Let's consider the results in Figure 1.7 more closely, with associated system tasks in Figure 1.4. The query result shows the top-2 matching

[16]A variation of adding tokens to queries and documents to increase possible matches in an inverted index. Symbol similarity is computed from all label probabilities assigned to each symbol. Tangent-v has also been used to search formula collections using unique symbol labels in PDF (Davila *et al.*, 2019).

videos, and not keyframes. To generate this view, the keyframe ranking from *querying* the index is restructured as a video ranking, with videos ranked by best keyframe match. Also, the annotated query and videos have been *consulted* to produce the graph matches shown for each video. The videos are consulted again for annotations not used for indexing: clicking the mouse on a symbol in a result keyframe makes the video player jump to where the symbol starts to be drawn.

How the search results filter and present the videos is motivated by tasks users carry out (see Figure 1.5). For example, having symbols linked to frames can help people *consult* videos by quickly navigating to where a formula is drawn and discussed. Results rank videos rather than keyframes to make the search results more concise and easier to consult. The *communication*, *annotation*, *indexing*, and *querying* tasks can also be supported from search results. In MathDeck formulas in results can be used directly for search, selected for editing or export, or annotated in a card with a title and description. Cards are also automatically indexed in a ‘deck’ searchable by formula or title.

Showing matching graphs in results is more helpful for designers than users; simple bolding or highlighting is more common. In contrast, MathDeck’s search results highlight matched query words and formulas located in PDF documents (*e.g.,* for papers from the ACL Anthology, Amador *et al.*, 2023).[17]

We’ve used just two systems here to illustrate search results that rank sources, and how they interact with human information tasks. However, results from other systems have different types. Some systems plot, simplify, and/or perform requested operations on formulas, or provide solutions for math problems posed in text and/or fomulas directly (*e.g.,* using WolframAlpha or a math-aware chatbot). In these cases the response is an answer to a (possibly inferred) question, rather than a ranked list of sources. These are designated as *question answering* systems, and interactions with chatbots addressing math queries are a type of *conversational search* where clarifying questions and additional information may be provided or received in multiple rounds of query/result interactions.

[17]ACL Anthology search demonstration: https://drive.google.com/file/d/1fbiMyHtlfEYUJvmrbZsWzhfL0X_zo9-t/view.

Chatbots using Large Language Models (*LLMs*) have proven intriguing and useful in some instances, but there is an increasing awareness of issues related to the validity of responses and other substantive concerns (Bender *et al.*, 2021). However, any retrieval result is only an information source – understanding and verifying any source requires additional work. Related to this, in Community Question Answering platforms (CQAs), many posts request clarification of a question, or clarify/correct posted answers and comments.[18] This illustrates how human responses to math queries also often contain misunderstandings, ambiguities and errors.

Regardless of the result type, how information is chosen and presented in results is important. It has a real impact on the usefulness of the result as a source of information, and on how tasks other than consulting the result itself are supported. In many cases usability testing can be used to check the effectiveness of result presentations and other interface design elements, and to discover refinements and alternatives.[19]

Supporting tasks for individual sources. Programs used to view/consult sources can also help with the user tasks illustrated in Figure 1.5. A nice example is the ScholarPhi system shown in Figure 1.8 (Head *et al.*, 2021). Reading formulas can be challenging, as symbols may be defined throughout a paper. ScholarPhi provides annotations decorating a selected formula/subexpression, providing symbol definitions in-place. Definitions are linked to where they appear, and text not associated with a selection is greyed out.

To produce the definition views in ScholarPhi, sources need to be annotated with formulas, symbols, and definition locations, and then definitions need to be linked with associated entities where they appear in the paper (*i.e.,* symbols or subexpressions). Definition segmentation and linking entities are performed with natural language processing techniques. The original prototype identified math symbols within LaTeX source files used to generate PDFs, simplifying formula detection.

[18] Including the classic, ‘my correction to @your correction.’

[19] *e.g.,* parts of the MathDeck were usability tested (Dmello, 2019; Nishizawa, 2020; Diaz, 2021), which led to substantial improvements.

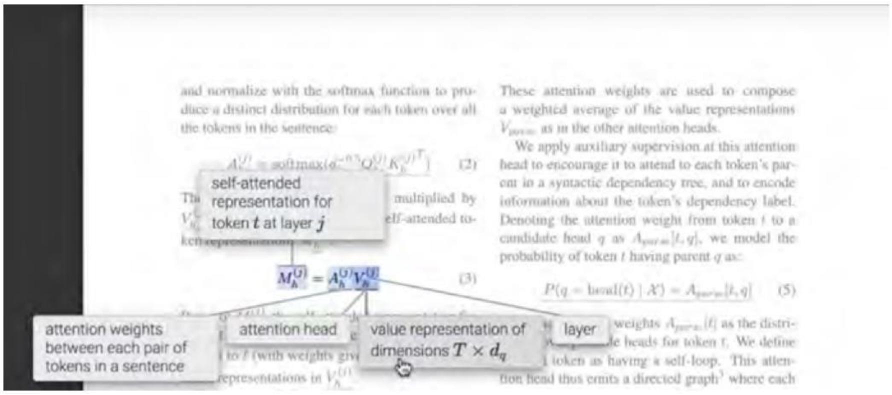

self-attended representation for token t at layer j
$M_h^{(j)} = A_h^{(j)} V_h^{(j)}$
attention weights between each pair of tokens in a sentence
attention head
value representation of dimensions $T \times d_q$
layer

Figure 1.8: ScholarPhi system showing definition for math symbols found within the same PDF paper (Head *et al.*, 2021). To assist skimming for details, text other than for definitions of a selected formula is greyed out (video: https://www.youtube.com/watch?v=yYcQf-Yq8B0).

A second example is the keyframe list at right in Figure 1.7. When viewing a video, all keyframes for handwritten content are available in a thumbnail list. Keyframes can be selected, and individual symbols clicked on to jump to where it is first drawn in the video (similar to the search results). This requires annotating video sources with generated keyframes produced using computer vision techniques.

Both ScholarPhi and Tangent-v require generating additional information using automated inference (*i.e.,* AI), and their usefulness is limited by the accuracy and scalability of the methods employed. However, we believe that this is an important future direction for mathematical information retrieval, because the content and organization of mathematical sources can be complex. Particularly for non-expert users, mature versions of these techniques may be very helpful.

A number of well-known formats were devised or augmented to support detailed annotation with links and tags, including TIFF, PDF, and XML. Unfortunately, detailed 'semantic' annotation has proven difficult at scale despite significant efforts. Some possible reasons include the time required to create sources before annotations, the diversity of information needs (*e.g.,* which information do we annotate?), attaching large annotations makes files large and unwieldy, and overall progress in scalable AI has been slower than many anticipated.

As AI continues to improve, creating source annotations to support examining and navigating math sources and other user information tasks within UIs seems likely to be beneficial. Perhaps application-specific annotations such as those used in ScholarPhi, Tangent-v, and MathDeck are a good starting point.

2

Annotating and Indexing Sources

In this section we focus on indexing formulas and text in sources, as illustrated in Figure 1.4. This requires consulting sources, annotating formulas and text with additional information, and then constructing 'sparse' inverted indexes for looking up discrete patterns directly (*e.g.,* words or paths in formula trees), and/or 'dense' indexes representing the same patterns in embedding vectors, which are searched based on the similarity of an embedded query pattern with items in the index (*e.g.,* using vector angles). With this data, a variety of sparse and dense retrieval frameworks can be used for a variety of search, question answering, machine learning, and evaluation tasks.

We will consider each of the three main tasks needed for indexing in turn, starting with the types of information and representations used in mathematical information sources.

2.1 Consulting Sources for Mathematical Information

Let's first consider how we consult sources for mathematical information using some examples. The examples come from technical documents and search queries, but much of what will be said applies equally well to videos, audio, conversations, text messages, and other source types.

We will start with a definition of the *Inverse Document Frequency (IDF)* shown in Example 2.1. IDF is used in a number of influential sparse retrieval models including variants of TF-IDF (*Term Frequency*-Inverse Document Frequency) and BM25 (Robertson and Walker, 1994; Robertson and Zaragoza, 2009). Its utility comes from a simple but profound insight: query terms appearing in fewer documents are rarer and thus *more specific*, and so should be given higher weight when ranking documents using matched query terms (Jones, 1972).

Example 2.1: Inverse Document Frequency (*IDF*)

Excerpt from Robertson (2004)

...Assume there are N documents in the collection, and that term t_i occurs in n_i of them ...the measure proposed by Sparck Jones, as a weight to be applied to term t_i, is essentially

$$idf(t_i) = \log \frac{N}{n_i} \qquad (1)$$

Variable and function definitions

...Assume there are N documents in the collection, and that term t_i occurs in n_i of them ...the measure proposed by Sparck Jones, as a weight to be applied to term t_i, is essentially

$$idf(t_i) = \log \frac{N}{n_i} \qquad (1)$$

For example, the term 'BM25' is predominantly found in sources on information retrieval, while the term 'weight' is used for many topics and in multiple senses, including the heaviness of an object and scaling numeric values. When scoring documents against the query 'BM25 weight', matches for 'BM25' will have higher IDF scores than 'weight,' reflecting the narrower usage of 'BM25.'

Despite its brevity, the excerpt in Example 2.1 contains a fair amount of directly represented and implied information. Recovering some of this information requires pattern matching and inference, *i.e.,* applying information known, found in the passage, or found in other sources. To illustrate this, we annotated the definition at the bottom of Example 2.1. Underlining and highlighting are used to associate variables and function names with their definitions in the text. ‘them’ is placed in a box with a thin outline to represent the *anaphoric* (backward) reference from ‘them’ to ‘N documents.’ Knowing that ‘them’ refers to documents, we can infer that n_i is the number of documents containing term t_i.

In this way, gathering information from a source involves a combination of *consulting* the source to identify stated information, and *analyzing* the source to reveal additional information from explicit and implied linguistic patterns and relationships. Both activities are informed by available knowledge, *i.e.,* readily available and actionable information that we have previously seen or inferred, or find by *navigating* to other sources or *querying* for new sources (*e.g.,* following hyperlinks, or asking a person).

Example 2.2: Information extracted from definition in Example 2.1

Variables: placeholders for a set of values, similar to common nouns

- The text identifies N as the number of documents in a collection. N is like a common noun, because the collection is not specified.
- The text defines t_i as any term appeari~~ng~~ n_i ~~times in a collection~~, with shared identifier i, *e.g.,* (t_3, n_3) could be `(‘weight’, 11)`.

Functions & Operators: create new from given values, like verbs

- log: log function with unspecified base.
- $idf(t_i)$: aside from the ~~unspecified log~~-base, a concrete function in Equation (1). The text says this gives a weight for term t_i.
- Division $(\frac{\cdot}{\cdot})$, application $(idf(\cdot), \log \cdot)$, and equivalence $(=)$ appearing in Equation (1) that are not defined in the excerpt.

Additional context:

- The text indicates Spärck Jones introduced the *idf* formula in a different, unspecified form (Jones, 1972).

Following this process, we gathered the information shown in Example 2.2. Note that the underlined missing details are *deliberate* omissions:

- By not specifying a collection, N is defined for any collection.
- t_i and n_i give any term and a count for documents containing it.

- Omitting the logarithm base emphasizes that logarithms increase with input size (*i.e.,* they are *monotonic*) and so any base suffices.
- Function application and the operators used are common, and their definitions are assumed to save space and reader effort.

These omissions are helpful, provided the reader can infer the missing details they need: the definition would be longer and harder to read otherwise. More generally, what an author chooses to omit is informed by (1) the context and focus of discussion, *i.e.,* items discussed earlier and the current topic, and (2) the assumed background knowledge of the audience. These determine what can and *should* be left out.

Note also that Equation (1) on its own conveys only partial information. A formula presents a hierarchy of operations, but its symbol definitions and its *purpose* generally come from surrounding text, other formulas, and assumed knowledge. In this example, the text defines all variables and the term weighting role of the *idf* formula, but not the operations shown in the formula.

Let's next consider an alternative definition for the *idf* function from Equation (1) that uses no mathematical symbols:

> A term's *inverse document frequency* is the percentage of documents containing the term, inverted and then log-scaled.

This seems simple enough – we invert the numerator and denominator in the percentage of documents containing the term (*i.e.,* 'flip' the fraction) and convert this value to a logarithm.[1] But we lose some useful patterns and information when we do not use math notation:

Visibility: Formulas are italicized and use distinct symbols, making them easier to find in sentences (*inline*). Also, they may be offset from the main text and indented (*displayed* like Equation (1)).

Compact Reference: Referring to symbol names is more efficient than reusing descriptions, *e.g.,* N vs. 'the number of documents.'

Compact Structure: Text describes relationships, while formulas *show* relationships spatially, *e.g.,* Equation (1) vs. the textual definition above. A good example is the *distributive property* from algebra, which is easily expressed using: $x(y+z) = xy + xz$.

[1]The log scale reduces the rate at which *idf* increases, which avoids having rare terms completely dominate rank scores.

Abstraction: Formulas often define properties and patterns applicable in multiple contexts. For example, *idf* can be applied to formulas if t_i is redefined to refer to a unique formula, and n_i the number of documents where the formula appears. We can also redefine N_0 and λ in the decay function $N(t) = N_0 e_{-\lambda t}$ to estimate decreases in (1) a retirement fund balance, (2) the rate of a chemical reaction, or (3) the potential contained in a capacitor.

Judicious use of formulas in technical writing makes mathematical information easier to find, analyze, and reuse/adapt. Which things are beneficial to formalize in notation again depends upon a document's focus and intended audience.

Another *idf* definition and visual formula representations. The following is an alternative definition for IDF taken from the Wikipedia page for the *tf-idf* retrieval model.

> *The* **inverse document frequency** *is a measure of how much information the word provides, i.e., how common or rate it is across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word (obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient):*

$$idf(t, D) = \log \frac{N}{|\{d : d \in D \textit{ and } t \in d\}|}$$

This definition for the *idf* formula is essentially equivalent to that in Example 2.1. However, the formula used to represent *idf* has changed. The document collection is given explicitly as a parameter D, and the number of documents containing a term, previously n_i, is now the set size for documents containing the term. The term itself is noted as simply t, and not t_i. Also, the wiki page includes a link to the *Logarithmic scale* article that we can follow to review that concept.

Example 2.3 provides an excerpt for the embedded formula in the *tf-idf* article HTML page. The formula appearance is represented using

Example 2.3: Presentation MathML from Wikipedia *tf-idf* article

```
<math xmlns="http://www.w3.org/1998/Math/MathML"
    alttext="{\displaystyle \mathrm {idf}...">
  <semantics>
    <mrow class="MJX-TeXAtom-ORD">
      <mstyle displaystyle="true" scriptlevel="0">
        <mrow class="MJX-TeXAtom-ORD">
          <mi>i</mi>
          <mi>d</mi>
          <mi>f</mi>
        </mrow>
        <mo>(</mo>
        <mi>t</mi>
        <mo>,</mo>
        <mi>D</mi>
        <mo>)</mo>
    ...
    </mrow>
    <annotation encoding="application/x-tex">
      {\displaystyle \mathrm {idf} (t,D)=
        \log {\frac {N}{|\{d:d\in D{\text{and}}t\in d\}|}}}
    </annotation>
  </semantics>
</math>
```

Source: https://en.wikipedia.org/wiki/Tf-idf?oldid=1236851603.

Presentation MathML, which is an XML encoding for the placement of symbols on writing lines along with their types (*e.g.*, `<mrow>` for writing lines and token groupings of characters (*idf*), operators in `<mo>`, variable and function identifiers in `<mi>`). The LaTeX string used to generate the MathML using MathJax[2] is included in the outermost `<math>` tag, and in the `<annotation>` tag near the bottom of of the excerpt.[3] Many web browsers can render Presentation MathML directly, or can use javascript-based LaTeX rendering tools, *e.g.*, applying MathJax to the `alttext` attribute of the `<math>` tag.

The vast majority of documents represent formulas by their appearance, whether as raster images (*e.g.*, pixel-based PNGs), vector images

[2]https://www.mathjax.org.

[3]The `<semantics>` tag can be misleading: math operations are not represented.

(*e.g.,* SVGs with drawing instructions), LaTeX, or Presentation MathML. This is because it is easier to create the appearance of a formula than to formally define and represent its operations consistently and correctly. We instead use a formula's appearance to *suggest* operations and leave it to the reader to go through the process of consulting and analyzing the formula and its context to infer its meaning.

This preference for visual representations is equally true for text. We write prose using a sequence of characters – we almost never provide a fully annotated parse tree or other semantic representation for passages (*e.g.,* using first-order logic).

In general, for human readers semantic annotations have complex structure and are often verbose. For example, imagine reading an enhanced version of our list-based summary of the information in Example 2.1 above, rather than the original passage. How to select and *ground* primitives and relationships in semantic representations is a slippery question, and using visual representations avoids this for authors.

Semantic representations for text and formulas. Despite their verbosity and complexity, semantic representations are useful when they capture information reliably enough for a task of interest (*e.g.,* search or question answering). Creating semantic representations manually is difficult, however, we can use automated tools for this purpose. Example 2.4(c) illustrates such a representation for the query:

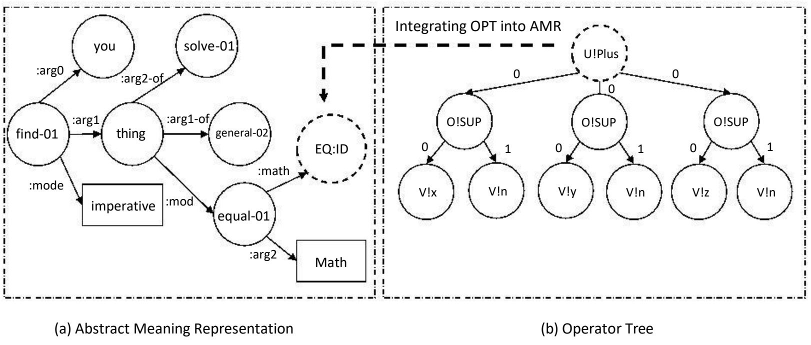
Find $x^n + y^n + z^n$ *general solution*

As seen in Example 2.4(a), an Abstract Meaning Representation (AMR) graph represents text semantics using a hierarchy of subjects, objects, actions, and attributes (Langkilde and Knight, 1998). Example 2.4(b) shows an *Operator Tree* (OPT) giving the hierarchy of operations in the query formula (*i.e.,* adding the exponentiated variables). Both have node and edge annotations capturing types and argument orderings.

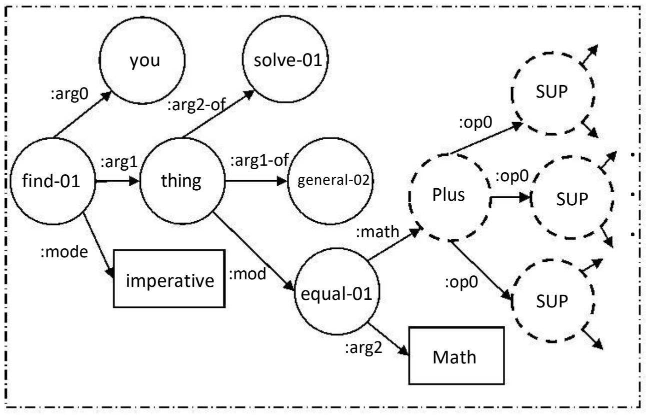
Interestingly, the AMR graph and OPT have similar purposes and structure. Both capture a hierarchy of events in a sentence or formula. For example, the root of the AMR tree is a verb (`find`) with a mode modifier (`imperative`) used to indicate that the statement is a command rather than a request. The verb has two arguments for who

Example 2.4: Augmenting AMR Trees with Operator Trees

Abstract Meaning Representation (AMR) tree, with inserted operator tree for formula in the query "Find $x^n + y^n + z^n$ general solution."



(a) Abstract Meaning Representation (b) Operator Tree



(c) Math Abstract Meaning Representation

receives the command (`you`) and what is requested: a `thing` that is the general solution to the provided equation. The root of the OPT is an unordered addition operator (`U!Plus`) applied to subexpressions from exponentiation operators with ordered arguments (`O!SUP`).

To produce the combined MathAMR tree in Example 2.4(c), an AMR tree was produced using a neural network after replacing the formula with identifier `EQ:ID` (see Example 2.4(a)). In Example 2.4(c) the identifier node is replaced by the formula OPT with annotation changes to match the AMR syntax. MathAMR was used to re-rank answers to Math Stack Exchange questions that had been converted to this representation (Mansouri *et al.*, 2022c). MathAMR inserts formulas at leaf nodes, but for longer passages one can imagine adding additional information such as links between variables and their definitions.

Consulting sources in MathIR systems. As shown in Figure 2.1, mathematical information retrieval systems require consulting sources for directly observable information, analyze and annotate their contents to generate additional information, and then organize this information in an index for lookup, search, generating training data, and use in evaluation.

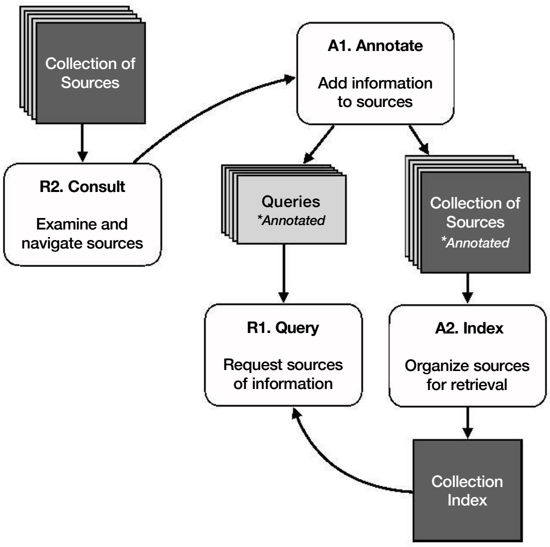


Figure 2.1: Tasks from Figure 1.4 for Representing Formulas and Text in an Index. For context, query annotation and querying the index are shown greyed-out. Content in sources is consulted, annotated to add information for text and formulas, and then indexed for later use in retrieval systems and system evaluation.

Unlike people, where we rapidly alternate between consulting and analyzing/annotating sources when trying to recover information, for large-scale systems we break processing up into steps to support batch processing. The design decisions made for each step are critically important. For example, word/sub-word and symbol vocabularies chosen to represent source contents impact the reliability of visual or semantic annotations and retrieval. We also cannot lookup or search using anything that we omit from these vocabularies, *e.g.,* if we remove frequent words and symbols such as 'the' and x to save space.

In the next section, we focus on annotating formulas and associated text with visual and semantic representations to enrich collections, and for later use in indexing.[4]

2.2 Annotating Formulas: Representations and Canonicalization

In terms of visual structure, the excerpt in Example 2.1 can be represented by a sequential graph of word and formula tokens shown in Example 2.5. Each of the blue formula nodes/tokens contain a visual structure representable using a Symbol Layout Tree (SLT) as shown for the *idf* formula in Equation (1). The blue nodes for variable names in the excerpt text each contain an SLT that can be found in the subtrees of the *idf* SLT.

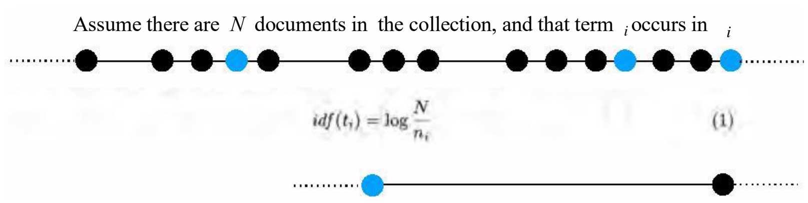
SLTs represent the placement of symbols on writing lines using spatial relationships between symbols and nested writing lines. An SLT is a directed, rooted tree with a parent and child in every relationship – the *idf* node is the root node in our example. The *idf* and *log* functions are single nodes with their characters grouped into tokens as seen earlier in Example 2.3. For easier reading, we have shown adjacent symbols on a writing line using undirected edges. The eight spatial relationships used in SLTs are: adjacent-at-right, sub/superscript, prefix sub/superscripts

[4]Diagrams and other graphics are frequently used in math, but outside of our discussion here; *e.g.,* commutative diagrams can be expressed as a matrix-like SLT container, but are really directed graphs with nodes/edges labeled by formulas:

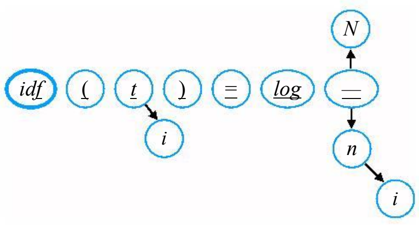
$$\begin{tikzcd} A \arrow[r, "f"] \arrow[dr, "g\circ f"'] & B \arrow[d, "g"] \\ & C \end{tikzcd}$$

Example 2.5: Text tokens, formula tokens, and visual formula structure

Sequence of text (black) and formula (blue) tokens


Assume there are N documents in the collection, and that term i occurs in i
idf(t_i) = log N/n_i
(1)

Symbol Layout Tree (SLT) for Equation (1)

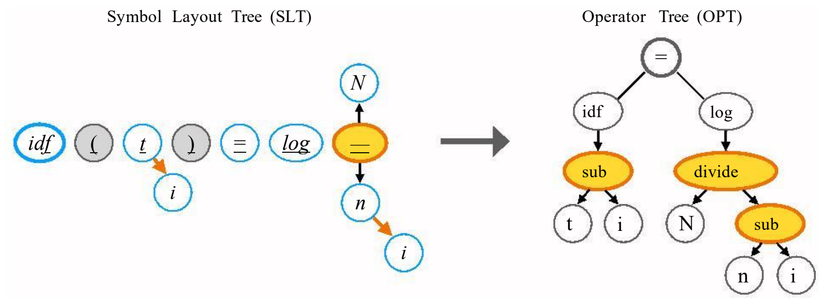


on the left side ($e.g.$, ${}^{n}_{2}C$ and ${}^{235}U$), inside ($e.g.$, $\sqrt{x}$), and above/below ($e.g.$, $\frac{N}{N_i}$).

In Example 2.1, one piece of missing information that we did not annotate is the hierarchy of operations in the *idf* formula. While a reader is unlikely to think about this consciously, interpreting the formula essentially involves converting the SLT to an Operator Tree (OPT). We saw previously that OPTs are (partial) semantic representation giving an operation hierarchy. Example 2.6 illustrates this conversion for the *idf* formula. In OPTs variables and other arguments appear at the leaves, with operations above the leaves in internal nodes. While SLTs are oriented left-right to reflect reading order, OPTs are oriented vertically to reflect operation order. The order of operations is bottom-up in the OPT with precedence decreasing away from the leaves, *e.g.,* the '=' used to represent a definition is applied last for our *idf* formula.

Operations appear directly above their arguments in an operator tree. If arguments have different roles ($e.g.$, N and n_i in $\frac{N}{n_i}$) this is captured by their left-right order below the operator. When argument order is unimportant, arguments are normally arranged in reading

Example 2.6: Translating a Symbol Layout Tree to an Operator Tree



Grey nodes in the SLT indicate parentheses removed in the OPT, where they are redundant. SLT subscript edges and fraction line are replaced by `sub` and `divide` nodes in the OPT.

order (*e.g.,* per the SLT). In our OPT directed edges indicate ordered arguments, and undirected edges indicate unordered arguments for '$=$.'

There are some subtleties with mapping formula appearance in an SLT to the operations in an OPT. For example, we use a `sub` operator to represent subscripted variable names. If we assume the intended semantic, these could be replaced by exponent nodes. However, this is not true in general, and so we map to a `sub` operator instead (*e.g.,* X^2 may be a Cartesian set product). Also, some operations without symbols in SLTs are explicit in OPTs, *e.g.,* xy represents the operation $x \times y$.

In general, we work with fixed operation sets and SLT mappings when producing OPTs automatically using tools. In some cases this leads to ambiguities or incorrect mappings. This is unfortunately unavoidable, because the meanings of symbols are community and context-dependent, and symbols are frequently redefined by authors for their own purposes. Despite these challenges with conversion, we should note that some state-of-the-art formula search engines use OPTs rather than SLTs for search, because SLTs do not fully capture the operation hierarchy.

There has been recent progress in this space. For example, In one approach to translating SLT representations as shown earlier for Wikipedia (Greiner-Petter *et al.*, 2023), noun phrases associated with SLTs are identified and then used in translating the SLTs to OPTs with additional information that allows computable functions to be recovered. An augmented LaTeX syntax represents this annotated OPT, which is then translated to computable representations for Computer Algebra Systems (Maple and Mathematica). Such semantically enriched formula encodings for use in CAS and theorem provers have long been a goal in the Mathematical Knowledge Management community (MKM), and we anticipate that there will be strong renewed interest in this problem moving forward.

SLTs and OPTs in code. Example 2.7 provides code representing the *idf* formula SLT in LaTEX, and two versions of the OPT in Python. The LaTeX is shorter because it represents only formula appearance. Commands such as `\frac` helpfully suggest operations, but define only the placement of symbols (*e.g.,* above and below a fraction line). We also see the LaTeX subscript operator in this example (`_`).

For the OPT implementations, all implicitly defined or unspecified operators and values from Example 2.1 must be provided to compute values. For example, *e.g.,* `math.log()` uses the base e, and the equals operator becomes Python's end-of-function-signature symbol (`:`). We also add variables and data structures to hold input values for terms, term counts, and the number of documents in our collection. The function signatures require additional arguments missing in the left-hand side of our OPT, because all values must be defined (*e.g.,* for `N` and term counts `n`).

The function is defined twice, first using built-in infix operators for division (`/`) and lookup (*e.g.,* `n[i]`) and a second time using functions (`idf_prefix`). While both produce the same output, note that the first version looks closer to the SLT, while the second matches the operation hierarchy in the right-hand side of the OPT.[5] idf values are assigned to the intermediate variable `idf_weight`, and annotated with their associated value of `i` and text term in the output.

In the output `"document"` has an idf value of $\log 100/100 = 0$. This

[5]Infix operators provide argument layouts more similar to typeset formulas. This may partly explain the popularity of languages with infix math operators vs. purely prefix-structured operations (*e.g.,* in `idf_prefix()` and Lisp).

Example 2.7: *idf* formula in LaTeX and Python code

LaTeX: Symbol Layout Tree representation

```
idf(t_i) = \log \frac{ N }{ n_i }
```

Python: Two Operator Tree representations

```
import math
t_all = [ "inverse", "document", "frequency" ]
n_all = [ 2, 100, 20 ]
D = 100
def idf(i, t, n, N):
   idf_weight = math.log( N / n[i] )
   return( t[i], idf_weight )

# Prefix form: ops before args to match OPT RHS
def divide(a, b): return a / b
def sub(a, b): return a[b]
def idf_prefix(i, t, n, N):
    idf_weight = math.log( divide( N, sub(n, i)) )
    return( sub(t, i), idf_weight )

for i in range(len(t_all)):
   print(i, idf(i, t_all, n_all, D))

OUTPUT:  0 ('inverse', 3.912023005428146)
         1 ('document', 0.0)
         2 ('frequency', 1.6094379124341003)
```

is valid, as the term appears in *all* documents, and so doesn't provide any distinguishing information. `"frequency"` appears in 20/100 of the documents, and has an idf less than half the value for `"inverse"`, which appears in only 2/100 documents. Note that without the log scaling, `"inverse"` would have ten times the idf score of `"frequency"`.

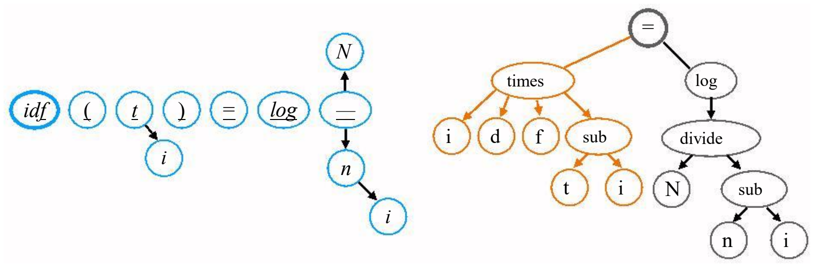
MathML: Presentation (SLT) and Content (OPT). For MathIR systems and evaluation benchmarks, MathML is a file format commonly used to represent SLTs and OPTs.[6] In MathML, OPTs are normally

[6]https://www.w3.org/Math.

defined without the additional context needed to compute values that we saw for the Python in Example 2.7.[7] SLTs are given in *Presentation MathML*, and OPTs in *Content MathML*. Presentation and MathML generated from the LaTeX for our *idf* formula are seen in Example 2.8.

Example 2.8: MathML generated from LaTeX using LaTeXML

idf is undefined in LaTEXML[a] and so *i*, *d*, and *f* are treated as variables.



Presentation MathML (SLT)

```
<math xmlns="http://.../MathML">
  <mi>i</mi>
  <mi>d</mi>
  <mi>f</mi>
  <mo stretchy="false">(</mo>
  <msub>
    <mi>t</mi>
    <mi>i</mi>
  </msub>
  <mo stretchy="false">)</mo>
  <mo>=</mo>
  <mi>log</mi>
  <mo>&#x2061;</mo>
  <mfrac>
    <mi>N</mi>
    <msub>
      <mi>n</mi>
      <mi>i</mi>
    </msub>
  </mfrac>
</math>
```

Content MathML (OPT)

```
<math xmlns="http://.../MathML">
<apply>
  <eq/>
  <apply>
    <times/>
    <ci>i</ci>
    <ci>d</ci>
    <ci>f</ci>
    <msub>
      <ci>t</ci>
      <ci>i</ci>
    </msub>
  </apply>
  <apply>
    <log/>
    <apply>
      <divide/>
      <ci>N</ci>
      <msub>
        <ci>n</ci>
        <ci>i</ci>
      </msub>
    </apply>
  </apply>
</apply>
</math>
```

[a] https://math.nist.gov/~BMiller/LaTeXML

[7] Some tools such as Maple and Mathematica provide MathML annotations and values needed to compute values from Content MathML.

Being XML-based, the syntax is similar to the prefix representation seen for `idf_prefix` in Example 2.7. Generally speaking, MathML commands begin with a start and matching end tag for the command, with a list of tags for arguments nested inside. All tags may also contain a list of attributes, *e.g.*, `xmlns` (XML namespace) or `stretchy` (controlling the rendering of brackets). An example is the `<apply>` command in Content MathML, where the first nested tag is an operator, and the remaining nested tags are the operator's arguments. MathML provides types for arguments, including `<mi>` and `<ci>` for variable identifiers, `<mn>` and `<cn>` for numbers. In Content MathML, defined operations have predefined tags, and so `log` appears as `<log/>` in Content MathML but as the identifier `<mi>log</mi>` in Presentation MathML.

The LaTeXML tool used to produce Example 2.8 knows `\log` is an operator, and inserts an *invisible* node in the Presentation MathML to capture its application to the fraction using hexadecimal Unicode value `x2061`. This symbol does not appear when this formula is rendered. In contrast, *idf* is not a defined operation or function, and is broken up into three variables in the SLT and Content MathML. In the Content MathML, these variables are multiplied with each other and t_i.

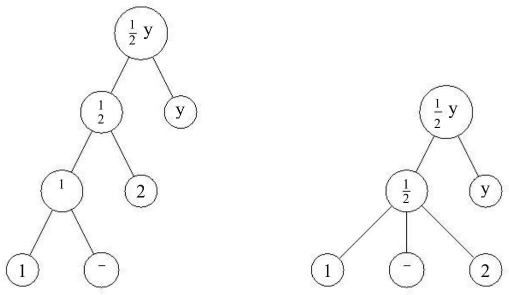
As discussed earlier, a fixed set of definitions must be used to convert formulas in LaTeX or Presentation MathML to Content MathML. This means that in large collections inconsistencies such as those seen in Example 2.7 are common along with the `<cerror>` tag for unrecognized symbols and structures. If these interpretations not intended by their authors are consistent in their representation, they still provide patterns useful for retrieval and other information tasks.

Visual and spatial region-based formula representations. Let us now consider some other visual approaches to representing formulas. For raster (pixel-based) images symbol locations are unknown. However, we can represent formulas directly as images, and compare formulas based on image similarity (we will return to this in the next section).

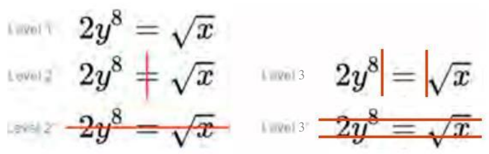
We can also capture symbol layout in raster images using an *XY-cut tree*, as shown in Example 2.9. XY-cut trees partition touching pixel groups (connected components) by cutting at horizontal and vertical whitespace gaps. The standard method strictly alternates the

Example 2.9: Region-based spatial representations for formulas

XY-Cut Trees (left: Recursive, right: Standard)



Pyramidal Histogram of Characters (XY-PHOC)



cutting direction, while the recursive version cuts the largest gap in either direction (Ha *et al.*, 1995; Nagy and Seth, 1984). Symbols can be recognized or features computed from sub-images at nodes for use in recognition and retrieval applications (Baker *et al.*, 2010; Zanibbi and Yu, 2011). A variant of XY-trees was used in one of the earliest systems for parsing math formulas from images (Okamoto and Miao, 1991).[8] XY-trees can also be produced from known symbols, *e.g.,* by cutting around symbol bounding boxes. They are also used to segment document pages into regions, which was the original purpose.

For images where symbol locations are known such as in SVG or PDF, or using symbol locations from OCR results, we can produce additional spatial representations. For example, in the previous section, we saw an example of this where line-of-sight graphs over symbols were used to search handwritten and typeset math in video keyframes. An alternative region-based spatial representation is the Pyramidal

[8]Cutting thresholds and rules avoid splitting symbols with multiple components (*e.g.,* ‘i’) and separate subexpressions from inside radicals (*e.g.,* $\sqrt{x}$).

Histogram of Characters (PHOC), which identifies which symbols appear in a fixed set of recursively partitioned regions. PHOC was originally created for retrieving words in handwritten text (Almazán *et al.*, 2014) but can be generalized in a straight-forward way for representing two-dimensional structures like formulas (Langsenkamp *et al.*, 2022; Avenoso *et al.*, 2021; Amador *et al.*, 2023).

As shown in Example 2.9, for the XY-PHOC representation the first region contains all symbols in a formula. Other regions split the formula into 2, 3, or more equal-size regions horizontally or vertically. Later versions also used concentric ellipses or rectangles to better capture symmetry (*e.g.,* for $x + y$ and $y + x$). The encoding is compact, using one bit vector per unique symbol to record occupied regions.

Canonicalizing formulas and formula representation tables. Example 2.10 visualizes common ways to generalize variables and operations in an OPT, which can also be applied to SLTs. Applying these transformations reduces the number of unique formulas in a collection, making additional formulas identical, and others more similar after these transformations are applied. The process of reducing variation in representations is known as *canonicalization.*

At the middle in Example 2.10 we number each unique variable from left-to-right in the OPT, starting from 1. This type of numbering for entities is known as an *enumeration.* '2' for i is repeated because it appears twice in the expression. We can use enumerations to capture formula structure while ignoring variable names. For example, using variable enumeration the *pythagorean theorem* expressed as either

$$x^2 + y^2 = z^2 \tag{2.1}$$

or

$$a^2 + b^2 = c^2 \tag{2.2}$$

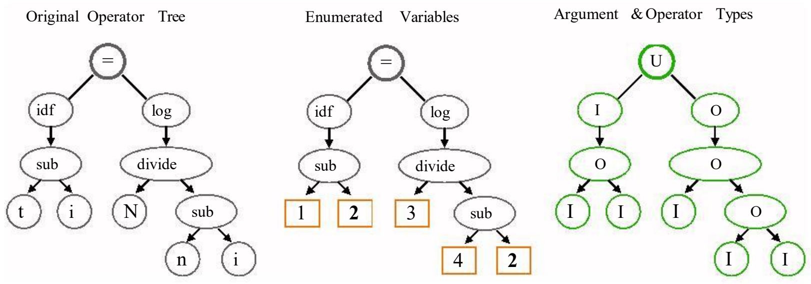
has the same form:

$$\boxed{1}^2 + \boxed{2}^2 = \boxed{3}^2. \tag{2.3}$$

We can apply this change for both the SLT and OPT representations.

For formulas, canonicalization often involves *normalizing* symbols and structures as well, *e.g.,* replacing different names for the same

Example 2.10: OPT Variable Enumeration and Symbol Types



operation with a single name, or always ordering subscripts before superscripts in SLTs (*e.g.,* in LaTeX or Presentation MathML). Another normalization re-orders variable and constant names lexicographically for operators with unordered arguments, *e.g.,* to have both $x + y$ and $y + x$ represented by $x + y$.

For MathML we often 'flatten' nested tags, such as repeated `<mrow>` tags belonging to a single writing line and sequences of unordered operations. Both of these transformations for `<mrow>` and `<times>` are applied in Example 2.8, flattening a chain of `<times>` nodes into one node in the Content MathML and removing `<mrow>` tags entirely from the Presentation MathML.

At right in Example 2.10 we have an OPT with nodes replaced by an assigned *type* for each symbol. I indicates identifiers (*i.e.,* names) for variables and operations/functions like the `idf` function identifier. Note that identifier `idf` must represent an operation, because it is an internal node in the OPT. Other predefined mathematical operations are labeled to indicate whether their arguments are (O)rdered or (U)nordered. Among other uses, types can be used to permit or constrain matches between constants, variables, and operations in two formulas. More sophisticated typing schemes have been used, *e.g.,* to distinguish numbers from alphanumeric identifiers and greek letters, and relational operations from set operations and arithmetic operations.

Canonicalization can remove unhelpful distinctions, however, too much canonicalization can also remove meaningful differences. A common compromise is to annotate all formulas with *multiple* representations stored in tables, including one table for the original encodings (usually LaTeX and/or Presentation MathML).

To annotate sources with different formula representations, each formula encountered in a source is assigned a unique integer identifier. Tables used to hold each formula representation are sorted by these formula instance ids, so that one integer can be used to retrieve any representation that we have produced. To save space, often the representation tables define only the unique formulas in each representation, and we add a second lookup table. The lookup table is used to map formula instance ids to unique ids, and the representation tables map unique ids to detailed representations (*e.g.,* canonicalized OPTs). This prevents detailed representations for x or an isolated canonicalized enumerated variable ($\boxed{1}$) from being stored millions of times.

Textual formula annotations: Math entity linking. We can also use markup available in sources along with analysis tools to capture formula-text relationships such as shown in Example 2.1 and Example 2.4. Textual formula annotations can be used for a variety of information tasks. For example, using formula symbol identifier descriptions as features for automatically generating Mathematics Subject Classification (MSC) subject codes (Schubotz *et al.*, 2016). MSC is a collaboratively-produced hierarchical classification scheme used to identify subject codes for papers in math journals. Recent math-aware search engines have also explored using annotated formulas as their collection, including the math entity cards in MathDeck (Dmello, 2019) that connect formulas to titles and descriptions from Wikipedia. Another retrieval system, MathMex (Durgin *et al.*, 2024) indexes formulas that appear with their textual descriptions in a document.

When text annotations for formulas are not provided directly in sources, Math Entity Linking (MEL) systems can be used to connect formulas with their surrounding context. Context may include descriptions for formulas and their symbols, other formulas defining symbols in a formula, and external sources (*e.g.,* linking formulas to Wikipedia pages). We note that symbols like x and λ are frequently re-defined within a single paper, leading to multiple definitions (Asakura *et al.*, 2022). This complicates the task of *coreference resolution*, where multiple references to the same mathematical symbol or entity need to be identified and disambiguated when symbols are redefined (Ito *et al.*, 2017).[9]

Most early methods for MEL were rule-based due to the limited data available for training machine learning models. One of the earliest textual formula annotations linked math expressions to their corresponding Wikipedia page (Kristianto *et al.*, 2016b); unfortunately not all math expressions have Wikipedia pages, and context provided in the document where a formula appears is likely more relevant and/or accurate for the formula. Another early system annotated formulas with descriptions and relationships to other formulas in dependency graphs (Kristianto *et al.*, 2017). Textual descriptions are extracted using an SVM-based model to link description nodes to formulas and symbols (Kristianto and Aizawa, 2014). References between formula nodes are captured through structural matching of formula sub-expressions.

Later systems including MathAlign (Alexeeva *et al.*, 2020) focused on textual annotations within the documents where formulas appear. There has also been work on automated variable typing, where pre-defined mathematical types (*e.g.,* integer, real) are assigned to variables in mathematical formulas using sentences containing descriptions where a symbol appears (Stathopoulos *et al.*, 2018).

The *SymLink* shared task at SemEval 2022 (Lai *et al.*, 2022) required extracting math symbols with their textual descriptions from LaTeX source files collected from the arXiv. The main task requires this to be performed within a LaTeX paragraph. First, all text spans (contiguous excerpts) containing math symbols and descriptions are identified, and then symbols are matched with their descriptions. The SymLink dataset provides more than 31,000 entities and 20,000 relation pairs, which allowed modern machine learning models (*e.g.,* BERT-based) to be proposed.

Math entity linking and other forms of annotating text-formula relationships are important directions for future research. They are chal-

[9]The Math Identifier-oriented Grounding Annotation Tool (MioGatto) (Asakura *et al.*, EasyChair, 2021) provides a tool for annotating different roles for formulas and symbol, linking identifiers to pre-defined math concepts extracted from the document.

lenging because incorrect detections can corrupt intended meanings, and because mis-detections can lead to cascading errors. This fragility and the computational cost of constructing explicit semantic annotations are partly responsible for the popularity of dense embedding models, which use language statistics to capture associations and usage patterns for tokens/formulas/passages etc. However, information-wise embeddings capture associations rather than discrete entities and relationships, and we expect that combining embeddings with constructing graph-based representations will prove beneficial in the future.[10]

2.3 Indexing Formulas and Text

Indexing is a critical component for both the implementation and evaluation of mathematical information retrieval systems discussed in the later sections of this work. One might consider indexing to be mostly a brute-force compilation and reorganization of source data in a collection. In fact, there are quite a number of important encoding details (*e.g.,* character encodings, file formats, and their myriad variations), organizational and retrieval unit design decisions, and resource constraints such as speed and storage requirements that must be carefully addressed if downstream model and evaluation implementation efforts are to be reasonable and effective. This is especially true for multimodal indexes used for math IR systems, where we may have multiple data representations for text, formulas, and their combination.[11]

Figure 2.1 illustrates the main tasks for indexing sources. When we talk about indexing, we're actually referring to a process that consults sources and annotates sources with additional information, and then creates a collection index. We discussed annotating formulas in the previous section, but annotations are needed for text as well. For example, if we plan to use dense retrieval with sentences, we first need to record where sentences are found in sources, *e.g.,* in a table containing pairs of start and end character positions. We often also create tables to hold metadata such as authors and logical regions such as titles, so that these may be quickly accessed or searched separately.

[10]There is related research in knowledge graph construction (Zhong *et al.*, 2024).

[11]In our experience, implementing new indexing tools is a substantially larger effort than implementing retrieval and machine learning models using indexed data.

The collection index contains data structures for organizing both sources and their contents, along with search indexes that organize source contents by *patterns* generated from sources and their annotations. More concretely, indexing involves:

1. consulting source text and formulas to generate dictionaries for fast lookup and analysis of their contents,
2. adding information to sources through additional dictionaries (*e.g.,* formula locations and representations), and
3. generating inverted index and/or dense vector index files from source contents and annotations.

Locating and extracting formulas from sources. We often start by identifying formulas in our sources. Videos and PDF documents generally do not identify formula locations, and so we create a table mapping integer formula identifiers to their locations. An example is the Page-Region-Object tables used for ACL anthology PDFs in the MathDeck system (Amador *et al.*, 2023). Each detected formula has an entry with integer source and page ids, and two x-y coordinates for the top-left and bottom-right corners of a *bounding box* containing the formula. Formula representation tables are then created from the detected formulas.

For text documents with demarcated formulas (*e.g.,* LaTeX and HTML with MathML), we extract encodings for formulas and construct the formula representation tables needed. For some applications such as generating MathAMR trees or training transformer models such as BERT, it can be helpful to replace formulas by an identifier, *e.g.,* `<math [...] </math>` becomes `EQ::42`.

Vocabularies for text and formulas. A simple but critical annotation are *vocabularies*. A vocabulary defines a set of unique objects/symbols seen in our collection including words, XML tag types, LaTeX commands, and other math symbols (*e.g.,* in unicode). Vocabularies for text and formulas may be stored separately, together, or both. Different formula representations will generally have their own separate vocabularies. In general we compile all unique ‘words’ for each vocabulary, prune some of them (*e.g.,* removing rare 'words'), and then enumerate vocabulary items for use in table lookups.

For formulas, in addition to individual symbols, we often need subexpressions or substructures in our formula vocabulary, *e.g.,* for formula search. Example 2.11 shows two common techniques for this, using unrooted and rooted paths. The example at top shows directed paths in an SLT. Shown are all unrooted paths of length 1, and four paths of length 2. In the example at the bottom, *leaf-root* paths that start from leaves of an OPT are shown, and we show all such paths for the left-hand side of the *idf* formula. Note that each OPT leaf-root path is a valid subexpression with one argument and one or more missing arguments and operations, while a number of the SLT paths capture visual patterns, but are not valid subexpressions (*e.g.,* '(' $\rightarrow$ 't'). Both types of paths can be extracted from SLTs and OPTs.[12] One constraint is that indexing too many paths for a formula representation increases index size, and can lead to slow retrieval times. So our choice of substructure vocabularies such as path types requires careful consideration.

For example, we can construct a text token vocabulary for use with BM25, and a variety of path vocabularies to search OPTs, SLTs, and additional canonicalized versions described in the previous section.

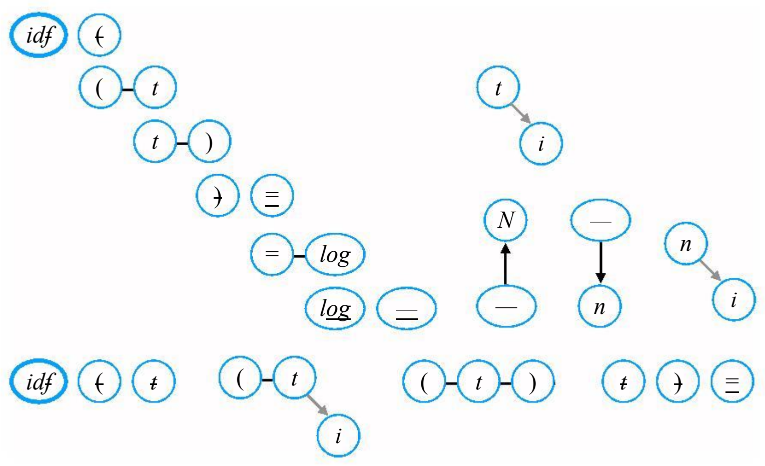
Inverted indexes for sparse retrieval. Among other things, vocabularies define the patterns that can be used to retrieve information directly from index tables. This type of pattern lookup in an index is often called 'sparse' retrieval, because a standard query can be represented by a largely empty vector with bits or counts representing the vocabulary terms in the query (Zobel and Moffat, 2006). BM25 is a sparse retrieval model, as are formula search models that retrieve formulas from tables using paths. A table that maps vocabulary terms to lists of documents, formulas or other object identifiers is called an *inverted index.*

As an alternative to inverted indexes, people have also used substitution indexing trees, originally developed for unification and matching of predicates in automated theorem proving (Graf, 1995). These trees

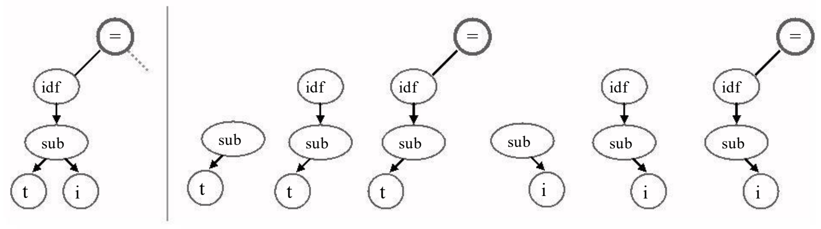
[12]Anchoring paths only at the root of an SLT or OPT is generally ineffective; many helpful patterns are missed. Leaf-root paths on SLTs are an interesting opportunity.

Example 2.11: SLT and OPT paths for the *idf* formula

Symbol Layout Tree Paths (directed)



Operator Tree Paths (leaf-root)



group OPTs or SLTs with shared structure using a hierarchy of symbol and operation replacements and enumerated variables (Kohlhase and Sucan, 2006; Schellenberg *et al.*, 2012). Substitution trees represent the complete set of possible operation sequences that produce concrete formulas in a collection at their leaves. Retrieval finds the most similar formulas in the substitution tree through transformations of the query.

Vector embeddings for dense retrieval. For dense retrieval patterns used for search are transformed into embedding vectors, and retrieval involves finding some k most similar items based on the geometry of the embedding space (*e.g.*, using the cosine of the angles between the vectors). This may be for tokens, sentences, formulas, paths, or other objects. A common approach for generating embedding vectors is having a neural network play an imitation game where we hide tokens (*i.e.,* mask them) in a text sequence or node/edge labels in an input graph (*e.g.,* an OPT or SLT), and have the network produce likelihoods for every alternative in a vocabulary. During training, this game is played repeatedly using a training data set, with network weights updated to improve estimates.

Masking and other *self-supervised* learning tasks capture a *language model* reflecting the likelihood of objects appearing in similar contexts. The often large amounts of computation required is somewhat confusingly referred to as *pre-training* because network weights are not optimized for retrieval or other tasks aside from the learned probability estimates.[13] To further improve dense retrieval performance, additional *learning to rank* tasks are run, which require generating ranks for individual sources (*point-wise*), comparing two items at a time (*pair-wise*) or revising entire rankings (*list-wise*). This training directly for retrieval or other 'downstream' tasks after learning a language model is known as *fine-tuning* of network weights for a specific task.

Learning to rank requires test collections where sources *relevant* to specific queries have been identified, as described in the next section; normally relevance labeling is at most partially automated (Faggioli *et al.*, 2024). As a result, data available for learning-to-rank is often much smaller than for language model learning. In contrast, large amounts of pre-training data can be created by randomly hiding labels or other random manipulations without human involvement.

After the embedding network has been trained, items to be indexed are converted to embedding vectors and stored in one or more tensors. Normally we also produce a companion dictionary mapping the tensor rows to integer identifiers for the objects that have been embedded, so that we can recover the passage, word, formula or other object they come from.

[13]'Pre-trained' language models often produce surprisingly strong retrieval baselines. For example, symbols or sub-expressions that look quite different may have similar vectors if they are often used in similar contexts, because when masking these, one or the other will be more likely than other vocabulary items.

3

Math Retrieval Tasks and Evaluation Metrics

In this section, we introduce the retrieval tasks considered in this monograph. There are two task types that we consider: *search tasks* requiring a ranked list of sources from a collection, and *question answering tasks* (QA) requiring a single response that may or may not reference sources. We categorize these tasks by expected responses below.

Search Tasks: Return ranked list of sources

1. *Formula search:* ranked formulas
2. *Math-aware search:* ranked sources or excerpts with formulas and text (*e.g.,* passages)

QA Tasks: Provide question answer possibly with justifications and/or step-by-step solutions

1. *Multiple choice:* selection from provided alternatives
2. *Value:* numbers or strings, possibly in data structures (*e.g.,* lists)
3. *Formula:* formulas or short programs for computing answers (often paired with a value response)
4. *Open response:* free response with formulas and text

While we have selected these tasks to focus our discussion, this is by no means the exhaustive set of information tasks required or studied for Math IR. For example, other pertinent tasks include math entity linking (Kristianto *et al.*, 2016b), tasks related to theorem proving such as natural language premise selection (Ferreira and Freitas, 2020b) (see Appendix B), and extraction and annotation of formulas that are not explicitly demarcated in collections (*e.g.,* for handwriting, Truong *et al.*, 2024, and PDF documents, Shah *et al.*, 2021). Additional tasks are described in other surveys of mathematical information retrieval (Zanibbi and Blostein, 2012; Guidi and Sacerdoti Coen, 2016; Dadure *et al.*, 2024).

3.1 Evaluation Overview

Figure 3.1 visualizes the people, data, and processes used to create retrieval task data and evaluate search and QA tasks. The figure illustrates an important but easy-to-miss fact: it is *people* and not formal definitions, systems, or algorithms that define retrieval effectiveness, and the target responses that retrieval systems are designed to produce. More specifically, these roles are:

- **Users:** Realistic search queries and questions come from human *users* either directly or from available data (*e.g.,* in query logs).
- **Assessors:** Identify relevant sources for search queries, and define correct question answers.
- **Designers/Researchers:** Collect queries, results, assessments, and set the measures and procedures that quantify performance.

As a result, a chosen evaluation design and the human actors within it influence the data produced, along with any observations and conclusions made from this data. It is important to consider this when reviewing evaluation results in the work of others, and when designing our own evaluation frameworks and experiments.

This dependency of system evaluation on people is unavoidable. The tasks retrieval systems perform are motivated by human information tasks, which system designers approximate through observation and then formalization within a system design. Also, large data sets recorded from

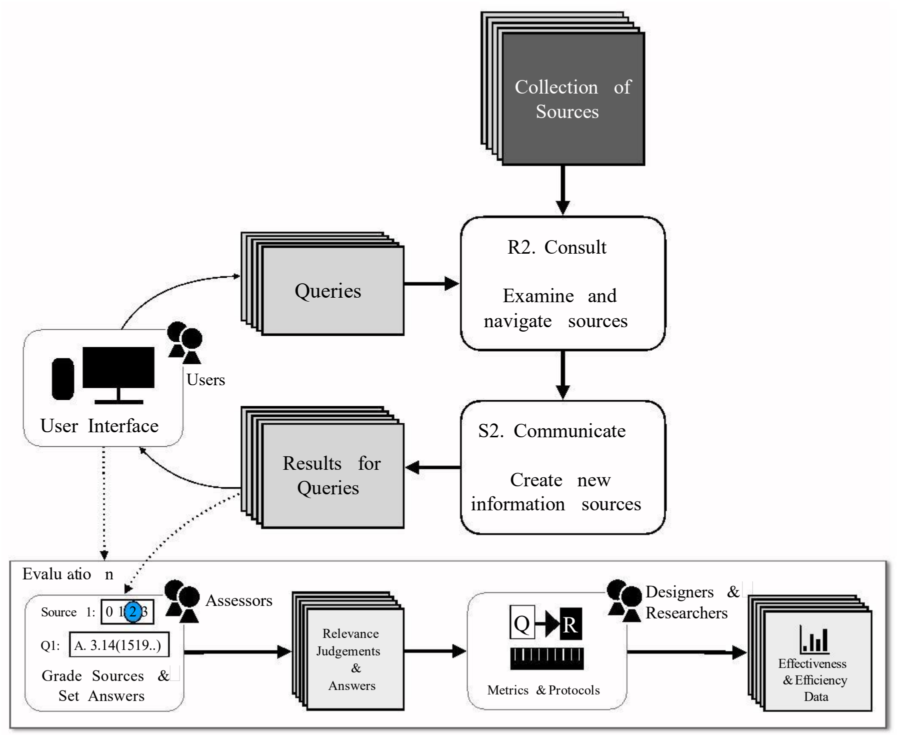


Figure 3.1: People, Data, and Methods in Evaluation (expands Figure 1.4). Search tasks require rankings sources in a collection, while QA tasks require a single response. System information tasks are in gray/hidden: systems are used to sample sources for grading search relevance, but are generally unused for QA evaluations.

users and sources authored by people are *human* data. Effectiveness is measured by how well system outputs imitate human responses collected by researchers. For systems utilizing machine learning, desired outputs are obtained by repeatedly playing imitation games scored by the distance between model outputs and human responses. For these reasons, people determine or constrain nearly every aspect of system design and evaluation.[1]

For search tasks, systems also have a direct role in evaluation aside from producing results, sometimes even for their own evaluation. This is because we normally *pool* sources returned from multiple systems

[1]This is a feature, not a bug.

before assessors make relevance judgements. This does introduce bias in evaluation, because only items returned by systems used in pooling are assessed. However, most collections are too large to proceed any other way, and we can choose metrics to mitigate the effects of this bias on evaluation outcomes.

Evaluation-wise, we focus here on the effectiveness of query results as measured offline using *test collections.* A test collection consists of:

1. **Collection of Sources:** sources to be searched for search tasks; optional, usually missing for question answering tasks.
2. **Topics:** queries to run. For search this also includes query information need descriptions and criteria for relevant sources. For QA questions may include context and/or explanations for answer requirements.
3. **Responses:** includes pooled sources with relevance judgements for search tasks, and question answers for QA tasks.
4. **Protocol:** metrics and methods for producing evaluation data.

In addition to being aware of the roles people play in retrieval evaluation, it is important to remember test collections provide a *sample*, and not all possible queries and results for a task. As such, the data that we collect provides *evidence* for hypotheses (*e.g.,* system A performs better than B for metric M), and not proof (Fuhr, 2017). With that said, many valuable things can be learned about system behaviors, evaluation data and frameworks, and even retrieval tasks themselves using test collections. They allow us to address questions using direct observation in addition to observations reported by others.[2]

Briefly returning to our 'jar' model for information tasks in Section 1 (Figure 1.3) using test collections for evaluation and related experiments is an important information *synthesis* task that involves *applying* information to *communicate* new information sources such as research papers. Reporting informative evaluation data requires significant effort: careful checking is needed at every step of data collection, measure-

[2]Galileo expressed concern about getting information primarily by following entries in book indexes rather than experimentation (see Duncan, 2021, pp. 9-10).

ment, analysis, and reporting. Test collections help ease this burden by providing standard data sets and methods for system comparison.

Efficiency metrics. While discovering new retrieval models and understanding their information use and effectiveness is generally the focus of IR research, for large collections and real-world systems efficiency is also very important.[3] Metrics such as mean query response time (*MRT*, *i.e.,* average seconds/query), query throughput (*i.e.,* average queries/second), and index size on disk and in memory are used to evaluate system speed and resource utilization. Efficiency metrics are also needed to check tradeoffs between time, space, and effectiveness.

3.2 Retrieval Tasks: Search and Question Answering

Some example formula search queries and results are shown in Example 3.1. Top-to-bottom, the examples include a formula used directly as a query,[4] a formula query with wildcard symbols that can be replaced by subexpressions, and a *contextualized* formula search query where the context the formula appears is included in the query, and returned formulas include their contexts.

The concrete and wildcard formula queries are symbolic similarity searches, with relevance determined by just formula appearance (SLTs) and/or operations (OPTs). This type of query is motivated by information needs including refinding a previously seen formula in a document collection, or browsing for similar formulas. Wildcards add boolean constraints to queries, as non-wildcard symbols and structures are ideally the same or as similar as possible in the formula returned. Wildcard names can also indicate repetition, *e.g.,* for wildcards **?f**, **?v**, and **?d** in the example.

Contextualized formula queries include the context where a formula appears, incorporating the types of formula-text interactions described in the previous section. Here relevance is determined by both the formula and the text within which it appears. For example, two instances of the formula X^2 are distinct if in one context X is defined as a number, and

[3]Quoting James Cordy: "Get it right, then make it fast." We add a proviso: "...but have a fast enough version for debugging and study, *e.g.,* using a small collection."

[4]A 'concrete' query.

Example 3.1: Formula Search Tasks

Formula similarity search

Query	Result 1	2	3	4	5
$y = \frac{a+bx}{b-x}$	$y = \frac{a+bx}{c+x}$	$y = a + bx$	$y = \frac{a-bx}{c-x}$	$y = \frac{a+bx}{x+c}$	$g(x) = \frac{x}{x-a}$

Formula similarity with wildcards (?w) (Aizawa et al., 2013)

Query	Result	
$\frac{?f(?v + ?d) - ?f(?v)}{?d}$	1	$g\prime(cx) = \lim_{h \to 0} \frac{g(cx + h) - g(cx)}{h}$
	...	...

Contextualized formula search

Query	Result	
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.	1	... which can be obtained by manipulating the second derivative of $\sum_{k=0}^{n} \binom{n}{k} z^k$ and let $z = p/(1-p)$ ...
	2	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 $n \le m$. This comes from the fact that ...

in the other it is defined as a set, whereas using symbolic search over SLTs, both formulas have the same representation.[5]

[5]For OPTs, the nodes for the variable and squaring operation may or may not differ, depending upon how the collection is created (see previous section).

Examples of math-aware search and mathematical question answering tasks are shown in Example 3.2. We first show an *ad-hoc* math-aware search task with queries that include formulas and text. *Ad-hoc* refers to the fact that queries can vary greatly, and may include patterns that are not proper phrases or sentences (*e.g.,* keyword queries, or the query ‘$\underline{x^2+5=30}$ `x value`’). In the example shown, a full question post from the Community Question Answering (CQA) platform Math Stack Exchange (MSE) is used as a query, and a collection of MSE answer posts is searched.

The bottom portion of Example 3.2 shows two question answering tasks. The first is an *open response* to an MSE question post, that was generated using GPT-3 (Mansouri *et al.*, 2022a). The second shows *math word problems* taken from two test collections. In both cases, the result should be an answer with two parts: an equation that can be used to compute the solution, and the solution value, here a number and a number list. *Multiple choice* questions are also common. These require choosing from a small number of provided alternatives (*e.g.,* 4 for a fourth alternative *(d) None of the above*). Research-wise, multiple choice questions allow varying the complexity of information associated with questions and alternative responses, while constraining system outputs to an alternative from a small set. In some collections questions and multiple choice answers include visual elements such as tables or diagrams. Using multiple choice questions makes it possible to study this type of *multimodal* question answering without needing to change the response format.

3.3 Creating Test Collections

An important first consideration is where to collect queries from, and how to select which queries to include. As our goal is evaluating performance on real-world tasks, it is usually best if search queries and questions come from real-world users and use cases. For example, for search tasks topics may come from query logs or community question-answering websites. Questions on standardized tests such as the Math SAT are commonly used for question answering. In some cases, topics generated by the test collection creators are designed to explore specific scenarios for new features (*e.g.,* wildcards in formula search queries).

Example 3.2: Math-Aware Search and Question Answering Tasks

Math-aware search (ad-hoc retrieval)

Query	Result	
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.	1	...which can be obtained by manipulating the second derivative of $\sum_{k=0}^{n} \binom{n}{k} z^k$ and let $z = p/(1-p)$ ...
	2	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 ...

Math Question Answering (Mansouri et al., 2022a)

Query	Result
What does it mean for a matrix to be Hermitian?	A matrix is Hermitian if it is equal to its transpose conjugate.

Math word problems

Query	Result	
	Equation	*Answer*
Sarah has 5 pens, David has 3 pens. How many pens do they have?	$x = 5 + 3$	8
Find two consecutive integers whose sum is 7.	$x + (x + 1) = 7$	3, 4

From MathQA (Amini *et al.*, 2019) & Dolphin18K (Huang *et al.*, 2016)

The final topics sets ideally provide a *representative* sample for the task being evaluated, while including some *diversity* in topics so that different system capabilities are tested. Diversity is sometimes addressed using separate sub-tasks for a test collection. For math retrieval, criteria to consider include mathematical subjects covered, modalities in queries and responses (*e.g.,* formulas, text, diagrams), and the complexity or mathematical difficulty (*e.g.,* target grade levels).

Train topics, test topics, and cross-validation. Normally a test collection divides topics into *training* and *test* topics, so that systems can be compared using the same *test* topics, while being tuned using a separate group of *training* topics. This way, all systems take the same 'test,' without having seen the test search queries/questions previously (*i.e.,* not 'cheating'). This allows us to observe and compare the information and task *generalization* captured in system data structures (*e.g.,* network weights) and algorithms for the same unseen topics. Systems should **never** be tuned on test topics when reporting test results. Published benchmarks for test collections are results for test topics by default.

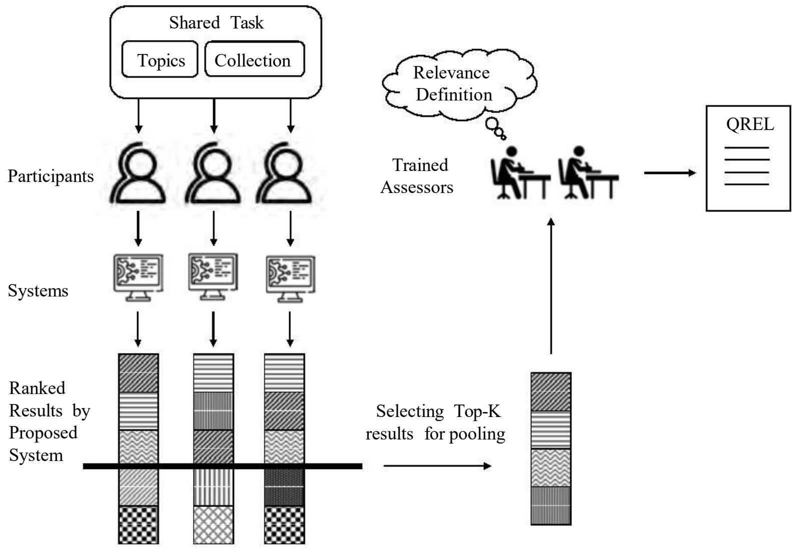
Training topics are provided for tuning system parameters. To obtain a more detailed characterization of system behavior using multiple train/test splits, *cross-validation* can be used, and average metric values across splits reported, ideally along with a standard deviation to characterize variance (roughly, consistency across splits). All train and test topics may be combined before generating splits for cross-validation, but it must be made clear which topics are used and how train/test splits are produced (*e.g.,* leave-one-out treats every topic as the test sample separately; 5-fold cross validation randomizes the topic order, and then makes 5 equal splits, with each split being rotated as the test split, etc.). While cross-validation provides more robust evaluation measures, it is important to again note that official benchmarks for test collections are computed for *unseen* test topics, and in this case test topics *cannot* be used in training.[6] Unless noted otherwise, for benchmarking test topics are scored just once, without cross-validation.

Test collections sometimes include smaller train/test topic sets used primarily for development, and/or to make use of the collection easier for those new to a task. These can also be safely produced from small subsets of training topics (*never* test topics), and are very helpful for fast testing and debugging.

[6]This can be easy to miss amongst multiple data set versions. Care is needed.

Responses: Pooled relevance judgments for search tasks. For question answering tasks, answer data is often compiled from available sources by those creating the test collection. In contrast, for search tasks relevance judgements are needed before a test collection can be released, as they are required to measure effectiveness. The most common approach is *shared tasks* in which multiple participants run their systems on provided search topics, and then share the outputs of their runs for pooling as illustrated in Example 3.3. After these assessments have been collected, relevance judgements are used to score participants' systems, and the relevance judgements produced are included when the final test collection is released.[7]

Example 3.3: Creating query relevance judgements (QRels)



In a shared task, all participants have their system(s) search the same collection of sources, and use the same topic queries. Assessors assign relevance scores to pooled sources, and these assessments are

[7]Ideally, the system *runs* (ranked responses for every topic by each participating system) are also included in the test collection for later study and comparison.

stored in *qrels* files (quantitative relevance assessments).[8] Shared tasks are run frequently at conferences including TREC (Text REtrieval Conference), CLEF (Conference and Labs of the Evaluation Forum), NTCIR (NII Testbeds and Community for Information access Research), and FIRE (Forum for Information Retrieval Evaluation).

Example 3.4 illustrates relevance assessments using a binary scale (*i.e.,* `1` is relevant, `0` non-relevant), and *graded* relevance where an ordinal scale of three or more values is used, *e.g.,* `Non-relevant` and `Low, Medium, High` relevance. Graded relevances can be easily converted to binary relevances by thresholding. For our graded relevance example, we might map `Non-Relevant, Low` (0,1) $\rightarrow$ 0 (non-relevant), and `Medium, High` (2,3) $\rightarrow$ 1 (relevant). We also see an example of *unknown* relevance for a formula search in the search results from system A at top (rank 3). Model A retrieved a formula that was not included in the pool created during the shared task, and so it is missing in the published qrels file. We will come back to this later in the section.

Normally measures of agreement between assessors are reported for search test collections. This is done by providing the same set of topics among assessors and comparing their assessments with agreement measures such as Cohen's Kappa coefficient. Properly training assessors can help increase agreement among assessors. For instance, in ARQMath-3 (Mansouri *et al.*, 2022a), for the formula search task, the Cohen's Kappa value increased from 0.21 to 0.52 from the first training to the last (third) training session.

Relevance assessment and tools. Assessing relevance for math search is inherently challenging. As discussed in earlier sections, a person's mathematical expertise influences their perception of relevance: a highly technical document relevant to an expert might be irrelevant to someone with a basic understanding of math. It is necessary for assessors to have an appropriate mathematical background and to be trained for each search task that they will assess. They should be provided with well-defined relevance definitions, including instructions on how to

[8]The standard qrels format is from TREC (https://trec.nist.gov/data/qrels_eng).

Example 3.4: Relevance Assessments

Binary and unknown relevance (?)

	Model A		**Model B**	
Rank	*Formula*	*Relevance*	*Formula*	*Relevance*
1	$y = \frac{a+bx}{c+x}$	1	$g(x) = \frac{x}{x-a}$	0
2	$y = a + bx$	0	$y = \frac{a+}{c+bx}$	1
3	$y = a-bx$	$\boxed{?}$	$y = \frac{a+bx}{x+c}$	1
4	$y = \frac{a+bx}{x+c}$	1	$y = \frac{a\ x}{b+cx}$	1
5	$g(x) = \frac{x}{x-a}$	0	$y = a + bx$	0

Graded relevance (0-3: Non-, Low, Medium, High)

Query	**Result**	**Relevance**
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.	1 . . . which can be obtained by manipulating the second derivative of $\sum_{k=0}^{n} \binom{n}{k} z^k$ and let $z = (1\ \ )\ldots$ $p/\ \ -p$	**(3) High relevance**
	2 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 . . .	**(0) Non-relevant**

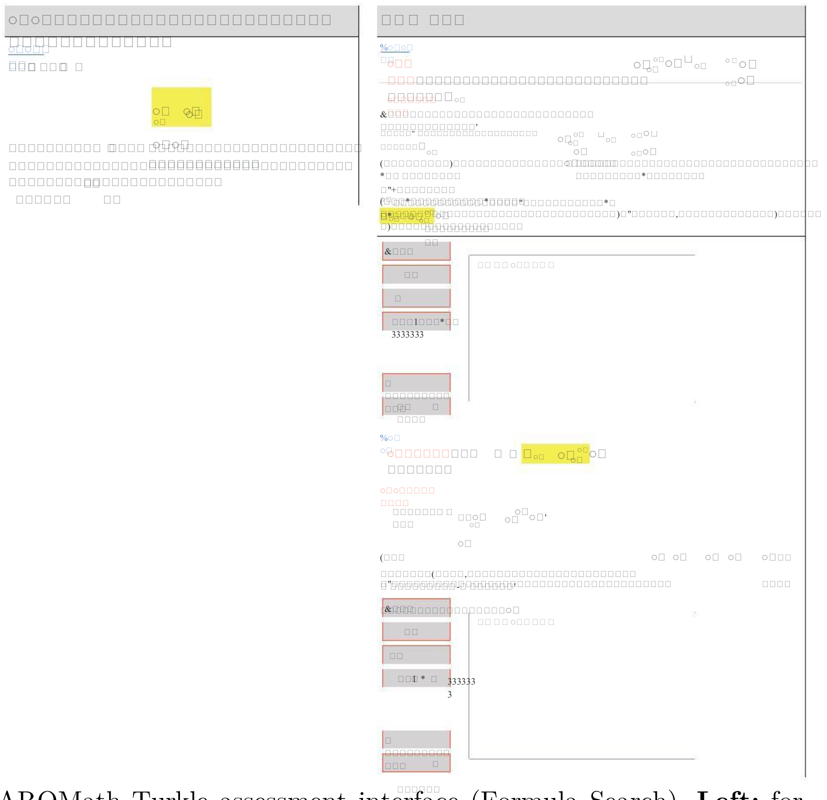
distinguish between different relevance degrees. It is also a good idea to allow assessors to decline assigning a score when they are very uncertain, or to consult an expert.[9]

[9] In our own work, a math professor acted in this role.

Example 3.5 shows the Turkle interface used for contextualized formulas search in the ARQMath shared tasks. Each assessor has an account, and is assigned topics to evaluate. Relevance data is compiled automatically and converted to qrels files by the system. In the example we see a query formula in its MSE question post on the left, and two formulas in their MSE answer posts taken the assessment pool. Assessors were allowed to view the question threads that queries and results appeared in using provided links. On the right we see buttons for the 4-level graded relevance scores, and two additional buttons for system failure (*e.g.,* when a result is unreadable), and when an assessor was uncertain how to rate the result. A box for comments was also included, and was primarily used to explain why assessors selected "System failure" or "Do not know."

An important practical consideration is the time and effort assessors require to generate answers or judge relevance for search results. For example, in ARQMath the contextualized formula search task had an average assessment time of roughly 35 seconds for *each* formula query and candidate formula. Using the same MSE question/answer posts there was a second answer retrieval task, where assessors had to decide how well an answer addressed a given question. Assessors found this task much more difficult, and average assessment time was nearly twice as long as for formula search: 64 seconds per answer.

Responses: Question answers for QA tasks. Unlike search tasks, QA test collections may be created without the participation of QA systems (*i.e.,* shared tasks are not needed to create QA test collections). A list of QA test collections is provided in Section 6. Answers are created, annotated, and checked in a variety of ways, with differing levels of effort for assessors and test collection creators. For example, LaTeX `\boxed{}` commands already identify final answers for the AMC and AIME QA data sets, while the Amazon Figure Eight annotation platform was used to manually select operations and arguments from provided lists for MathQA. Other collections have assessors generate answers for questions directly (GSM8k), or use machine learning techniques to automatically segment final answer values from available CQA answers (*e.g.,* Example 3.5: Relevance assessment for formula search



ARQMath Turkle assessment interface (Formula Search). **Left:** formula query highlighted in a question post. **Right:** two question posts containing the same formula. Assessors consider posts when deciding relevance for each pooled formula. They could also check question threads associated with posts using the 'Thread' links.

from answers posted in YahooAnswers using SVMs for Dolphin18K). For AQuA-RAT, crowdsourced workers were used to modify questions (*e.g.,* changing variable values) and provided rationales as needed from available GMAT and GRE test questions.

In addition to final answers, some QA test collections include step-by-step answers or textual annotations. For machine learning models, some of this textual data associated with *training* topics can be used to improve system answers by providing additional contextual data/information, and used to improve generated explanations for answers or improve responses to queries and comments in interactive discussion (*e.g.,* for large language models).

In all QA datasets, there are also post-processing steps to normalize the format of answers, to avoid missing correct answers. Where people are used to generate and annotate answers, the same concerns regarding assessor training, consistency, and effort as mentioned for search tasks apply here as well.

3.4 Evaluation Metrics

To be focused in their purpose, in addition to collecting queries, responses, and assessments (*e.g.,* in qrels, or target answer files) test collections also need to define the measures of success and how they are computed. This way people can use test collections independently, and compare their results in a consistent, meaningful way. We want our measures to be automated for consistency and to avoid error as well.

Selecting appropriate evaluation metrics is subtler than it may seem at first glance. For example, consider the top-5 retrieval results from two formula search models in Example 3.4. Which system would you prefer? We might prefer model 'A', because the first result is relevant. However, for some information needs we may prefer model 'B', because more relevant formulae are retrieved. This is a simple example of how information needs influence which effectiveness metrics are better suited to an individual topic or task.

As another simple example, suppose that we ask a QA system to `'provide the value of` π`'`, and that we have stored the answer as 3.14159. If a single value is expected, we need to define the required number of digits for a response to be scored as correct. We might accept 3.14, but probably not 3; if more digits of π are returned than in our stored answer, we probably don't want to penalize this either. The differences in answer formats mean that measures of *Accuracy* for math QA systems include normalizations of target responses and provided answers, which include tolerances and constraints to avoid penalizing correct answers.[10]

[10]This often loses detail, *e.g.,* π is irrational, with infinite decimal places.

Some protocols for evaluation are more complex, such as the use of *visually distinct* formulas for evaluating formula search in ARQMath (see Section 4), which impacts pooling for assessment and the scoring of search results. To help people using test collections, normally provided evaluation scripts are used to run evaluation protocols automatically.

Search metrics. Table 3.1 presents metrics used to evaluate search effectiveness using relevance assessments (*i.e.,* qrels data). Note that while most metrics are defined for a single query, they are often reported using their average value for a set of test queries. It is helpful when standard deviations from the mean are also reported, to characterize how much the metric values vary across queries. Most common metrics use binary relevance values for their computation. As described earlier, we can binarize graded relevance values to compute these metrics.

There are trade-offs that occur between some of these metrics. A classic example is the tradeoff between *recall* and *precision*: the more items we return in search results, the more likely that relevant items will be included, which tends to increase recall metrics. However, returning additional items tends to produce more non-relevant than relevant items, which decreases precision metrics. Intuitively, this is because to retrieve more items, patterns used for matching need to be less constrained, making them more likely to match non-relevant sources. In theory, we can return the entire collection for all queries to obtain an average recall ($R@hit$) of 100% because all relevant items are returned. However, the average query precision ($P@hit$) will be close to 0%, because few/none of the returned sources are relevant for an individual query.

Another important trade-off occurs when systems are designed to maximize *mean reciprocal rank* (mRR), the average inverse rank of the first relevant hit ($\frac{1}{r}$), or *precision*@1, the percentage of queries with a relevant source at rank 1 (*i.e.,* at the top of the ranking). Doing this often reduces metrics for effectiveness over a full ranking such as mean average precision (mAP) or normalized discounted cumulative gain ($nDCG$). This occurs because using retrieval patterns and scoring narrowly focused on matching high relevance sources can bias the model away from capturing rarer and/or partial relevance signals.

Table 3.1: Search Metrics. By default, sources not in relevant set R are in non-relevant set N. S_k: first k sources returned. **Note:** reported metrics are typically *averaged* over queries but only mAP and mRR explicitly mention mean query values. Ch. 8 of Croft *et al.* (2009)'s textbook provides an overview of these metrics.

	Name	Formula	Description
Binary Relevance			
RR	**Reciprocal Rank**	$1/k_f$	Inverse of rank k_f for the first relevant source
mRR	Mean Reciprocal Rank	$\frac{1}{\lvert Q\rvert}\sum_{q\in Q} RR(q)$	Avg. RR for query set Q
R *or* R@hit	**Recall**	$\lvert S\cap R\rvert/\lvert R\rvert$	Percentage relevant in returned
R@k	Recall at (rank) k	$\lvert S^k\cap R\rvert/\lvert R\rvert$	Percentage relevant in first k sources returned
P *or* P@hit	**Precision**	$\lvert S\cap R\rvert/\lvert S\rvert$	Percentage returned in relevant
P@k	Precision at (rank) k	$\lvert S^k\cap R\rvert/k$	Percentage first k sources returned in relevant
AP	Average Precision	$\frac{1}{\lvert R\rvert}\sum_{k\in K^r} P@k(S_k)$	Avg. $P@k$ for relevant documents at ranks K_r in sources returned
mAP	Mean Average Precision	$\frac{1}{\lvert Q\rvert}\sum_{q\in Q} AP(q)$	Avg. AP for query set Q
Graded Relevance			
DCG@k	**Discounted Cumulative Gain**	$r_1 + \sum_{i=2}^{k}\frac{r_i}{\log_2 i}$	Sums relevance scores r_1 through r_k for first k sources returned using log discount from rank 3 on
iDCG@k	*Ideal* DCG		DCG@k for first k pooled assessment scores after reverse sorting, *e.g.,* $(a_1,\ldots,a_5)=(3,3,2,1,0)$
nDCG@k	Normalized DCG at (rank) k	$DCG@k/iDCG@k$	Percentage of ideal DCG obtained for first k sources returned
nDCG or nDCG@hit	Normalized DCG	$nDCG@k$ for $k=\lvert S\rvert$	nDCG for all returned sources S
Scored Sources Only (N **: Non-relevant sources)**			
Bpref	**Binary preference** *(w. binary relevance)*	$\frac{1}{\lvert R\rvert}\sum_{k\in K^r}\left(1-\frac{\min(\lvert S_k\cap N\rvert,\lvert R\rvert)}{\lvert R\rvert}\right)$	Avg. percentage relevant before non-relevant for relevant sources; treats $\lvert N\rvert = \lvert R\rvert$ to balance classes.
M′	**Prime metric**	*(as defined for metric M)* *e.g.,* P′@5, nDCG′	Compute M with $S' = S\cap(R\cup N)$ rather than full ranking S

Not all of the retrieval metrics in Table 3.1 behave as expected, or are always applied or interpreted appropriately in the research literature.[11] For example, both mAP and $nDCG$ ($nDCG@hit$) characterize relevance in complete rankings. They are helpful for understanding differences between full rankings and the retrieval models that produce them. However – most users consider just a small number of results returned, and so other measures are more appropriate for user-oriented evaluations (*e.g.,* $P@5$ or $nDCG@5$). The tendency for users to examine few items also motivates the logarithmic discount used for $nDCG$. $nDCG@5$ gives decreasing credit for relevant items starting at rank 3, in contrast to $P@5$ where all relevant items in the top 5 hits have the same weight. $P@5$ uses binary relevance: items are identified as *either* relevant or non-relevant. $nDCG@5$ instead uses graded relevance, where relevance scores may have different 'levels' (*e.g.,* 1 for 'Low,' 2 for 'Medium,' and 3 for 'High' relevance).

In the end, no metric is *better* on its own – it depends upon what we want to measure. If say we want to know how often relevant items are in the top-5 hits, $P@5$ is simpler to interpret than $nDCG@5$. But if we instead want to characterize how often *highly* relevant sources appear in the top-5 hits, then $nDCG@5$ is more helpful.

To avoid bias in model comparisons, even when there are disputed or incorrect relevance assessments in a qrels file, we report performance measures using the *original* qrels file. 'Cleaning up' a qrels file by adding or revising entries is to be avoided, as it prevents direct comparison with other published work using the collection: this changes the ideal response, and the specific corrections are likely motivated by improving performance for a specific system. It is acceptable to note in a paper where issues in qrels are found, and in some cases to do additional experiments using modified qrels. However, this is only acceptable if results using the 'official' qrels file from a test collection are reported, and the qrels changes are unbiased.[12]

[11]See Fuhr (2017) for a critique of mRR and a proposed replacement, as well as common uses of mAP.

[12]Automated qrel changes are preferred, *e.g.,* using topic and result data. Manual 'cherry-picking' of qrel topics for a model family is weak science.

Qrels and items with unknown relevance. Qrels provide judgements for pooled sources used in assessment. Unless the collection is extremely small, we do not have the relevance judgements for all collection sources per topic: only pooled sources are evaluated. This means that people using an existing test collection often retrieve sources that are unevaluated. This is illustrated for Model A in Example 3.4, where the 3rd formula retrieved was not included in the assessment pool for the query. This unrated formula will be treated differently, depending upon the metrics that we use. By default, unrated hits are considered non-relevant, *e.g.,* the *Precision@5 (P@5)* for Model A giving the percentage of relevant items within the first 5 returned would be 2/5.

Some metrics ignore sources with unknown relevances, such as the Bpref and *prime* ($_\prime$) metrics shown in Table 3.1. For example, if a 6th formula returned by Model A was graded relevant, then the $P_\prime$@5 would be 3/5; if the 6th formula is non-relevant or no 6th formula was returned, $P_\prime$@5 is still 2/5. Ignoring unevaluated items allows evaluation using only graded items in qrels files, and avoids assuming unpooled sources are non-relevant.[13]

Bpref measures the number of consistent rank *preferences* of relevant vs. non-relevant sources for a query. A relevant source's preference is *consistent* will all other sources at lower ranks; a non-relevant source is *inconsistent* with relevant sources at lower ranks. Bpref gives no credit for a relevant source with $\geq |R|$ non-relevant sources above it in the ranking, where R is the number of relevant sources for a topic/query. Relevant sources missing in a ranking are also given no credit.

Question answering metrics. The main metrics for question answering are simpler than for retrieval, as shown in Table 3.2. Most evaluations report the percentage of correct responses (with normalizations/tolerances as described earlier) and/or the number of responses that match the target answers in the test collection exactly. Some test collections report *perplexity* to characterize system uncertainty in making multiple choice question answer selections. Perplexity for the correct answer a_c

[13]These metrics are also helpful in early design, as models can simply rank scored sources. However, metrics obtained this way *cannot* be compared with published systems, because the retrieval step that filters the collection is skipped.

Table 3.2: Question Answering Metrics. a_k: answer for question q_k. A_k: prob. distribution for possible answers to q_k (*e.g.,* multiple choice). **Note:** Metrics are computed using target answers, and text measures are tokenization-dependent.

	Name	**Formula / Reference**	**Description**
Correctness and Uncertainty			
EM	Exact match rate	$\frac{1}{\lvert Q\rvert}\sum_{q_k \in Q} \delta(a_k, q_k)$	Percentage answers identical to target answers (δ returns 0/1)
Accuracy	Correct answer rate	$\frac{1}{\lvert Q\rvert}\sum_{q_k \in Q} e(a_k, q_k)$	Percentage answers within tolerance of function e (returns 0/1)
Perplexity	Avg. correct answer perplexity	$\frac{1}{\lvert Q\rvert}\sum_{q_k \in Q} \frac{1}{P(ac\lvert A_k, q_k)}$	Correct answer prob. uncertainty as #items chosen from randomly
Similarity to Text Answers (including Rationales, Step-by-Step)			
Token Level			
Token F1	Token F1 score	$\frac{2RP}{R+P}$	Harmonic mean: % target answer tokens in response (*R*ecall), % response tokens in target answer (*P*recision)
BERTScore	Token F1 w. token embedding cos similarity	$\frac{2R_{ac}P_{ac}}{R_{ac}+P_{ac}}$	Highest target/response token pair cos. similarities give avg. token similarity for target (R_{ac}), avg. token similarity for response (P_{ac})
N-gram Level			
BLEU	Bi-Lingual Evaluation Understudy	Papineni *et al.* (2002)	$[0, 1]$ score from shared target answer/response n-grams + penalty for short outputs.
sBLEU	Sentence-BLUE		BLEU for individual sentences.
String Level			
Edit distance	String edit distance	Yu *et al.* (2016)	Operation count to convert one string to another (*e.g.,* insert, delete, replace). Can be normalized to $[0, 1]$ using string lengths.

$\delta(a, b) = (a = b)$; $e(a, b) = (normalize(a) = normalize(b))$
BLEU and edit distance have many variations.

in the answer probability distribution A_k for question q_k is converted to a random choice between n options, using $n = 1/P(ac|Ak, qk)$. For example, if a model estimates the target answer is 25% likely to be correct, then the perplexity is $n = 1/0.25 = 4$.

For text responses, open response questions, and comparing explanations or step-by-step solutions against target answers in a collection, text similarity measures are used at the token, n-gram, and string/sequence levels.[14] The token F1 measure is the harmonic mean for the percentages of target answer tokens in the response (*i.e.,* recall), and response tokens in the target answer (*i.e.,* precision).

For embedded tokens, a variation called the *BERTScore* (Zhang *et al.*, 2020b) has been used. This is similar to the token F1 score, but makes use of a trained embedding model (*e.g.,* BERT). All target answer and response tokens are embedded using this model, and the highest cosine similarity between each target answer and response token is first computed. From these maximum similarities, the average cosine similarity for a target answer to a response token is used for recall, and the average cosine similarity from a response token to a target answer token is used for precision. F1 is then computed as before. Embeddings capture token context missing in the token F1 measure, *e.g.,* this can avoid penalizing synonyms, but the token F1 measure is more easily computed and interpreted.

Similarity based on token sequences such as n-grams or full strings can also be used. *BLEU* was originally developed to measure the success of translations by comparing a translation against one or more accepted translations of a sentence (Papineni *et al.*, 2002). It computes the similarity of n-grams (*i.e.,* token sequences of a fixed length, for different values of n) between a response and target answer(s), with an additional penalty for short answers. *sBLEU* modifies this to compute similarity at the sentence level, rather than for complete responses. Edit distance considers an entire token sequence, and computes the number of operations from a fixed set needed to transform one to the other (*e.g.,* insert, delete, replace, Yu *et al.*, 2016).

For the text similarity measures, it is important to realize that they reflect surface structure or correlations learned by a model, and do not directly quantify *semantic* similarity. These metrics are certainly correlated with semantic similarity, but they only indirectly capture differences in *information content* between responses and target answers. What they actually measure is how closely one string imitates the other based on tokens or token embeddings. They are still very useful, but one has to be a bit careful about their interpretation.

[14]Note: All string-based measures, including token F1 and BERTScore are affected by the method used to split words into tokens.

3.5 System Comparisons and Statistical Tests

Imagine that we have two math search systems returning 10 results per query for a test collection with binary relevance grades. We determine that the average P@10 is 50% for both systems. However:

1. For the first system, 5 of 10 results are relevant for every query (*i.e.,* every query has P@10 of 50%).
2. For the second system, half of the queries return *no* relevant sources (*i.e.,* P@10 of 0% per query) while the remaining half return only relevant sources (*i.e.,* P@10 of 100% per query).

While the average P@10 scores are identical, we would probably much prefer the first system because it is more consistent and avoids missing relevant answers altogether.

To capture variance in our evaluations, such as for rank metrics or differences between target and provided numeric answers to questions, we want to compare *distributions* (*i.e.,* sets) of values rather than simply averages. Statistical *hypothesis tests* are used to check whether differences in average measures are likely to be stable when running additional queries. They include an estimate for the probability of detecting a difference incorrectly (*i.e.,* a *Type-1 error*) given as the *p-value*. Generally we consider a *p-value* less than either 5% or 1% (*i.e.,* $p < 0.05$ or $p < 0.01$, chosen *before* running an experiment) to be a 'statistically significant' difference suggesting that the averages are *unlikely* to be the same after running a large number of additional queries/questions.[15] Note that hypothesis tests are probabilistic estimates, and not certain answers regarding whether averages are *actually* different in the limit. It is not possible to run all possible queries/questions to know for certain. Despite this limitation, statistical tests provide a more rigorous and nuanced characterization of differences in metric values than comparing average values directly.

Commonly used statistical hypothesis tests for performance metrics include the standard t-test for comparing two distributions, and the Bonferonni *corrected* t-test when comparing two or more models to a single

[15]Important note: using 'significantly improved' or 'significantly different' without a hypothesis test is a short path to having a research paper rejected.

baseline system. The correction here adjusts computed p-values when multiple comparisons are made, because without correction the probability of detecting a difference increases with additional comparisons. Many other tests and comparison types are also used. The selection of a chosen measure or test is motivated by the goal of a comparison, variable data types, and data distribution assumptions (*e.g.,* correlation coefficients, χ^2 ('chi-squared'), and Wilcoxon rank sum tests).

It is also very important when comparing two systems to check raw metric values, and to examine the specific topics where performance differs substantially. For example, visualizing raw metric data can reveal whether metrics are similar across queries/questions, or vary dramatically (*e.g.,* for the P@10 example from the start of this section). One simple approach is to sort the metric values and then produce a 'ski jump' bar graph. Specific queries/questions where larger differences in metric values are seen can help identify specific limitations, patterns of behavior, and information use by the models. Equally importantly, this also helps identify bugs in system implementations, including where computed metrics are unusually strong, but *not* because the model is effective.[16] For search tasks, frameworks like PyTerrier[17] provide easy access to query-specific differences between models, and can be used to compute common statistical tests. Additional helpful evaluation tools include `trec_eval`,[18] `pytrec_eval`[19] and `ranx`.[20] For QA tasks, frameworks such as `nltk`[21] can be used to compute standard text metrics. Standard data matrix tools (*e.g.,* Pandas[22]) and statistical tools (*e.g.,* Scipy stats[23]) can also be used to compile descriptive statistics and compute hypothesis tests.

[16]Based on a true story. Or three.
[17]https://pyterrier.readthedocs.io/en/latest.
[18]https://github.com/usnistgov/trec_eval.
[19]https://github.com/cvangysel/pytrec_eval.
[20]https://github.com/AmenRa/ranx.
[21]https://www.nltk.org/.
[22]https://pandas.pydata.org.
[23]https://docs.scipy.org.

4

Formula Search

As described in Section 2, the information conveyed in a formula is primarily structural, representing a hierarchy of operations over arguments. This hierarchy can be represented in an operator tree (OPT) obtained by mapping symbol layout to an operation hierarchy. For brevity and clarity, authors often assume that readers are familiar with common operations and variable types for the subject area they are writing upon. The full information that a formula conveys and is associated with includes these notation conventions and related information presented in surrounding text, other formulas, and even other graphics (*e.g.,* tables or figures). Considering these pieces of context is very helpful for formula search.

With that said, there are certainly situations where searching for isolated formulas is helpful. This includes defining unfamiliar notation, re-finding sources using part of a formula (*e.g.,* `ctrl-f` for formulas), browsing through variations of a formula (*e.g.,* loss functions using the cross-entropy loss), identifying applications in different domains (*e.g.,* medicine vs. computer science), and formula autocompletion.

Figure 4.1 illustrates the information tasks used in formula search, using the model from Section 1.[1] We assume that formulas have already been indexed using one or more representations (*e.g.,* sparse and/or dense: OPT, SLT, visual-spatial, etc.) as described in Section 2.

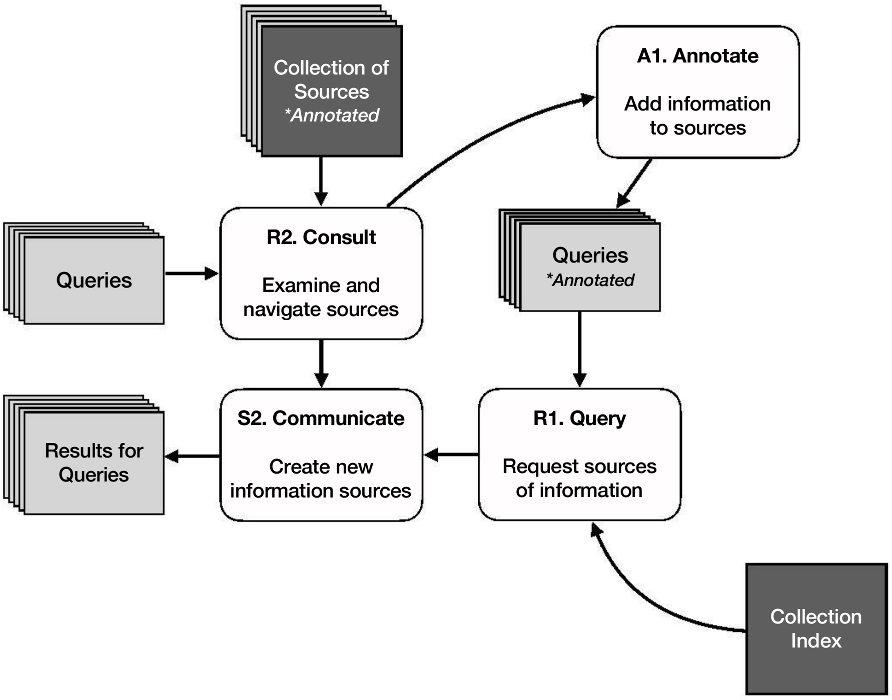


Figure 4.1: Information Tasks Performed in Formula Search. Prior to search, formula patterns in a collection of sources (*e.g.,* OPT and SLT paths) are enumerated or embedded in vectors. These patterns annotate formulas and provide lookup keys in the collection index. Formulas in the index with patterns identical to the query (sparse retrieval) or similar to the query (dense retrieval) are selected, ranked, and then communicated to the user in a new source (*e.g.,* search result page).

In this section, we first present test collections used for developing and evaluating formula search. We then present formula search models organized by the formula representations they use, and then summarize their effectiveness on the test collections. State-of-the-art models use more than one formula representation. This is because dense retrieval

[1]Communicating formula search results is important, but little studied.

models can more flexibly match related formulas using abstract/latent contextual patterns, while concrete patterns (*e.g.*, SLT and OPT paths) are better for retrieving highly similar formulas using specific symbols and structures.

4.1 Test Collections for Formula Search

Relevance definitions for formula search differ based on information needs, and formula search relevance definitions have evolved over time in test collections. There are two basic formula search tasks that have been explored, which differ in their consideration of context.

1. **Isolated:** structural similarity of query vs. candidate formulas (SLT and/or OPT), sometimes with optional wildcard symbols
2. **Contextual:** formula relevance depends upon text where query and candidate formulas appear

Both tasks are illustrated in Example 3.1 from the previous section.

NTCIR: NTCIR-10 was the first shared math-aware search task (Aizawa *et al.*, 2013). It had a formula search subtask, in which systems needed to retrieve formulas similar to a given formula query. The collection included 100,000 technical papers from arXiv (mathematics, physics, and computer science) with 35.5 million formulae.

In NTCIR-11 the formula search task was a known-item retrieval task (Aizawa *et al.*, 2014). The query was either identical to a specific formula instance in a Wikipedia article, or a version with wildcard replacements for subexpressions. Systems were evaluated based on ranks for target formula instances. A Wikipedia page collection with mathematical formulas was used, which was much smaller than the NCTIR-10 arXiv collection.

NTCIR-12 introduced the Wikipedia Formula Browsing (WFB) task that is similar to NTCIR-10: retrieve relevant formulas for a formula query (Zanibbi *et al.*, 2016a). NTCIR-12 uses 319,689 articles from English Wikipedia with over 590,000 formulae in the corpus.

All three test collections use lab-generated topics. In NTCIR-10, 21 formula queries were chosen by the organizers for arXiv papers, of which 18 queries included wildcards and 3 were concrete queries. NTCIR-11 had 100 queries, with 59 including wildcards, and 41 concrete queries without wildcards. Queries were randomly sampled from Wikipedia pages and then modified to include query variables. The NTCIR-12 task had 40 queries, divided into 20 concrete queries and 20 wildcard queries. The wildcard queries are created by replacing one or more sub-expressions in each concrete formula query with wildcards. The intent was to observe differences in retrieval behavior when wildcards were added to queries.

Different pooling processes are used in each NTCIR collection (Mansouri *et al.*, 2021a). No pooling was needed for NTCIR-11, because retrieval targets were specific formulas. In contrast, every formula instance was treated as a separate source for the NTCIR-12 WFB. This led to limited diversity in the judgement pools after selecting the top-20 instances from each submitted run. For example, for the query β (a short formula consisting of a single symbol), every formula instance in the pool of formulas to be judged was β.

Both the NTCIR-10 and -12 test collections use graded relevance (0-2): 2: Relevant (R), 1: Partially Relevant (P), or 0: Non-relevant (N). For NTCIR-10, the assessors were mathematicians or math students who viewed each formula instance from the judgment pool in isolation, considering the query-specific scenario and judgment criteria specified for the query. For the NTCIR-12 task there were two groups of assessors, with each group independently judging pooled formulas. One group was computer science graduate students, and the other was computer science undergraduates. Pooled formula instances were shown to the assessors in context by highlighting them in sources, but assessors were not asked to interpret the pooled formula in that specific context. Instead, the assessment was to done based on the pooled formula alone.

For each topic in NTCIR-11, the single Relevant (R) formula instance was defined as the formula instance that had been used as the formula query. Note that there may 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). NTCIR-11 used mean reciprocal rank (mRR), which is appropriate for single retrieval targets per topic.

For assessment, NTCIR-10 and -12 combined the judgments from two assessors to form a 5-level "Aggregate" relevance score. This was done by summing the two scores from assessors for each pooled formula. Relevance scores ranged from 0 (both assessors judged N) to 4 (both assessors judged R). To compute evaluation measures the 4 level-relevance is binarized, by treating scores of 0-2 as non-relevant, scores of 3-4 as relevant.

NTCIR-10 reported P@5, P@10, P@hit (i.e., for all returned results), and MAP. NTCIR-12 uses P@k for $k = \{5, 10, 15, 20\}$. Later, researchers used Bpref (Buckley and Voorhees, 2004) to avoid penalization for unevaluated formulas.

ARQMath: The Answer Retrieval for Question on Math (ARQMath) lab introduced a *contextualized* formula search task illustrated in Example 3.1. The test collection was developed over three years, generating test collections referred to as ARQMath-1 (2020), ARQMath-2 (2021), and ARQMath-3 (2022).

ARQMath's collection consists of question and answer posts from a math community question-answering website, Math Stack Exchange (MSE). These question posts provide a diversity in subject areas and required mathematical expertise, ranging from simple questions from high school to advanced topics. Formula queries are taken from question posts, and the task is to find relevant formulae inside other question and answer posts. All MSE questions and answers posted from 2010 to 2018 are used as the collection of sources. Formula queries for topics were selected from questions posted in 2019 (ARQMath-1), 2020 (ARQMath-2), and 2021 (ARQMath-3). Additional training topics are provided in the test collection.

To make topics diverse, ARQMath attached a complexity label to topic formulas, dividing them into low, medium, and high-complexity topics. Additional details on topic selection can be found elsewhere (Mansouri *et al.*, 2021a). To avoid the lack of diversity in pooled formulas seen in NTCIR-12, ARQMath pooling selects *visually distinct* formulas: two formulas are *visually distinct* if their Symbol Layout Trees differ. The canonicalized SLT representation from Tangent-S (Davila and Zanibbi, 2017a) was used to identify visually distinct formulas when two formulas are parseable, or had identical LaTeX strings otherwise.

For each visually distinct pooled formula, up to five instances of that formula were shown to the assessors. Example 3.5 shows the Turkle[2] interface used for assessment. As shown in the left panel of the figure, the formula query $\sum_{k=0}^{n} \binom{n}{k} k$ is highlighted in yellow. The assessors can use the question post 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. The final relevance score for a formula is the *maximum* relevance score for any judged instance of that formula.

While the official evaluation using visually distinct formula pools, ARQMath introduced the use of "Big Qrel Files", where nearly all assessment data is provided. This includes assessment for each individual formula instance, along with assessor ID. This can be used to study effectiveness of formula search models, under assessment of different people, and to *change* how final relevance scores are defined (*e.g.,* using average rather than maximum relevance scores).

ARQMath organizers hired undergraduate and graduate in mathematics or with strong mathematical backgrounds to act as assessors. Each year, the assessors were trained by a math professor during three training sessions. The sessions included discussing relevance ratings for practice topics, with the goal of reducing variation in ratings across assessors, and minimizing assessment errors. After some discussion between organizers and assessors in ARQMath-1, relevance for retrieved formulas was defined 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 assigned each formula instance in the judgment pool one of four scores as defined in Table 4.1. 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 review the question post containing the query (and, optionally,

2https://github.com/hltcoe/turkle.

Table 4.1: Relevance scores and definitions for ARQMath Formula Search task.

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

the thread containing that question post) to understand the searcher's information need. Here the question post fills a role akin to Borlund's simulated work task (Borlund, 2003), although here the title, body, and tags from the question post are included in the topic and may be used by retrieval systems. Assessors also consult the posts where retrieved formula instances come from (these may be question or answer posts), along with the associated thread to see whether the formula would have been a useful basis for a search, *i.e.,* how likely useful content would be found if this or other instances of the retrieved formula were returned by a search engine.

The ARQMath organizers did make one change to the way this relevance definition was interpreted for ARQMath-2 and -3. ARQMath-1 assessors were instructed during training that if the query and candidate formulas had the same appearance, then the candidate was highly relevant. For ARQMath-2 and -3, the interpretation of 'exact match' was clarified to take the formula semantics and context into account. 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 be considered 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 both formulas refer to the same concept. For the formula query $\frac{S}{n} \geq \sqrt[n]{P}$ (ARQMath query B.277), formula $\frac{1+2+3+\ldots+n}{n} \geq \sqrt[n]{n!}$ has medium relevance. Both formulas are referring to the AM-GM inequality (of Arithmetic and Geometric Means).

System evaluation is performed after removing duplicate instances of visually identical formulas from search results, and then calculating effectiveness measures over the ranked series of visually distinct formulas. This is done by replacing each formula instance with its associated visually distinct formula id, and then removing duplicates starting from the top of the ranking. To avoid earlier issues with unevaluated hits for people using the test collection after assessment was complete, the organizers chose the nDCG' measure (read as “nDCG-prime”) introduced by Sakai (2007) as the primary measure. The nDCG measure on which nDCG' is based is widely used when graded relevance judgments are available. ARQMath also uses two other measures: Mean Average Precision (MAP'), and Precision at 10 (P'@10), after removing the unjudged hits. For MAP' and P'@10 High+Medium binarization is used, meaning only the medium and high relevance ratings (2 and 3) were considered relevant.

AccessMath. As an example of a very different (albeit small) test collection, the AccessMath system described in the first section (Davila and Zanibbi, 2018) was developed using lecture videos and LaTeX lecture notes produced for those lecture videos.[3]

4.2 Formula Retrieval Models

We have organized formula retrieval models by the formula representation they use for search. Some of the representations imply sparse vs. dense representations as noted below. A more detailed discussion of formula representations can be found in Section 2. The formula representation types we distinguish here are:

Text-Based: Formulas represented by tokens in text encodings (*e.g.,* LaTeX tokens). Early systems used this with traditional sparse (*i.e.,* inverted index-based) retrieval models such as TF-IDF.

Tree-Based: Use formula tree representations (*e.g.,* SLT and OPT). Retrieval is performed over sparse tuple indexes for substructures (*e.g.,* paths, subexpressions) and/or (re-)ranking by tree edit distance or graph alignment.

Visual-Spatial: Captures formula appearance symbolically without writing lines (*e.g.,* in SLTs) or operation-argument relationships (*e.g.,* in OPTs).

[3]Notes: https://www.cs.rit.edu/~dprl/files/TangentV-data_results.zip, Videos: https://www.cs.rit.edu/~accessmath/am_videos.

Embedding: Text, tree, or visual-spatial representations for formulas and/or subexpressions are embedded in vector spaces. Nearest-neighbor search using vector similarity identifies candidates.

Other: Use images or other representations not described above.

Example 4.1 compares the formulas $y = x^2$ and $y = x$ in different representations. At top-left we see a text based representation, where we have two lists of tokens produced by linearizing SLTs represented in LaTeX. At the top-right we have operator and symbol layout trees at left and right respectively. Finally, at the bottom we see the formulas represented as vectors (points) in a 3d embedding space. Such a representation can be created from text, tree, or other representation. The specific positions of vectors depend upon training data and the training tasks, learning algorithms, and loss functions used for embedding.

Example 4.1: Formula comparison in different representations.

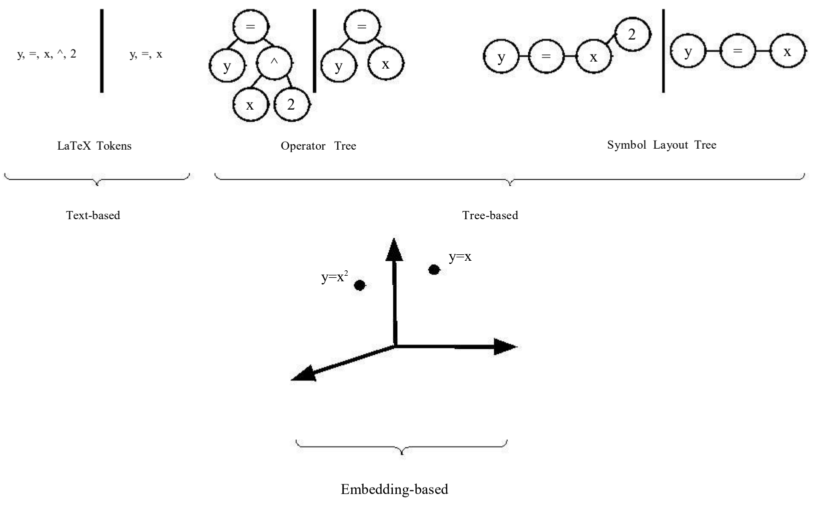


Table 4.2 shows the first formula returned by models using different formula representations. Each model has a ‘reasonable’ first hit. While it is helpful for identical or near-identical formulas to be highly ranked, matching identical/nearly-identical formulas is easy for reasonably expressive representations and indexing patterns.

Table 4.2: The first formula returned by different search engines for three queries of increasing structural complexity.

		Query		
Model	**Repr.**	$\boldsymbol{Cov(x,y) = 0}$	$\int_0^1 \frac{\sin^{-1}(x)}{x}$	$\boldsymbol{x! = \sqrt{2\pi x} * \left(\frac{x}{e}\right)^x}$
MathDowsers	Tree paths (SLT)	$COV(X,Y) = 0$	$J = \int_0^a \frac{\sin^{-1}(x)}{x} dx$	$x! \approx \sqrt{2\pi x} \cdot \left(\frac{x}{e}\right)^x$
Approach0	Tree paths (OPT)	$Cov(x,y) = 0$	$\int_0^1 \frac{\sin^{-1}(x)}{x} dx = \frac{\pi}{2} \ln 2$	$x! \sim \sqrt{2\pi x}\left(\frac{x}{e}\right)^x$
Tangent-CFT	Tree embed.	cov(y,x) = 0	$\int_1^\infty \frac{\sin^2(x)}{x}$	$n! = \sqrt{2\pi x}\left(\frac{x}{e}\right)^x$
XY-PHOC	Visual-spatial	$Cov(x,y) = 0$	$\int_0^1 \frac{\sin(x)}{x} dx$	$n! = \sqrt{2\pi x}\left(\frac{x}{e}\right)^x$

Effectiveness-wise, the ability to capture relevance for formulas that are progressively more distinct from the query is what differentiates most formula search models. This is one of the reasons that the ARQMath test collections used full rank metrics for ranking formula search systems (*i.e.*, $nDCG'$, with mAP' added for comparison), in addition to observing metrics focused on the top of a ranking (*e.g.*, P'@10, mRR).

A summary of formula search models is provided in Table 4.3. In the remainder of this section, we will discuss the families of formula search models based on their representation types.

4.3 Text-based and Tree-based Models

It is common to use traditional sparse retrieval models for more complex domains such as math. Particularly in the early days of formula search, traditional token-based sparse models such as TF-IDF were used. An example is one of the earliest large-scale formula search engines created for the Digital Library of Mathematical Functions (DLMF) (Miller and Youssef, 2003). LaTeX is parsed into an SLT-type tree, which is then linearized after normalizing token symbols. Normalizations include converting symbols to text tokens, and mapping non-alphanumeric characters to alphanumeric strings. For instance, $x_{t-2} = 1$ given as '`x^{t-2}=1`' is converted to the token sequence:

$$x,\ \ BeginExponent,\ \ t,\ \ minus,\ \ 2,\ \ EndExponent,\ \ Equal,\ \ 1.$$

Table 4.3: Formula Search Models. 'Others' representations includes images.

	Representation				**Canonicalization**			
Model	SLT	OPT	Others	Context	Unif.	Norm.	**Rank**	**References**
Text								
DLMF			✓			✓	TF-IDF	Miller and Youssef (2003)
ActiveMath			✓	✓			Tokens	Libbrecht and Melis (2006)
MathDex	✓				✓	✓	TF-IDF	Miner and Munavalli (2007)
EgoMath	✓			✓	✓	✓	TF-IDF	Mišutka and Galamboš (2008)
MIaS	✓			✓	✓	✓	TF-IDF	Sojka and Líška (2011)
LCS			✓		✓	✓	LCS.	Pavan Kumar *et al.* (2012)
Tree								
MathWebSearch		✓				✓	Paths	Kohlhase and Sucan (2006)
WikiMirs		✓	✓	✓	✓	✓	TF-IDF	Hu *et al.* (2013)
SimSearch	✓						TED	Kamali and Tompa (2013)
MCAT	✓	✓		✓	✓	✓	Paths	Kristianto *et al.* (2016a)
Tangent-3	✓			✓			Paths	Zanibbi *et al.* (2016b)
Tangent-S	✓	✓		✓			Paths	Davila and Zanibbi (2017a)
Tangent-L	✓			✓			BM25+	Fraser *et al.* (2018)
Approach0		✓		✓	✓	✓	Paths	Zhong and Zanibbi (2019)
MathDowsers	✓			✓	✓	✓	BM25+	Ng *et al.* (2020)
Tangent-CFTED	✓	✓					TED	Mansouri *et al.* (2020)
Embedding								
SMSG5	✓		✓	✓	✓		Cosine	Thanda *et al.* (2016)
Formula2vec			✓	✓			Cosine	Gao *et al.* (2017)
EqEmb.	✓			✓			Cosine	Krstovski and Blei (2018)
Tangent-CFT	✓	✓			✓		Cosine	Mansouri *et al.* (2019a)
NTFEM			✓				Cosine	Dai *et al.* (2020)
Semantic Search	✓						Cosine	Pfahler and Morik (2020)
Forte		✓					Cosine	Wang *et al.* (2021)
MathEmb		✓		✓	✓	✓	Cosine	Song and Chen (2021)
MathBERT		✓		✓			Cosine	Peng *et al.* (2021a)
MathAMR		✓		✓			Cosine	Mansouri *et al.* (2022d)
Visual-Spatial								
Tangent-V			✓				Tokens	Davila *et al.* (2019)
XY-PHOC			✓				Cosine	Avenoso (2021)
EARN	✓	✓	✓				K-NN	Ahmed *et al.* (2021)
Other								
TanAPP	✓	✓			✓	✓	Ens.	Mansouri *et al.* (2019a)
Math-L2R	✓	✓			✓	✓	SVM$_{rank}$	Mansouri *et al.* (2021b)
MathAPP		✓		✓	✓	✓	Ens.	Peng *et al.* (2021a)
FORTEAPP		✓			✓	✓	Ens.	Wang *et al.* (2021)

A second normalization is canonical orderings: for commutative operations where argument order is unimportant (*e.g.*, multiplication and addition), a fixed ordering is produced using the lexicographic order of argument tokens. After linearization, DLMF creates an inverted index used with TF-IDF scoring of query tokens in the same manner as text.[4]

Approaches like DLMF later included additional canonicalization steps. For example, EgoMath (Mišutka and Galamboš, 2008) canon-

[4]Text and formula tokens are stored and retrieved from the same index, using a unified token representation.

icalizes argument ordering, and enumerates variables and constants, using identical symbols to capture variable repetitions.[5] For constants, formula $74 + a^2 + b^2$ is also indexed as $const + a^{\,const} + b_{const}$. With variable normalization, formula $a - b$ is also indexed as $id1 - id2$. Other normalizations such as removing brackets using distributivity rules are also applied. The goal in these normalizations is to increase recall by increasing the number of formulas with similar token representations.

Aside from sparse retrieval, some other approaches such as using the Longest Common Subsequence (LCS) of a string (Pavan Kumar *et al.*, 2012) have been used to produce similarity scores. As before, formulas are canonicalized before applying LCS so that each function, variable, and number is mapped to a unique token, and constants and variables are enumerated.

Tree-based models. As discussed in Section 2, we normally use graphs to represent structured data, and specifically for formulas, trees to capture a hierarchy of writing lines in SLTs, and a hierarchy of mathematical operations and arguments in OPTs.

Tree-based approaches can be categorized into two main groups: part-based, and full-tree matching. One of the earliest part-based models is MathWebSearch (Kohlhase and Sucan, 2006) that relies on subexpression indexing used originally to unify terms in theorem provers (substitution indexing trees, Graf, 1995). Using operator trees, relationships between progressively more concrete formulas are produced by a series of variable substitutions. A search for expressions with similar operator structures and operands starts from the lowest-precedence operators. Nodes in the substitution indexing tree correspond to expressions with common structures at the top of their operator trees. Moving from the root to the leaves of the substitution tree yields increasingly concrete expressions (*i.e.,* after more variable replacements).

Another category of tree-based models represent formula tree substructures. The Math Indexer and Searcher (MIaS) (Sojka and Líška, 2011; Ruzicka *et al.*, 2016) system uses Presentation MathML, encoding subtrees as compact strings. For example, $a + b$ is represented by

[5]See Section 2 for discussion of symbol enumeration.

math (mi(a) mo(+) mi(b)). A similar approach that uses subtrees of differing structural complexity is WikiMirs (Hu et al., 2013). WikiMirs creates terms (patterns) for search from SLTs by recursively replacing subexpressions with wildcards. For example, the formula $(x+3)\times\frac{a}{b}$, is tokenized into 4 concrete terms, and four generalized terms with wildcards for argument subexpressions:

$$\begin{array}{lll} \text{Concrete terms} & : & \{\,(x+3)\times\frac{a}{b},\quad (x+3),\quad \frac{a}{b},\quad x+3\,\} \\ \text{Generalized terms} & : & \{\,(*)\times *,\quad (*),\quad \frac{*}{*},\quad *+*\,\} \end{array}$$

Term construction is performed recursively until no new terms can be produced. Unique tokens are enumerated, and then used to create an inverted index that is searched using TF-IDF. This system was later extended, incorporating text keywords and using operator trees (Gao et al., 2016).

MCAT (Kristianto et al., 2016a) improved part-based retrieval by encoding path and sibling information in symbol layout and operator trees. Tuples capturing tree paths are used for retrieval patterns in an inverted index. In addition to the path-based lookup, this model also uses a hashing-based formula structure encoding scheme, and also includes text at three levels of granularity. The first level considers words around a formula within a context window of size 10, along with descriptions and noun phrases in the same sentence as the formula. The second level includes all words from the paragraph where the formula appeared. At the third level, the title, abstract, keywords in the document, descriptions of all the formulas, noun phrases, and all words in the document are considered. A formula query combines lookup up in multiple inverted indexes for both formula and text representations. This was perhaps the first model to capture surrounding context for formulas in a detailed manner.

Tangent. Tangent-3 (Zanibbi et al., 2016b) is a two-stage part-based retrieval model. From an SLT, path tuples are generated in the form of $(s_1, s_2, R, \#)$ with parent symbol s_1, child symbol s_2, the spatial relationship sequence R from s_1 to s_2, and a count used to capture repetitions (#). These tuples are used to identify an initial set of top-k candidates using a sparse bag-of-words model, scoring by F1 (i.e., harmonic mean of tuples matched on the query and a candidate formula, also known as the dice coefficient). Top candidates are then re-ranked using full-tree matching, aligning the query SLT to each candidate SLT. After alignment, each top-k candidate is scored using the harmonic mean of symbol and relationship recall (the Maximum Subtree Similarity (MSS)) and two tie-breakers: symbol precision after unification, and symbol recall without unification.

The symbol layout tree representation developed for Tangent-3 has been used in a number of retrieval models. The model includes a container object for matrices, tabular structures, and parenthesized expressions, as well as explicit whitespace, and variable and operation types attached to names (e.g., N!x for the number x).[6] The Tangent-S model later included retrieval using both symbol layout and operator trees (Tangent-S, Davila and Zanibbi, 2017a). In operator trees, commutative and non-commutative operators have node type (U!) and (O!) for unordered and ordered operations, respectively. Tangent-L (Fraser et al., 2018) improved retrieval results further through richer indexing patterns/features, and scoring with language statistics using BM25+ (Lv and Zhai, 2011).

The Tangent-L tuple generator was later re-implemented in the MathDowsers system (Ng et al., 2020; Ng et al., 2021; Kane et al., 2022). The new generator adds additional patterns for repeated symbols, and additional normalizations. Normalization rules are defined to support operation ('semantic') matches. For example, for commutative operators ($A + B$, $B + A$) and symmetry ($A = B$, $B = A$) the order of adjacent symbols is ignored. Using a canonical symbol for operator equivalence classes, the model also canonicalizes alternative notations ($A \times B$, AB), operator unification ($A \prec B$, $A < B$), and inequality equivalence ($A \leq B$, $B \geq A$). This captures OPT-type relationships in an SLT representation.

Approach0. Approach0 is a state-of-the-art formula retrieval model that uses OPT leaf-root paths in an inverted index within a two-stage model for retrieving operator trees (Zhong and Zanibbi, 2019). An illustration of OPT leaf-root paths is shown in Example 2.11. OPTs are generated from LaTeX using a small but robust expression grammar. To

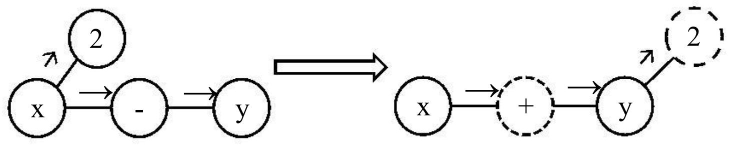
[6]See Zanibbi et al. (2016b) for details.

boost recall, variable enumeration is applied. Like Tangent-3/-S, retrieval is performed in two steps. Candidates are first retrieved using matching leaf-root paths in a sparse index, and then re-ranked using matches of up to three largest common subtrees identified via dynamic programming. Similarity is scored by a weighted sum of matched leaves (operands) and operators from the common subtrees. In the later version of this system text context is used (Zhong et al., 2021). A textual similarity score is produced using Lucene BM25, and formula structure-based scoring uses the IDF of paths and symbol similarity. These scores are combined in a linear combination.

Full tree matching and tree-edit distance (TED). In addition to tree alignments used for reranking in Tangent-3/-S and Approach0, full-tree matching from tree-edit distances (TED) have been used. Tree edit distance generalizes string edit distance, defined by the number of operations needed to convert one tree to the other. The SimSearch model uses tree-edit distance (TED) on SLTs directly as the similarity measure (Kamali and Tompa, 2013). Three editing operations are used: insertion, deletion, and substitution. Example 4.2 shows operations converting the SLT for x^2-y to $x+y^2$. Accelerations such as cost-based pruning of candidates and caching sub-trees can be used. In SimSearch operation costs are defined using the similarity of node labels, a node's parent's label, and whether they are leaf nodes. The final ranking is the inverse edit distance normalized by tree sizes, as given in Equation 4.1.

$$\mathrm{sim}(E_1, E_2) = 1 - \frac{\mathrm{dist}(T_1, T_2)}{|T_1| + |T_2|} \tag{4.1}$$

Example 4.2: Converting SLT $x^2 - y$ to $x + y^2$ in three edits:
Delete 2, replace $-$ by $+$, and add 2 as superscript of y.



Tangent-CFTED reranks results from a path-based dense retrieval model using tree-edit distances (Mansouri et al., 2020). Unlike SimSearch where edit operation weights were defined using heuristics, here weights are learned for each edit operation. The model uses inverse edit distances for scoring, as shown in Equation 4.2.[7] Tangent-CFTED uses both symbol layout and operator trees, and the final ranking score is a weighted combination of individual rank scores.

$$\mathrm{sim}(E_1, E_2) = \frac{1}{\mathrm{TED}(T_1, T_2) + 1} \tag{4.2}$$

4.4 Dense Retrieval with Formula Tree Embeddings

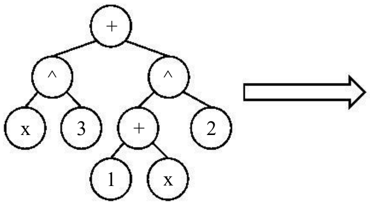
As for information retrieval and natural language processing in general, researchers working in math IR turned to embedding models to avoid the types of vocabulary problems that traditional sparse model have, and to make greater use of context in patterns used for matching, as discussed earlier. Early text embedding models such as Word2Vec (Mikolov et al., 2013) produced revolutionary results for text problems. These models were then extended to graph data types. The earliest approaches to graph embddings were simple: linearizing a graph using different traversals, treat nodes as tokens, and then apply Word2Vec. Graph embedding models of this type include Node2Vec (Grover and Leskovec, 2016) and DeepWalk (Perozzi et al., 2014).

SMSG5 was the first known embedding model for math formulas, and used for re-ranking text-based sparse retrieval results (Thanda et al., 2016). For first-stage retrieval formulas in Presentation MathML (SLTs) are linearized and indexed as keywords in a sparse index using ElasticSearch. For re-ranking the doc2vec embedding model was used (Le and Mikolov, 2014) to covert binarized expression trees into real-valued vectors. Each operator and its operands are treated as tokens for the doc2vec model, and linearized using an in-order traversal. If the operand is a subexpression rather than a symbol, a token identifier for the subexpression is used. Example 4.3 shows tokens extracted for formula $x^3+(1+x)^2$. The tuples/patterns produced are similar to those used for other tree part-based retrieval models described above, and includes variable and subexpression enumeration. The final similarity

[7] 1 is added to the denominator to avoid division by zero.

score for reranked formulas is the cosine similarity of the query and candidate vectors.

Example 4.3: Binary OPT for $x^3 + (1 + x)^2$ with tokens generated for operator nodes. MiAS, SMSG, WikiMIRS and other models index similar patterns.



Token	ID
x,^,3	id1
1,+,x	id2
id2,^,2	id3
id1,+,id3	id4

Tangent-CFT (Mansouri et al., 2019a) was the first dense retrieval model to use both symbol layout and operator trees for formula embeddings, using an approach similar to SMSG5. Tangent-S is used to generate OPT and SLT path tuples. Canonicalization is performed using enumeration of variables and constants, and in later versions, operator types. The modified tuples are enumerated and grouped into n-grams, which are then embedded individually. This process is shown in Figure 4.2, where both the formula query and candidate formula go through the same pipeline to generate the vector representations. The novelty of this model lies in using an n-gram embedding model, fastText (Bojanowski et al., 2017). This approach is better suited for queries not seen in the collection, as it represents formulas using subexpressions. n-gram vector representations for formulas were averaged to obtain the final embeddings used for retrieval, with ranking by cosine similarity. In the early version of this model, the vectors of different representations (SLT, OPT, Unified SLT) were averaged to get the final vector for a formula. Later, this was converted to combining retrieval results from each representation using a modified Reciprocal Rank (Mansouri et al., 2020). A similar approach was applied on other representations such as N-ary trees in the N-ary Tree-based Formula Embedding Model (NTFEM, Dai et al., 2020).

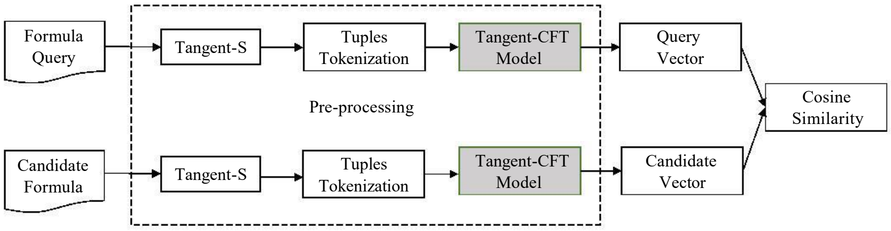


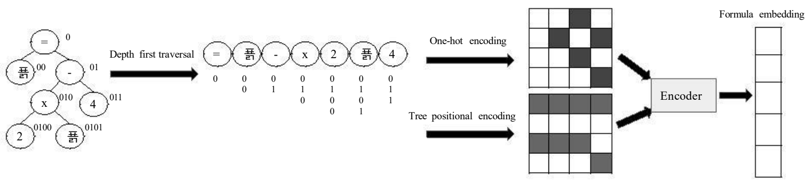
Figure 4.2: Retrieval with Tangent-CFT model. Query and candidate formulas are passed to the same pre-processing pipeline to extract their vector representations. The cosine similarity between these two vectors is the similarity score.

The approaches seen so far for embedding use linearized tree representations and apply sequence embedding models. As formulas are more naturally represented as trees, graph convolutional neural networks are well-suited to formula embedding. The Semantic Search model generates graph representations from Presentation MathML (SLTs), and derives context features from tags, attributes, and text (Pfahler and Morik, 2020). These features were then used to represent nodes as one-hot encoded vectors. A graph convolutional neural was trained using two unsupervised tasks: 1) a contextual similarity task where labels are generated from the surrounding contexts of mathematical expressions, 2) a self-supervised masking task. Other graph-embedding approaches have considered using symbol layout and operator trees. For example, MathEmb uses operator trees with a Graph Convolution Network, Graph SAmple and aggreGatE (GraphSAGE), and a Graph Isomorphism Network (Song and Chen, 2021).

EARN (Ahmed et al., 2021) is a multimodal embedding model that takes advantage of both image and graph formula representations. An image encoder uses a formula image rendered from LaTeX passed to a ResNet model (He et al., 2016), followed by a Bi-LSTM to produce an image embedding. For the graph representations, a message-passing-based graph encoder is used. The distances between graph-based and image-based embeddings are used as patterns for retrieval. The visual and graph-based similarities are combined using a linear combination in a manner similar to the Tangent-S system.

Encoder-decoder architectures have also been used for formula search. Similar to NLP tasks, reconstruction (also known as the ‘fake task’) where a formula must be decoded from an embedded vector can be used to train these architectures. After training, only the formula encoder is needed for embedding formulas. FOrmula Representation learning via Tree Embeddings (FORTE) uses this architecture by taking an operator tree as input, generating the vector embedding, and then reconstructing the formula in the decoder (Wang et al., 2021). The encoding process is shown in Example 4.4. On the encoder side, trees are traversed depth-first, with each node represented by an embedding. To preserve formula structure, a positional encoding in a fixed-length vector is concatenated to each node embedding. The positional vector represents the binary branching path from the root to a node in the tree. On the decoder side, this model uses a novel tree beam search generation algorithm to reconstruct a slightly different version of the input tree, with attached ‘end’ nodes.

Example 4.4: FORTE encoding process for formula x = 2x − 4.



Some embedding models include the textual context surrounding a formula. Early attempts include an embedding model generating embeddings for words and formulae using textualized formulas (Krstovski and Blei, 2018). Linearization is done using SLT tuples from the Tangent-3 system, after which Word2Vec is used within a larger context window size for formulas than words. A similar idea was adopted in the early days of BERT transformer models (Devlin et al., 2019), which were trained on general text. Later models brought attention to the need for specific tokenizers for math. Note that directly fine-tuning a BERT-based model for formula search is not ideal, as their tokenizers (byte pair encoding or WordPiece) are trained using general text, and may not handle formulae correctly (e.g., by splitting LaTeX commands).

MathBERT pre-trains a BERT model using two tasks: Masked Language Modeling and Context Correspondence Prediction (Peng et al., 2021a). For formula search, they use a Masked Substructure Prediction task as their masking task, with masked structures representing an operator along with its parent node and child nodes in an operator tree. During training, the input to MathBERT includes formula LaTeX tokens, context, and operators:

[CLS] LaTeX [SEP] Context [SEP] (OPT Nodes)

with [CLS] and [SEP] defined as special tokens. To further incorporate structural information from the operator tree, this model modifies the attention mask matrix, leveraging the edges between nodes in the operator tree.

Rather than linearize formulas to produce a unified formula and text representation, MathAMR (Mansouri et al., 2022d) uses Abstract Meaning Representation (AMR) graphs (Banarescu et al., 2013) to produce a structured unified representation. The process to get the unified tree representation is shown in Example 2.4. First math formulae are enumerated and replaced with a special token 'EQ:ID' (where ID is an enumeration). Then, an AMR parser produces an AMR tree, after which the root of the formula OPT is inserted at the enumerated formula node. OPT edge labels are modified to be consistent with AMR conventions. A special edge label 'math' is added to the set of AMR edge labels to indicate a math formula. In the first version, despite having a unified tree representation, the AMR tree is linearized and then used to fine-tune a Sentence-BERT model. Unfortunately the linearization loses structural information, and Sentence-BERT tokenizer may not be well-suited to the AMR annotations.

4.5 Visual-spatial Models and Formula Autocompletion

Many math formulas in digital libraries are represented in PDF documents. Even for online resources such as Wikipedia, formulae are often represented by images rather than Presentation MathML or LaTeX. Formulae are also often represented as images in videos, handwriting, and slide decks. Formulas may be recognized and converted to LaTeX for use with retrieval models that we have discussed. An alternative approach is to search using visual-spatial representations of formulae that require symbols but no representation of writing lines or operation hierarchies.

Tangent-V retrieves mathematical formulas and other graphics in PDF and PNG images (Davila et al., 2019). Built on top of the Tangent-S system, Tangent-V utilizes symbol pairs extracted directly from images: for PDFs, symbols are taken directly from the file, and for PNGs, symbols are identified using an open-source OCR system (Davila et al., 2014). Line-of-sight graphs are created to capture which pairs of symbols are unblocked by other symbols. The visible symbol pairs are indexed with their relative angles in a $2\frac{1}{2}$D representation to capture symbols inside square roots and other containers. For search, candidates with shared symbol pairs are retrieved from a sparse index, and formulas with large differences in displacement angles and/or symbol size ratios relative to the query pairs are filtered.[8] A re-ranking step aligns matched pairs one-to-one with the query, and then the Tangent-S Maximum Subgraph Similarity (MSS) from Tangent-S scores candidates by the harmonic mean of query node and edge match percentages (F1).

Another visual-spatial representation is 2d histograms of symbols. XY-PHOC uses a sparse visual-spatial representation for retrieval (see Example 2.9, Avenoso, 2021). This representation generalizes a one-dimensional spatial encoding previously used for word spotting in handwritten document images, the Pyramidal Histogram of Characters (PHOC) (Sudholt and Fink, 2016). Scoring is done by cosine similarity of the PHOC embedding vectors. Formulas are represented by a bag of symbols; for each symbol, a binary vector of 29 elements is generated, where each element corresponds to a region, and 1 represents the existence of that symbol in that region.[9]

Later work found that using concentric rectangles improved PHOC-based retrieval, and that similar effectiveness is obtained using fewer region partitions, e.g., only odd-numbered partitions (i.e., 1, 3, 5, etc.) (Langsenkamp et al., 2024). As seen in other formula retrieval models,

[8]This is a Boolean query constraining symbol angles and relative sizes.

[9]PHOC may be a 2d generalization and/or variation of an unweighted binary independence term model (see Croft et al., 2009).

using PHOC for part-based rather than whole formula matching can also improve retrieval effectiveness (Tucker, 2024). The model is surprisingly effective for formula search despite its simplicity; models that incorporate IDF-like information (e.g., BM25), SPLADE-like token expansions and dense retrieval have not been properly explored with this representation yet. Also, because PHOC is domain-agnostic, requiring only a symbol vocabulary, it might provide a simple but effective unified representation for visual-spatial search of text, formulas, and other graphics.

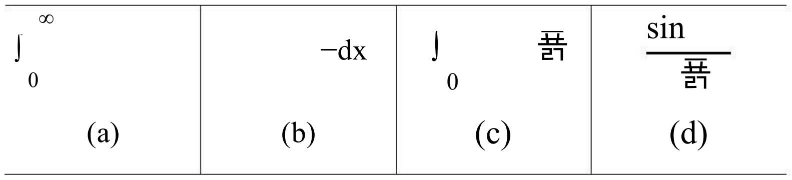
Formula autocompletion. Query auto-completion (QAC) can help users input queries more quickly, and with formulating queries when they have a specific intent but lack a clear way to express it in words. For text queries this helps prevent spelling errors, particularly on devices with small screens. It was reported in 2014 that for English queries, using QAC by selecting suggested completions saved over 50% of keystrokes for global Yahoo! searchers (Zhang et al., 2015).

Formula auto-completion is employed in search engines like WolframAlpha. This system employs prefix matching for retrieving candidates. Consequently, mathematical expressions that are reordered around commutative operators (e.g., $a + b = b + a$) or use different symbols than the query are not presented as candidate completions.

Despite extensive research in general query autocompletion, formula autocompletion remains underexplored. Rohatgi et al. (2019) proposed an approach that uses LaTeX strings and considers three methods: exact matching, prefix matching, and pattern matching. MathDeck (Diaz et al., 2021) uses TangentCFT (Mansouri et al., 2019a) to search a small collection of indexed formulae online as a user inputs a formula, displaying similar formulas. Both approaches complete the right side of a query assuming that the left side has been entered. For math formulas, entry is not always left-to-right: for example, when writing fractions or integrals.

For autocompletion, XY-PHOC has been used with conjunctive queries where all query symbols must be present in a candidate. An additional boolean constraint is also added: that returned formulas must contain no fewer symbols than the query. This is the first known model to allow symbols to be inserted in any order for formula autocompletion, because the XY-PHOC is a spatial representation rather than a tree-based one. For evaluation, formula search test collections were used. Four different symbol entry orders for XY-PHOC were compared, as illustrated in Example 4.5. Experiments confirmed the outside-in ordering shown in Example 4.5(c) constrains formula completions most quickly, raising target formulas to the top of the completion list using fewer symbols on average.

Example 4.5: Different entry orders for three symbols in $\int_0^\infty \frac{\sin(x)}{x}dx$
a) left-right, b) right-left, c) outside-in, d) middle-out.



4.6 Retrieval Effectiveness and Combining Representations

In this section we summarize the effectiveness seen to-date for the different retrieval model families presented in this section, using standard formula test collections (see Table 4.4).

Table 4.4: Formula Search Test Collections

Test Collection		Sources	Queries	Results	Metrics
NTCIR-10	(2013)	arXiv—papers	Organizers$_w$	F–in–paper	mAP, P@{5,10,hit}
NTCIR-11	(2014)	Wikipedia	Individual Wiki F (known item)	F in articles	mRR
NTCIR-12	(2016)	Wikipedia′	Organizers$_w$	F in articles	P@{5,10,15, 20}, Bpref
ARQMath-1	(2020)	2018 MSE As	2019 MSE Qs	F in MSE As	nDCG′, mAP′, P′@10
ARQMath-2	(2021)		2020 MSE Qs		
ARQMath-3	(2022)		2021 MSE Qs		
AccessMath†	(2018)	Videos + LaTeX notes	Image region or LaTeX	Image regions or LaTeX form.	R@10, mRR@10

F: Formula; As: Answers; Qs: Questions; MSE: Math Stack Exchange
$_w$Wildcard symbols in at least some query formulas
†Cross-modal or cross-language retrieval; frags.: Fragments (roughly paragraphs)

While easier to implement, text-based models are generally less effective than other representations for formula search. This is because these models do not capture the hierarchical structure of formulas. In contrast, the strongest models from the most recent ARQMath formula search tasks are tree-based models. Full-tree matching approaches, e.g., using tree-edit distance can be time-consuming and seem to be better suited for re-ranking. Also, these models are stricter than path-based models as they match full tree representations.

Results from the ARQMath-3 formula search task show tree-based approaches obtaining the strongest formula retrieval results. Approach0 obtains the highest nDCG′ of 0.72. Interestingly, Approach0 uses only OPTs for its formula representation. Tangent-CFTED obtains an nDCG′ of 0.69 using both SLT and OPT representations.[10] MathDowsers uses an SLT-based representation, and also obtains a high nDCG′ score of 0.64. There is also evidence that multi-modal representations can be helpful. For example, the multimodal image + SLT dense model EARN obtained higher Bpref scores compared to the tree-based Tangent-S model that it extends (0.69 vs. 0.64 on the NTCIR-12 Wikipedia Formula Browsing task).

Experimental results suggest that despite providing rich contextual features, dense formula retrieval models may be better suited for finding similar or partially-relevant formulae than very similar or fully relevant formulas. The embedding vectors do well at capturing shared contexts, but not necessarily specific symbols and structures in current models. Given their ability to match similar formulas well, albeit not rank them in the ideal order, many approaches use embedding-based models to select candidates, and then re-rank the results using similarity scores from tree-based models. Looking at the ARQMath-3 formula search task, Tangent-CFT's first stage dense retrieval obtains an nDCG′ of 0.64; after re-ranking with tree-edit distance this increases to 0.69. A similar pattern is seen for MathBERT on the NTCIR-12 formula browsing task: for partial matching the Bpref score is higher than that for reported tree-based models (0.74), but drops to 0.61 for full relevance matching.

[10]Tangent-CFTED is actually a two-stage model using dense embeddings of tree paths for first-stage retrieval, and tree edit distance for reranking.

For visual-spatial models, the original XY-PHOC had much lower effectiveness than other approaches at ARQMath-3 for full rank metrics (nDCG′ of 0.47). However, it was surprisingly competitive in metrics focused on the top of rankings (e.g., P′@10). Later refinements including using additional levels and rectangular regions increased nDCG′ to 0.623, and P′@10 to 60.9%. These measures were respectively 10% and 8% lower than the best performing tree-based model at ARQMath-3 (Approach0). This is interesting because PHOC models use less complex representations and a simple sparse retrieval model, and do not employ statistical weighting or machine learning (Langsenkamp et al., 2024).

Ensembling and learning-to-rank. Ensembling and learning-to-rank approaches have been used to combine the benefits of different representations, and to combine dense and sparse retrieval models. TanApp (Mansouri et al., 2019a), FORTE-App (Wang et al., 2021), and MathApp (Peng et al., 2021b) are models that use a linear combination of relevance scores from dense embeddings (Tangent-CFT, FORTE, and MathBERT, respectively) with tree-based sparse retrieval and re-ranking (Approach0). These ensemble models provide better effectiveness compared to each of the individual models. MathApp and TanApp both obtained higher Bpref values on the NTCIR-12 formula browsing task compared to their component systems.

Using learning to rank models for formula search is underexplored. An early attempt used RankBoost with formula and text features (Gao et al., 2016). A more recent study used only formula features, combining similarity features from tree-based and embedding-based models and training an SVM-Rank model (Joachims, 2006; Mansouri et al., 2021b). The input features in this approach include similarity features from tree paths, full-tree matching, and embeddings. The SVM-Rank weights allowed observing feature importance. Consistent with results seen for the best-performing tree-based retrieval models, it was found that full-tree matching features on both symbol layout and operator trees were among the important features.

5

Math-Aware Search

When we have math information needs for topics that include text, or are too complex to be addressed by formula search alone, we can use math-aware search engines supporting queries with both formulas and text. Math-aware search tasks range from the simple ad-hoc query "$a^2 + b^2 = c^2$ proof" to complete questions expressed with formulas and text. Math-aware search can be understood as a multi-modal extension of traditional text-based search models.

In this section, we first present test collections for math-aware search. We then present multi-modal math-aware search models, which either (1) retrieve text and formulas separately and combine relevance scores for individual sources, or (2) use a unified formula + text representation to retrieve sources directly. Large Language Models (LLMs) have recently been applied to math-aware search, including a state-of-the-art model that transforms the problem of retrieving question answers. A question answer in text/LaTeX is generated using an LLM, and the LLM answer is then embedded using a unified formula/text representation to search embedded answers rather than the original question. The section closes with some additional insights related to math-aware search and LLMs.

5.1 Test Collections for Math-aware Search

As with formula search, currently the primary standard test collections are NTCIR and ARQMath.[1] Test collections for math-aware search are summarized in Table 5.1.

Table 5.1: Math-aware Search Test Collections.

Test Collection	Sources	Queries	Results	Metrics
MREC (2011)	arXiv papers	—	—	—
CUMTC (2015)	arXiv papers (MREC data)	MathOverflow Qs	Papers	mAP
NTCIR-10 (2013)	arXiv papers	Organizers$_w$	Papers	mAP, P@{5,10,hit}
NTCIR-11 (2014)	arXiv frags.	Organizers$_w$	Paper frags.	mAP, P@{5,10,hit}, Bpref
NTCIR-12 (2016)	1. arXiv frags. 2. Wikipedia′	1. Organizers$_w$ 2. Organizers$_w$	1. Paper frags. 2. Wiki articles	P@{5,10,15,20}
ARQMath-1 (2020)	2018 MSE As	2019 MSE Qs	F in MSE As	nDCG′, mAP′, P′@10
ARQMath-2 (2021)		2020 MSE Qs		
ARQMath-3 (2022)		2021 MSE Qs		
Cross-Math† (2024)		ARQMath 1-3 Qs (4 languages)		P′@10, nDCG′@10

F: Formula; As: Answers; Qs: Questions; MSE: Math Stack Exchange
$_w$Wildcard symbols in at least some query formulas
†Cross-modal or cross-language retrieval; frags.: Fragments (roughly paragraphs)

MREC and CUMTC: The first math-aware search collection with annotated formulas we are aware of is the Mathematical REtrieval Collection[2] (MREC) (Líška et al., 2011). MREC consists of 439,423 scientific documents from arXiv with more than 158 million formulae with MathML annotations. Four years later, the Cambridge University MathIR Test Collection (CUMTC) (Stathopoulos and Teufel, 2015) built on MREC, adding 160 test topics derived from 120 MathOverflow discussion threads. This was one of the first attempts to use math community question-answering websites for producing real-world topics rather than topics created by shared task organizers. CUMTC topics were selected from question excerpts from 120 threads. These threads have at least one citation to the MREC collection in their accepted

[1]Additional details and comparison of these collections are available (Mansouri et al., 2021a).

[2]https://mir.fi.muni.cz/MREC/index.html.

answer.[3] The majority of topics (81%) have only one relevant document, and 17.5% have two relevant documents.

NTCIR: NTCIR-10, -11, and -12 used largely the same collections as for formula search tasks described in the previous section, consisting of arXiv papers and Wikipedia articles. Sources for NTCIR-10 were complete technical documents, which makes assessment challenging. For NTCIR-11 collection sources were reduced to excerpts (roughly paragraphs) resulting in 8,301,578 search units. NTCIR-12 uses the NTCIR-11 arXiv collection, along with a collection of (full) Wikipedia articles.

The NTCIR collections contained an increasing number of math-aware (formula+text) search topics with assessments for each lab (NTCIR-10: 15, NTCIR-11: 50, NTCIR-12: 29 (arXiv collection) + 30 (Wikipedia collection)). In NTCIR-11, topics (queries) all had at least one keyword and one formula. In NTCIR-12, topics were developed for two different collections (arXiv and Wikipedia), and all topics contained at least one formula, however 5 arXiv and 3 Wikipedia topics had no keywords. All NTCIR math-aware search topics are lab-generated, and only the query is provided with no additional description of information needs and search scenario. Pooling methods also differ between the different collections (Mansouri et al., 2021a).

The assessment process in NTCIR-10 for text+formula searches was similar to the formula search task, with the same assessors. Relevance was decided from retrieved formulas rather than documents due to their size and complexity. For each formula, assessors used a graded 0-2 scale, to represent 0: non-relevant (N), 1: partially relevant (PR) or 2: relevant (R) judgements. Each formula was assessed by one or two assessors.

In NTCIR-11, assessors were shown the title of the topic, the relevance description, and an example hit (if any) as supplementary information. For this collection, relevance was determined using roughly paragraph-sized sources rather than individual formulas. The assessors were undergraduate students in mathematics for the arXiv topics, and computer science for the Wikipedia topics. Each hit was evaluated by two students, with their judgements combined. Relevance levels were

[3] Answer accepted by the user who posted the question.

defined the same as for NTCIR-10. The final relevance score from two assessments assigned R/R and R/PR to relevant (2), PR/PR, R/N, PR/N to partially-relevant (1), and N/N to not relevant (0).

To evaluate effectiveness, NTCIR-10 uses MAP and P@{5,10, hit}. In NTCIR-11 used MAP, P@{5, 10}, and bpref to accommodate unjudged instances. NTCIR-12 reported P@{5, 10, 15, 20}. In all cases, relevance judgments for sources missing from the pools (as can happen for P@hit and MAP) were treated as not relevant.

ARQMath: ARQMath's main task is Answer Retrieval, where Math Stack Exchange (MSE) question posts containing text and formulas are used to search MSE answer posts. ARQMath's topics and collection were built as shown in Figure 5.1. All questions and their related answers posted from 2010 to 2018 are provided for training, including roughly 1 million questions and 28 million formulas. ARQMath topics were selected from new questions posted in 2019, 2020, and 2021.

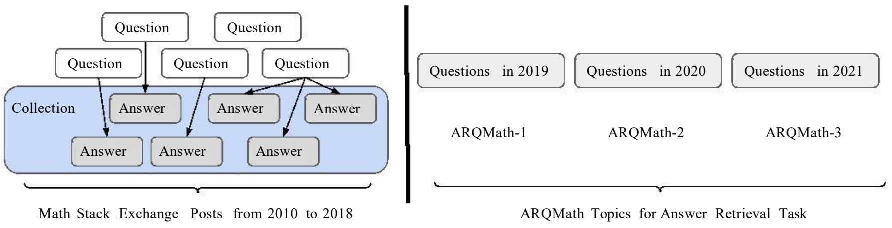


Figure 5.1: Topics and Collection for ARQMath Answer Retrieval Task. Math Stack Exchange (MSE) question answers from 2010–2018 are the collection searched, while topic questions were posted in 2019 or later. Question and answer posts from 2010–2018 are also provided for training (i.e., all posts shown at left).

The answer retrieval task was motivated by three things. First, short answer posts are easier to assess, as they usually contain at most a few paragraphs, and are organized within question threads. Second, a query log analysis showed that the number of question queries was almost 10% higher for math searches compared to searches on other topics (Mansouri et al., 2019b). Third, question posts act as both queries and information need descriptions.

There are 226 assessed test topics in total (ARQMath-1: 77, ARQMath-2: 71, ARQMath-3: 78), along with assessments for additional training topics. ARQMath also provides all system runs used for pooling.

This provides a way to study different approaches and understand their behavior in greater detail. Assessors use four relevance ratings, as defined in Table 5.2. All relevance ratings organized by topic and assessor may be found in a 'big' qrels file available with the test collection.

Table 5.2: ARQMath Answer Retrieval: Relevance Assessment Criteria

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

To be selected as a topic, a question needs to contain at least one formula. To diversify topics, 3 categorizations were assigned to candidate question posts, and a stratified sampling strategy was used to select the final topic sets. These categories are:

1. Topic type: computation, concept or proof
2. Difficulty: low, medium, and high
3. Representation dependency: text, formulas, or both

Assessors were selected from students in mathematics, similar to the ARQMath formula search task. For each edition, there were 2-3 training sessions with a math professor to introduce the task and train the assessors. 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 assessors having to guess the level of mathematical knowledge available to the person posing the question, 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. Finally, for evaluation, the same evaluation measures as the formula search task were used: nDCG$'$, MAP$'$ and P$'$@10. Prime metrics avoid issues with unevaluated hits (see Section 3).

Cross-lingual math information retrieval: Current test collections for math-aware search are primarily developed for the English language, limiting their accessibility and inclusivity. Cross-lingual math information retrieval (CLMIR) is a new task, focusing on retrieving mathematical information across languages. CLMIR has been explored for math-word problems (Tan et al., 2022), where existing datasets were translated into Chinese using online machine translators, and manually refined the translations. CrossMath (Gore et al., 2024) is a novel CLMIR test collection comprised of ARQMath Answer Retrieval task topics that have been manually translated into four languages (Croatian, Czech, Persian, and Spanish). This collection helps address a research gap in need of filling. Some approaches for machine translation of mathematical text have been proposed (Ohri and Schmah, 2021; Petersen et al., 2023), and we expect more in the future.

5.2 Searching with Formulas and Text

Unlike text or formula search where queries and sources have one representation, for math-aware search queries and sources combine text with one or more formula representations (e.g., SLT, OPT, or LaTeX tokens). Because of the different representations (i.e., modalities), this is a math-specific variation of multimodal information retrieval (Zhu et al., 2024; Shirahama and Grzegorzek, 2016).

A key challenge in multi-modal search is bridging the gap between diverse data types such as text, images, videos, and audio (Bozzon and Fraternali, 2010). There are two main approaches for searching multiple representations: searching modalities separately and then combining rank scores for individual sources (i.e., federated search), or by searching a single unified representation for all modalities. Below is a summary of approaches for combining formula and text search in sparse and dense search indexes.

Sparse retrieval:

Formulas are represented using text tokens (e.g., in LaTeX) or token sequences annotated on formulas after traversing nodes of formula trees (e.g., depth-first traversal of OPT or SLT). These formula tokens may represent individual symbols, or tuples, e.g.,

'+' node in OPT for $x + 1$ as operator-prefix tuple $(+, x, 1)$, or $(x, 1, \rightarrow\rightarrow)$ for the SLT path from x to 1.

- Federated: text and each formula representation have their own inverted index. Query text and formula representations are separated before searches are run, and relevance scores are combined to score individual sources.
- Unified: text and formula tokens belong to one vocabulary, and a 'traditional' inverted index is used to search both together (see Section 1). Linearized formula tokens are inserted in source text before indexing, and text/math tokens produced for queries are looked up in the unified inverted index.

Dense retrieval:

Formulas and/or subexpressions are annotated with vectors in embedding spaces (see Section 2). Before embedding, formula representations may be linearized tokens (e.g., LaTeX), trees (e.g., OPT, SLT) or other representations (e.g., PHOC).

- Federated: each text granularity (e.g., token vs. sentence) and formula representation have separate embeddings. Query text and formula vectors return their nearest-neighbors, and similarity scores are combined to score individual sources.
- Unified: text and formula elements for sources and queries are embedded in the same dense vector space. Multiple space may be used for different granularity (e.g., text+math in passages, vs. individual math/text token embeddings). Vector(s) for sources close to queries in the unified embedding space(s) are used to score sources.

5.3 Federated Search: Combining Formula and Text Results

With several sparse and dense formula search models available, one approach to math-aware search is combining text search with one or more separate formula searches, and then combining the results to produce scores for matches sources. Techniques for combining results include boolean constraints, linearly combining formula and text scores, learning-to-rank, and voting methods.

Figure 5.2 shows a math-aware search model that uses independent searches for text and formulas identify relevant sources (e.g., documents or passages). At indexing time, the extractor is used to separate the formulas and text of sources into two separate search indexes, and formulas are annotated with token sequences for sparse models, or embedding vectors for dense models. At query time, the same extractor splits a query into sub-queries for text and formulas, and annotates formulas with a token sequence or vector. The results from both searches are combined into the final score for retrieved sources, and the final result is communicated to the user.

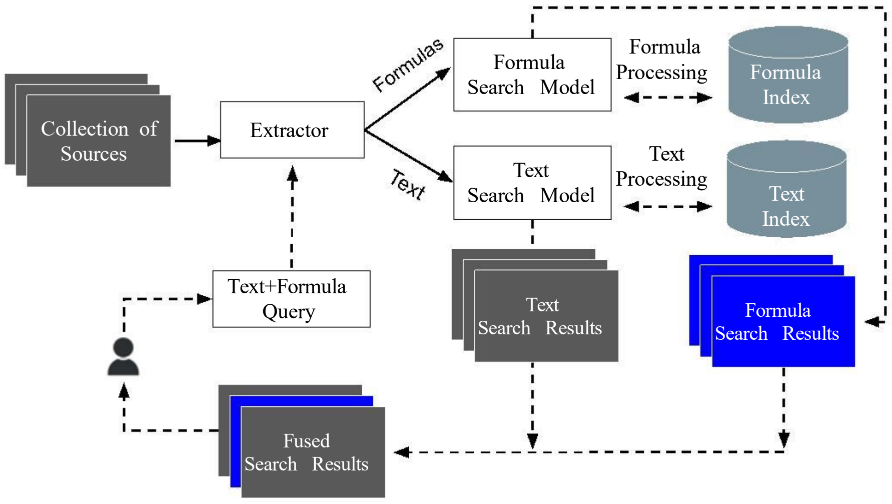


Figure 5.2: Federated Search for Formulas and Text. Formulas and text are first retrieved and scored independently. Multiple relevance scores for formulas and/or text passages from individual sources are then fused before producing a final ranking.

The first approach that we'll consider for fusing independent formula and text searches is using Boolean queries to filter sources that do not contain both formula and text matches. MathWebSearch (Hambasan et al., 2014) represents formulas as tokens for OPT subexpressions stored in a separate inverted index for formulas (implemented in ElasticSearch). Text search results are used to define Boolean queries of the form $(\text{formula}_1 \vee \ldots \vee \text{formula}_n) \wedge (\text{term}_1 \vee \ldots \vee \text{term}_n)$, requiring at least one formula and one text token from the query to match a source. Sources without text token matches are removed from the formula rank score list, which produces the final ranking. Despite the simplicity of this approach, the system achieved a P@5 of 0.79 for the NTCIR-11 math-aware search task (arXiv paper collection).

The same boolean constraint is used to filter sources that do not contain both formula and text matches in the MIaS system (Sojka and Líška, 2011), which uses canonicalized tuples as formula tokens (e.g., with variable unification; implemented in Apache Lucene). Rank scores for sub-queries generated from different combinations of text and formula tokens are used to re-rank the remaining candidates multiple ways. Results from this rankings set were interleaved in the final result to improve the diversity of returned sources (Sojka et al., 2018).

Simple averaging and linear combinations of rank scores have also been used. For the ARQMath Answer Retrieval task three baselines methods were used for answer retrieval:

1. Tangent-S formula search (linearly combined OPT and SLT scores),
2. TF-IDF text search, and
3. average of normalized ($[0, 1]$) Tangent-S and TF-IDF scores.

For Tangent-S, the largest formula (SLT) in the question's title was selected, and if no formula was used in the title, the largest formula in the question body was used. The combined model was more effective then the formula or text model in isolation.

Linearly combining formula and text rank scores was used in the MathDowsers (Ng et al., 2020) system, where Tangent-L results for formula search are combined with BM25+ text search by scaling and adding relevance scores for sources. For text search, a keyword extraction model was used to select tokens in question answer posts for use in text queries. MCAT (Kristianto et al., 2016a) also linearly combines formula and text rank scores, but for multiple formula (OPT, SLT) and text indexes. Text is indexed separately at the paragraph and document (title, abstract, keywords, ...) levels. This model was the most effective for participating teams in the NTCIR-12 math-aware search task (arXiv collection), in part due to the rich variety of formula and text representations.

The WikiMirs system (Gao et al., 2016) uses a learning-to-rank approach to combining formula and text scores. OPT-like tokens are generated from the SLT representation for a LaTeX string, using two representations with (1) concrete symbols, and (2) subexpressions replaced by wildcards ($*$). Sources are retrieved using inverted indexes for text and formula tokens. These candidates are then re-reranked using RankBoost (Freund et al., 2003) applied to features focused on formulas. This system achieved the highest P@K values among the participating teams in the NTCIR-12 math-aware search task (Wikipedia collection).

MaRec (Math answer Recommender) (Gao and Ng, 2023) uses the borda count to combine formulas and text scores for answer retrieval. A Naive Bayes classifier is used to categorize topics in answers and queries/questions (e.g., algebra, geometry, etc.). At query time, answers from topics inferred for the query question are selected, and then ranked separately by text and formula similarity. Text similarity is scored by the Kullback-Leibler (KL) divergence between question/answer token frequency distributions, using a vocabulary chosen from terms characterizing topics as detected via Latent Dirichlet Allocation (LDA, Blei et al., 2003). Formula similarity is computed from average SLT tree-edit distances, and a depth score based on the sum of leaf-root OPT path lengths. The final rank score uses the Borda Count, adding the number of answer posts that rank lower than a source in the formula and text rankings.

Most systems that combine independent formula and text searches to date use sparse retrieval (i.e., inverted indexes) to produce the initial retrieval results. However, some models combine sparse and dense retrieval models. For example, ColBERT (Khattab and Zaharia, 2020) has been used for text search, and the similarity scores linearly combined with formula search results produced using from Approach0 (described in the previous section). The MSM team at ARQMath-3 use Reciprocal Rank Fusion (RRF) to combine sparse TF-IDF and BM25 retrieval models with a RoBERTa dense retrieval model. Final rank scores for sources are computed using Equation 5.1. In the RRF equation, R is the set of formula and text rankings, and r(d) is the rank of document d. MSM achieved nearly the same nDCG$'$ as the top participating system at the ARQMath-3 answer retrieval task (0.504 vs. 0.508).

$$\mathrm{RRF}(d) = \sum_{r \in R} \frac{1}{60 + r(d)} \tag{5.1}$$

5.4 Unified Formula + Text Representations

Combining results from separate indexes for different representations can often produce useful results quickly, especially for sparse retrieval models. However, the text-notation interactions described in Section 2 provide important context that is missing when formulas and text are indexed separately. We next consider searching unified representations for formulas and text.

DLMF was the earliest unified sparse retrieval model indexed formula and text tokens together, using a variation of TF-IDF for scoring (Miller and Youssef, 2003). More recently Latent Dirichlet Allocation (LDA) has been used to weight formula and text tokens. In these sparse models, formulas are represented as LaTeX tokens (Yasunaga and Lafferty, 2019) or linearized tree tokens (Thanda et al., 2016), and a single inverted index is used to retrieve and score sources. MIaS used a similar approach (Sojka et al., 2018).

The introduction of the ARQMath answer retrieval task coincided with the emergence of transformer-based dense retrieval models. A common technique is using a pre-trained transformer (e.g., BERT variants) that is fine-tuned using pairs of MSE questions and answers with their associated assessor relevance ratings. Reusch et al. (2022) studied dense retrieval using ColBERT and ALBERT models. To fine-tune ALBERT, 1.9M triples containing questions with one relevant and one non-relevant answer were fed to the model. The model attempts to match assessor scores by classifying answers using the learned vector embedding for the [CLS] token that starts each token sequence. A similar approach was used to fine-tune ColBERT, but using more relevant and non-relevant answers. Somewhat surprisingly, both models proved less effective than sparse retrieval models that participated in the ARQMath-3 shared task.

In subsequent work, Reusch et al. (2024) explored how mathematical formulas affect a transformer model's training. They found that transformer models consider formulas when scoring relevance for answers to a given math question, but a study of the transformer attention weights for formula and text tokens suggests that structural relationships between formula tokens are lost, and that the attention maps do not capture associations between variables appearing in both questions and answers.

This finding motivates creating transformer-based models that can better capture formula structure and interactions. One approach is adding additional tokens. The MathPredictor model (Jo et al., 2021) extends BERT's tokenizers to support 2,651 new tokens. This addresses the BERT WordPiece tokenizer's oversegmention of LaTeX commands such as '\overline', which is split into three tokens: {\, over, ##line}. This allows formulas such as $\overline{h}$ (in LaTeX expressed as $\overline h$) to be correctly tokenized as {$, \overline, h , $ }. The model was then fine-tuned using masked tokens in formulas.

A hybrid approach that incorporates a unified representation with independent formula and text searches is taken in the MABOWDOR system (Zhong et al., 2023). The PyA0 toolkit is used for preprocessing mathematical formulas before tokenization by WordPiece. PyA0 canonicalizes math tokens by merging those likely to be semantically identical, such as \emptyset, \empty, and \varnothing. 1,000 new math tokens are added to the token vocabulary, and sources are then annotated with the formula math tokens, which are treated the same as regular text in dense embeddings. The final search combines dense and sparse retrieval, and uses both independent and unified formula+text representations. A unified single-vector dense retriever is used for passage-level representations of formulas and text (DPR, Karpukhin et al., 2020). To take advantage of precise formula symbol and structure matching, formulas are searched independently using Approach0 (combining path-based sparse retrieval with structural alignment). A dense embedding-augmented sparse retriever (SPLADE, Formal et al., 2021) is also used to search the text independently, and these different retrieval results are combined.

For the MABOWDOR unified representation dense passage retriever, a new pre-training dataset for the math domain was created using Coco-MAE, a retriever architecture and pretraining scheme. The pretraining task is Masked Auto-Encoding (MAE), which is similar to masked token pre-training but attempts to decode whole input passages with masked tokens using a decoder. For the final ranking, search results from each component are merged using a convex linear interpolation. This system currently archives the highest P′@10 for ARQMath's answer retrieval task.

In an alternative approach, MathBERT (Peng et al., 2021b) creates a unified formula+text representation using linearized OPT tokens and LaTeX formula representations. BERT is pre-trained using Masked Language Modeling, Context Correspondence Prediction (similar to next sentence prediction), and Masked Substructure Prediction (for masked formula tokens). Using an improved tokenization approach for math formulas, this model archives better effectiveness compared to BERT in tasks such as formula topic generation (predicting the topic (tag) associated with a mathematical formula, using a TopicMath-100K dataset created from arXiv papers), and formula 'headline' generation creating a concise description of a formula using the formulas and descriptions in a MSE question (using EXEQ-300K, Yuan et al., 2020).

Researchers then used MathBERT's language model for retrieval and other downstream task such as automatic short-answer grading (Zhang et al., 2022) using an integer scale from 0 to 4. The MathBERT model is fine-tuned for this task, with pairs of questions and student responses. Additional information including the grade scale and example graded answers are also used for in-context meta-learning.

5.5 Using LLMs for Math-aware Search

The use of LLMs to generate question answers and re-rank have been explored using ARQMath's answer retrieval task. Satpute et al. (2024) considered both general LLMs like GPT-4 and math LLMs like ToRA to generate answers for ARQMath topics. These answers were then used as queries to search for relevant answers. This can help by transforming the query into a format more similar to sources in the collection, as the query and collection are expressed as answers. For each question, an embedding (using BERT_cocomae) of the generated answer was used to search dense embeddings for ARQMath answers. Cosine similarity was used to find the most similar answer in the collection for generated answers. This approach currently obtains state-of-the-art full-ranking results for the ARQMath answer retrieval task, with nDCG' of 0.486 (vs. 0.464 of BERT_cocomae).

In another study, the applications of general LLMs, (LLaMA-2 and Orca-2) were studied for three tasks in math information retrieval: relevance assessment, data augmentation, and point-wise re-ranking (Mansouri and Maarefdoust, 2024). This study was done using ARQ-Math. For each task, an appropriate system message is used; for example, to assess the relevance of a question answer, the system prompt was created based on the ARQMath assessment protocol:

> You are a math professor who will assess the relevance of an answer to a given math question.

The results of this study revealed that while general LLMs are not yet suited for relevance assessment or re-ranking, data augmentation from the Orca-2 LLM system may be useful for expanding the ARQMath training set for use in fine-tuning neural math answer retrieval systems. The data augmentation process is performed by generating additional relevant answers for training topics using LLMs. We further discuss the use of LLMs for data augmentation in the next section.

6

Math Question Answering

Solving math problems by computer has been a goal for artificial intelligence research for a long time, beginning with work by Bobrow, Feigenbaum, Feldman, and Charniak in the 1960's (Zhang et al., 2020a). More recently, an exciting challenge is the AI Mathematical Olympiad (AIMO)[1], which is awarding a financial prize for the first publicly available AI model capable of winning a gold medal at the International Mathematical Olympiad (IMO).

In this section we focus on the rapidly developing area of math question answering. Work in this area recently spiked with the advent of transformers and Large Language Models (LLMs) trained on large volumes of text. These models may be further trained on mathematical text specifically (e.g., with formulas in LaTeX), and can convert questions in mathematical prose to formulas, program code, and answers with accompanying explanations. Further, they can be used to generate additional training data by reformulating questions and generating question variations, which may then be used to retrain and improve a model. However, there are currently limitations in effectiveness arising from the types of math representations used, and challenges with producing sound mathematical reasoning.

[1] https://aimoprize.com/

As in the previous two sections, we will begin with an overview of existing test collections for math question answering, followed by an overview of systems that have tackled the challenges in these benchmarks. A summary of the evaluation metrics used, and comparisons with formula and math-aware search may be found in Section 3.

6.1 Test Collections for Math QA

A summary of available test collections is shown in Table 6.1. Math question answering test collections require target answers to be generated or extracted for questions, in order to compute accuracy, the percentage of questions that match the target value for a question. These target values are often a numeric quantity, multiple choice alternative, or a list of values. Some collections also require open responses (i.e., free text), in which case text similarity measures are used for evaluation. Most test collections are either focused on or include math word problems, where questions are written primarily with words rather than numbers or equations (see Example 3.2).

Earlier test collections such as Dolphin18K (Huang et al., 2016) focus on arithmetic and algebraic problems, with solutions provided only in the form of equations or final answers. MathQA (Amini et al., 2019) contains 37k English multiple-choice math word problems built on AQuA-RAT (Ling et al., 2017), with efforts aimed at addressing issues in AQua-RAT including incorrect solutions and problems that required brute-force enumeration. Despite the corrections to the AQuA-RAT dataset, around 30% of MathQA solutions had inconsistencies.

ASDiv (Academia Sinica Diverse MWP Dataset) (Miao et al., 2020) contains 2.3K math word problems, with each question labeled with a problem type (e.g., Basic arithmetic operations, Aggregative operations) and grade level to show the difficulty level of the problems. This dataset provided different diversity metrics which helped the development of future datasets. This includes GSM8K (Cobbe et al., 2021), which contains 8.5K school math problems created by human problem writers, with 1K problems as the test set. These problems need 2 to 8 steps to get to the solution, using primary calculations with basic math operations to find the answer. This dataset contains questions that need basic knowledge, such as the number of days in a week.

Table 6.1: Math Question Answering (QA) Test Collections. QA systems communicate one answer which may be a formula, computer program, value (e.g., numbers, lists), multiple choice alternative, and/or written responses. Many questions come from standardized tests and math competitions.

Test Collection		Questions	Answers	Metrics
Math Problems				
Dolphin18K	(2016)	Arithm./algebra	Number set	Accuracy◦
AQuA-RAT	(2017)	MC△ WP†	Rationale + chosen answer	Accuracy, Perplexity, Expert ration. + BLEU
SemEval	(2019)	SAT questions	Numeric or chosen answer	Accuracy
MathQA	(2019)	MC WP	Rationale + chosen answer	Accuracy
ASDiv	(2020)	WP	Equation and value	Accuracy
MATH	(2021)	AMC, AIME	Step-by-step soln + final answer	Accuracy
GSM8k	(2021)	WP	Step-by-step soln + final answer	Accuracy
DROP	(2019)	Paragraph questions	Number, entity, etc.	Accuracy, token F1,EM
LILA	(2022)	WP	Numbers or formulas	token F1
AIMO	(2024)	Int'l. Math. Olympiad (IMO) questions	Integers in [0,999]	Accuracy
Math Problems with Graphical/Visual Content				
GeoQA	(2021)	MC geometry problems w. diagram	Chosen answer	Accuracy
UniGeo	(2022)	MC w. diagram (GeoQA extension)	Chosen answer	Accuracy, Accuracy@10
MathVista	(2023)	Figure QA, WP, geometry problems, textbook questions, VQA	Chosen answer or numeric	Accuracy
Math-Vision	(2024)	Math reasoning with visual elements	Step-by-step soln + final answer	Accuracy
TabMWP	(2023)	MC WP w. tabular data	Number or text	Accuracy
Open Response				
ARQMath-3	(2022)	ARQMath-3 MSE Qs	Written response (formulas + text)	Relevant MSE answers, + token F1, + BERTScore
MathDial	(2023)	LLM-simulated student Q. or comment (GSM8k questions)	Responses to Q. and comments	Expert resp. + sBLEU, + BERTScore

†Word Problems; ◦Accuracy = test@1 = % correct ≈ Exact Match (EM)
△Multiple Choice

SemEval 2019 Task 10 (Hopkins et al., 2019) provides question sets from MathSAT (Scholastic Achievement Test) practice exams in three categories: Closed Algebra, Open Algebra, and Geometry. Most questions are multiple choice, with some numeric answers. This test collection contains 3860 questions, of which 1092 are used as the test set. The MATH dataset (Hendrycks et al., 2021) also contains multiple choice questions, developed from American high school math competitions including AMC 10/12[2], and AIME[3]. This dataset provides 12,500 problems (7,500 training and 5,000 test). Problems are assigned difficulty levels from 1 to 5, and categorized into seven areas (Pre-algebra, Algebra, Number Theory, Counting and Probability, Geometry, Intermediate Algebra, and Precalculus). Answers in this dataset include step-by-step solutions in LaTeX with final answers annotated explicitly, making it suitable for training models that provide explanations for their answers.

Other test collections aim to test mathematical reasoning by systems. The DROP (Discrete Reasoning Over the content of Paragraphs) test collection (Dua et al., 2019) is a reading comprehension task where a paragraph is provided along with questions about the passage. DROP has often been used to evaluate the reasoning capabilities of large language models such as Gemini and PaLM.

After DROP, LILA (Mishra et al., 2022) was developed to unify various mathematical reasoning benchmarks. LILA augments 20 existing datasets with solution programs added to answers, along with instructions for producing answers in natural language. Answers are represented as Python strings that print a number, expression, list, or other data structure. For each task, instruction annotations are provided with a clear definition of the task, a prompt providing a short instruction, and examples to help learning by demonstration. System effectiveness is measured using the token F1-score between the model output and the target answer.

Questions with graphics. Plots, diagrams, and geometric concepts are commonly used in math. While text-based problems have been investigated extensively, visual math question-answering has been explored far less. In recent years there have been attempts to automatically answer questions that include graphics such as tables and diagrams. One of the earliest such MathQA tasks focused on retrieving

[2] https://maa.org/amc-10-12-information-and-registration/

[3] https://artofproblemsolving.com/wiki/index.php/American_Invitational_Mathematics_Examination

formulas from Wikidata in response to questions in natural language (Schubotz et al., 2018). An example test collection is TabMWP (Lu et al., 2023), which contains 38K open-domain grade-level problems that require mathematical reasoning on both textual and tabular data. To solve these problems, systems need to consult data in given table cells.

GeoQA (Chen et al., 2021a) is a geometric question-answering dataset. Each question provides a textual description of the problem and its related diagram, and systems choose from answers given as multiple choice alternatives. UniGeo (Chen et al., 2022a), expanded this dataset, by including 4,998 calculation problems (from GeoQA) and adding 9,543 proof problems.

MathVista (Lu et al., 2024) provides 6,141 examples of mathematical and visual tasks and studies several large language models, including multimodal models.[4] These questions focus on five tasks: figure question answering, geometry problems, math word problems, textbook questions, and visual question answering through multiple-choice or free-form questions. Math-Vision (Wang et al., 2024) is another recent test collection for studying large multimodal models for math questions with visual content, providing 3,040 diverse problems.

Open response questions and dialogues. ARQMath-3 (Mansouri et al., 2022a) had a pilot open domain question-answering task, which used the same topics as the math-aware search task. Evaluation measures were computed from lexical overlap of tokens, where answers are treated as a bag of tokens, using the maximum F_1 score between system answers and each relevant answers from the answer retrieval task. Using a similar approach BERTScore (Zhang et al., 2020b) is also used to measure token overlap.

There has also been early work on creating test collections for conversational math dialogue systems.[5] For example, MathDial (Macina et al., 2023) provides math tutoring dialogues produced by connecting an expert annotator, who role-plays a teacher, with an LLM that simulates the student working through problems, including reasoning errors.

[4]https://mathvista.github.io/.

[5]The use of clarifying questions in Math Stack Exchange has been studied using comments on math questions (Mansouri and Jahedibashiz, 2023).

The dataset has 3,000 tutoring conversations grounded in math word problems from GSM8k. The dataset aims to capture nuances of student-teacher interactions, with a focus on multistep math problem-solving scenarios. The main task in MathDial is Tutor Response Generation, aiming at modeling the teacher in a dialogue by generating follow-up turns to guide the student to solve problems. An earlier example of a related system is MathBot (Grossman et al., 2019), a text-based tutor capable of explaining math concepts providing practice questions and offering feedback to students. Tasks such as generating clarifying questions, and detecting ambiguous questions can be studied using these test collections and systems.

6.2 Solving Math Word Problems

In this section we focus on systems that solve math word problems, which have been the most commonly studied math question type.

Symbolic and logical rule-based approaches. Traditional approaches to solving math-word problems are rule-based. The word problem is first converted to a symbolic representation (e.g., first-order logic) which is solved using an inference algorithm. The STUDENT system (Bobrow, 1964) solved algebraic problems in Lisp using string transformations to generate symbolic problem instances.

This basic approach remains effective to this day. For example, the AiFu model (Liu et al., 2019) converts word problems into a logical representation using rule-based templates. This representation is based on assertional logic where mathematical objects are formalized as constants, variables, concepts, functions, or relations. For example, “the integer x equals to 3”, is transformed into

Integer(x), Equal(x,3).

This representation is then translated via rules to a math representation for the the Satisfiability Modulo Theories (SMT) solver Z3[6] to produce a solution. The rule-based translation uses two templates for unification: one for known concepts and operators, and the other for new entities.

[6]https://github.com/Z3Prover/z3.

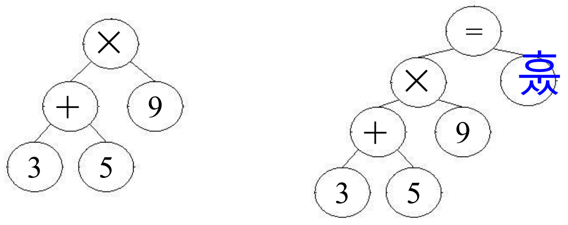
This model had the highest accuracy among participating teams in the SemEval 2019 task.

Generating and computing solutions from operator trees (OPT) is also a common approach for solving math word problems, as illustrated in Example 6.1. These trees represent the solution quantity, and a variable to which this solution is assigned for the math-word problem:

> There are 3 boys and 5 girls in a group. Each person wants to buy 9 pencils. How many pencils do they need to buy altogether?

As shown in Example 6.1, the question is transformed into an OPT with constants defined in the question at leaves in the tree. Answers are produced by applying operations bottom-up (i.e., evaluating the expression). In one approach the generated OPTs use four basic binary operations, with quantities at the leaves (Roy and Roth, 2015). The ALGES system (Koncel-Kedziorski et al., 2015) uses a similar approach, where the OPT includes a variable for the answer χ in the tree attached to an equivalence operator (referred to as an equation tree).

Example 6.1: OPT for $(3 + 5) \times 9$ generated from a word problem, and variation with answer variable χ (equation tree).

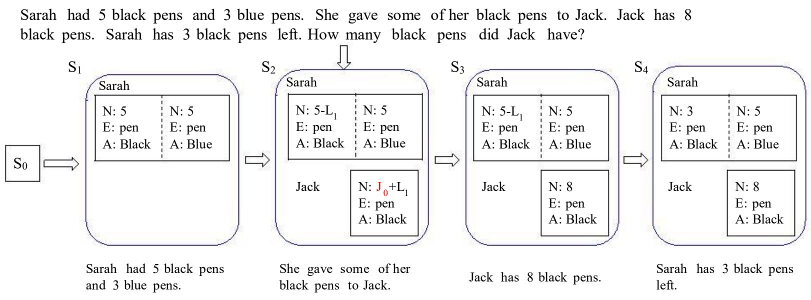


(a) Operator Tree (b) Equation Tree

For questions involving properties of elements in sets, it is important to correctly identify variable quantifiers to distinguish universal and existential quantification (e.g., $\forall x$, $\exists x$). Roy and Roth train a binary SVM classifier to select quantifiers as needed, and inserts them at leaves of the tree. The lowest common ancestor (LCA) node between pairs of quantifiers are used to capture constraints. The ALGES system instead generates all possible equation trees without dropping irrelevant quantifiers, and then uses Integer Linear Programming to score the likelihood of each tree using local and global discriminative models.

Early machine learning approaches. Several statistical machine-learning approaches have been investigated for math-word problems. Like rule-based approaches, they first produce a word problem representation, from which the answer is then generated algorithmically. The ARIS system (Hosseini et al., 2014) was developed for arithmetic problems using addition and subtraction. Steps toward the solution are represented as a state sequence, with one state per sentence. States contain subjects and objects from the question, each of which have associated entity triples. Entity triples have quantities (i.e., constants or variables), types, and attributes.

Example 6.2: ARIS sentence state generation (Hosseini et al., 2014). Each state has one or two containers in black boxes.


Sarah had 5 black pens and 3 blue pens. She gave some of her black pens to Jack. Jack has 8 black pens. Sarah has 3 black pens left. How many black pens did Jack have?
S_0
S_1
Sarah
N: 5 E: pen A: Black
N: 5 E: pen A: Blue
Sarah had 5 black pens and 3 blue pens.
S_2
Sarah
N: 5-L_1 E: pen A: Black
N: 5 E: pen A: Blue
Jack
N: J_0+L_1 E: pen A: Black
She gave some of her black pens to Jack.
S_3
Sarah
N: 5-L_1 E: pen A: Black
N: 5 E: pen A: Blue
Jack
N: 8 E: pen A: Black
Jack has 8 black pens.
S_4
Sarah
N: 3 E: pen A: Black
N: 5 E: pen A: Blue
Jack
N: 8 E: pen A: Black
Sarah has 3 black pens left.

As an example, consider this math word problem, with two subjects possessing quantity containers (Sarah and Jack):

> Sarah had 5 black pens and 3 blue pens. She gave some of her black pens to Jack. Jack has 8 black pens. Sarah has 3 black pens left. How many black pens did Jack have?

The ARIS state representation for each sentence is shown in Example 6.2, which is produced using the Stanford NLP library. J_0 shown for the second sentence represents the question quantity, the number of black pens Jack had initially. Determining this value depends also on variable

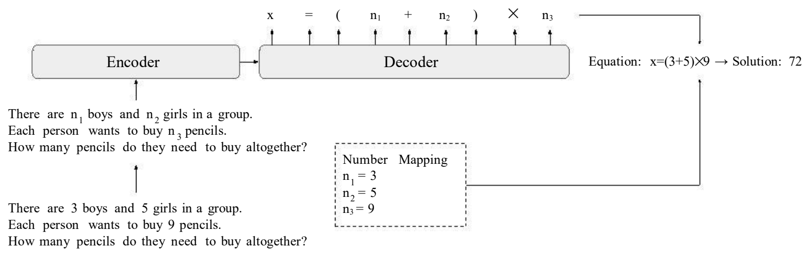
L_1, the number of pens that Sarah gave to Jack. After the sentence state representations are generated, constraints in the question are used to compute the question answer. Here we solve for J_0 by noting that Jill's final 'Black' pen entity count is $L_1 = 5 - 3$, and Jack's 'Black' pen entity is $J_0 = 8 - L_1$, giving 6 black pens that Jack originally had.

State transitions are identified by verbs classified by an SVM classifier using WordNet-based and other features. Verbs are categorized into three groups: (1) observation, (2) positive (increase), and (3) negative (decrease). For sentences with two containers (i.e., subjects/objects with entities), four other categories are considered: (4) positive transfer (from second to first container), (5) negative transfer (from first to second container), (6) construct (increase in both containers), and (7) destroy (decrease in both containers).

Recently deep learning techniques are among the strongest approaches for math-word problems. The earliest attempt, Deep Neural Solver (DNS) (Wang et al., 2017b) used a seq2seq model to translate problem statements into equations (i.e., infix OPT representations) by embedding question text in a vector and then generating a formula expression starting from the question vector. The generated expression is evaluated to produce an answer. Numbers in the question are mapped to enumerated variable tokens and stored in a dictionary (e.g., $\{(n_1, 3), (n_2, 5), (n_3 9)\}$). Numbers are replaced by their enumeration variable in the question text before being passed to a sequence-to-sequence (seq2seq) model that generates an equation over those variables. The variables of the equation are then replaced by their values from the dictionary, and the expression value is computed. Using the 'pencils' example above used to illustrate rule-based approaches, the architecture of the model is shown in Example 6.3.

After DNS was introduced there were several attempts to improve generated OPTs through better structural constraints. The multi-encoders and decoders model (Shen and Jin, 2020) uses both seq2seq and graph-based encoder/decoders. Two graph encoders are defined using GraphSage (Hamilton et al., 2017). The first is a dependency parse tree capturing relationships between words in the sentence. Initial token embeddings used for the tree come from the sequence-based encoder. A second graph-encoder captures numerical comparisons: nodes represent

Example 6.3: Seq2seq OPT generation for word problems (Wang et al., 2017b).


x = (n_1 + n_2) × n_3
Encoder
Decoder
Equation: x=(3+5)×9 → Solution: 72
There are n_1 boys and n_2 girls in a group.
Each person wants to buy n_3 pencils.
How many pencils do they need to buy altogether?
There are 3 boys and 5 girls in a group.
Each person wants to buy 9 pencils.
How many pencils do they need to buy altogether?
Number Mapping
$n_1 = 3$
$n_2 = 5$
$n_3 = 9$

numbers from the question with relations $>$ or $\leq$ defining constraints that were ignored by DNS when mapping values to tokens. Sequence and tree-based decoders are used. The decoders produce operation and value sequences that correspond to OPT traversals that can be used to compute the final answer using a simple stack-based algorithm.

For the equation $(n_1 + n_2) \times n_3$, the sequence decoder generates an OPT preorder traversal of the expression in Example 6.1, with arguments before operations $(n_1 n_2 + n_3 \times)$. The tree decoder instead generates an OPT postorder traversal with operations before arguments $(\times + n_1 n_2 n_3)$. The operation sequence used is selected by maximum decoder likelihood. This model achieves higher accuracy than DNS on the Math23K dataset (5-fold cross-validation of 76.9% vs. 58.1%).

Transformers. Solving math word problems automatically has gained increasing attention since the advent of transformers in 2017 (Vaswani et al., 2017). Transformers integrate surrounding context in token sequences by consulting all other tokens embeddings in the input over a series of stages. This produces contextually-enriched embedding vectors that dramatically improved their usefulness in prediction and generation tasks.

Thinking back to information tasks in Section 1, with a transformer the input tokens act as an initial information source, from which we produce a new information source containing contextualized token vectors. The contextualized vectors are shaped by token co-occurrence statistics seen in a large collection of sources used for training. In terms of information tasks performed by the transformer:

Retrieve

Consult: all current token vectors are examined and weighted using a self-attention mechanism used in generating new vectors.

Query: input tokens are annotated with control tokens (e.g., [CLS], [SEP], [MASK]) to (1) drive parameter learning, and (2) allow users to 'program' outputs in query prompts.

Analyze

Annotate (Designers/Users): input annotated with additional tokens, e.g., [CLS] for the first 'classification' token, [SEP] for ending sentences/sections, and signaling answer start (e.g., [A.]). Positional encoding vectors mark input position (e.g., integer enumeration) and/or token relationships. These annotations are more detailed and expressive than for earlier models.

Index: network weights represent language statistics in a collection, and can be used to produce dense vector indexes.

Synthesize

Apply: token embeddings are generated in steps, applying network weights and attention to update token vectors. This uses 'depth' to improve generalization (a known property of deep nets). Training tasks apply network weights to make decisions (e.g., guess masked tokens) and then update weights using backpropagation.

Communicate: contextualized token vectors are communicated in a tensor, providing a new information source. Learned network weights specifying the embedding function represented in the network is another communicated information source.

As described in Section 2, 'pre-training' transformers on large text corpora is the norm, followed by 'fine-tuning' to produce outputs for specific tasks. 'Pre-training' (i.e., initial language model learning) generally involves imitative games that require predictions after token sequence manipulations such as token masking and re-ordering, while fine-tuning requires replacing the output layers of the network for the new imitative game(s) of the specific task the network will be used for (e.g., classification or generation tasks).

As for retrieval, we again find that token representations used for math with transformers are important. Simple tasks such as generating answers to questions adding and subtracting two numbers were explored early on (e.g., 'what is 52 plus 148?'), using a T5 seq2seq model (Nogueira et al., 2021). This was motivated by earlier work where for tasks such as reporting the maximum value of a list of numbers, a BERT model obtained an accuracy of 52% (Wallace et al., 2019). This poor performance was caused partly the default tokenization by WordPiece, which prevented correctly encoding numbers.

The first applications of BERT model to math word problems was performed using the AQuaA-RAT collection (Ling et al., 2017). The focus of this work is fine-tuning, including a proposed Neighbor Reasoning Order Prediction (NROP) coherence loss (Piękos et al., 2021). This considers whether steps in the rationale for an answer are in their original order, or have been swapped. Fine-tuning with this task and loss function improved accuracy by roughly 10%.

Subsequent work focused on improving mathematical reasoning output for transformers. MWP-BERT (Liang et al., 2022) used BERT and RoBERTa along with three families of fine-tuning tasks/objectives.

Self-supervised (questions): (1) masked language modeling, (2) number counting (i.e., quantities in a word-problem), and (3) number type grounding (e.g., integer, real).

Weakly-supervised (questions with answers): (1) answer value type prediction (i.e., discrete or continuous), (2) context-answer type comparison (i.e., whether question quantities have the same type as the answer), and (3) number magnitude comparison (i.e., predict the relative answer size vs. question quantities).

Fully-supervised: (1) operation prediction, and (2) tree distance prediction. Operation prediction infers the operator between two quantity nodes in the solution OPT from five types: {+, -, ×, /, ^}. Tree distance prediction estimates differences in depth for numbers in the solution tree.

These different training objective families in isolation produce similar accuracy on Math23K dataset, with the fully-supervised objectives providing the highest accuracy. MWP-BERT is fine-tuned on MathQA. For Math23K it achieves 82.4% accuracy vs. 58.1% for the DNS model (5-fold cross-validation). Similarly, for the Math23K test set MWP-BERT achieves 96.2% accuracy vs. 53.6% for the DNS model.

6.3 LLMs and Mathematical Reasoning

The next generation of math-word problem solving models use Large Language Models (LLMs) which are (roughly) very large transformer models trained on very large amounts of data, and embedded within a text generation system (e.g., a recurrent neural network). The capability of large language models to solve advanced math questions is closely studied when a new LLM is developed, and math-word problems are commonly used for this purpose. General LLMs are not trained specifically for math questions; open-source models such as Mistral (Jiang et al., 2023) and LLaMA (family) have been used to fine-tune math-specific language models, which helps improve the responsiveness of models to math-focused system prompts.

From prompts the model can be given context and instructed on producing specific outputs, providing a way to indirectly 'program' outputs as described for transformers in the previous section.[7]

Improving LLM math question answers. A common, non-automated way to improve answers without retraining is prompt engineering, where an LLM is prompted multiple times, and then a preferred answer is selected. Aside from this, there are four common methods for improving answers to math questions given to LLMs vs. directly providing a word problem in a prompt:

Instruction tuning:
a labeled dataset of (prompt, response) pairs is used for additional training of the model, i.e., fine-tuning. Often done with state-of-

[7]For users, prompt=query: these are requests for generated information sources.

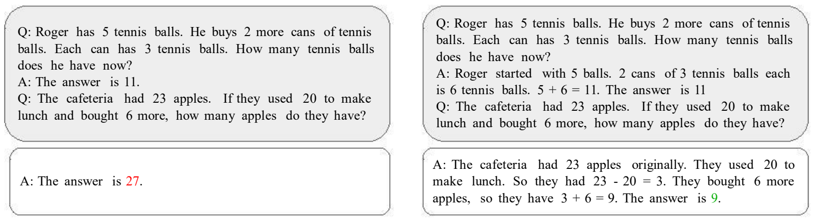
the-art language models such as GPT-4 to generate high-quality training data.

Chain-of-Thought (CoT) prompting (Wei et al., 2022):
the prompts include example questions and answers that include a step-by-step reasoning process. The goal is to exploit the LLM tendencies to mimic patterns in the prompt, in anticipation of a similar reasoning step pattern appearing in the answer, and help with producing correct answers. COT is distinct from few-shot prompting (see Example 6.4), where only questions and final answers are provided within examples included in the prompt. CoT can also be used with zero-shot or few-shot prompting, by simply adding "Let’s think step-by-step" at the beginning of the prompt.

Program-of-Thought (PoT) (Chen et al., 2022b):
similar to CoT. Instead of natural language, computer programs represent the solution process formally. PoT frees LLMs from having to generate equations in natural language, and to instead provide explicit computational steps. In a zero-shot setting (i.e., giving no example question and answer in the prompt), PoT has produced a 12% improvement over CoT for math word problems.

Program-Aided Language models (PAL) (Gao et al., 2023):
interleaves natural language (as in CoT) and programming language statements (per PoT) in answers. The final solution is a program that is evaluated by an interpreter.

Example 6.4: Standard prompting (left) vs. Chain-of-Thought (right). Input is in grey boxes. Adapted from Wei et al. (2022).



Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.
Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

A: The answer is 27.

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11
Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9.

Math-specific LLMs. Most research on math-specific LLMs focuses upon input token representations and prompts, and expanding datasets for fine-tuning general models. One of the earliest math-specific LLMs is Minerva (Lewkowycz et al., 2022). This LLM is based on the PaLM (Pathways Language Model) model, and is fine-tuned on 118GB of scientific and mathematical text. Minerva can correctly generate $\LaTeX$ formulas, which is challenging for general LLMs. With proper tokenization, formulas such as $E = mc^2$ are processed as $E=mc^2$, and not the single token Emc2.

To improve answers, Minerva uses CoT prompting with several step-by-step solutions to questions before posing the question to be answered. It also generates multiple solutions, and then uses majority voting to select the most likely final answer. From manually analyzed samples, the most frequent issues in responses include incorrect reasoning, incorrect calculation, and question misunderstanding. For example, for the question:

$$\text{If } \sqrt{400} = \sqrt{81} + \sqrt{n}, \text{ then what is the value of } n?$$

Minerva infers that $400 = 81 + n$, and then correctly provides 319 as the final answer, but for the incorrectly inferred formula.

LLEMMA (Azerbayev et al., 2024) is another model using Chain-of-Thought. The model is built atop LLaMA-Code, and fine-tuned on another dataset Proof-Pile-2 containing 55 billion tokens from mathematical and scientific documents. The documents are taken from three large collections: arXiv, open-web-math, and algebraic-stack. After fine-tuning, LLEMA is more effective at answering questions from GSM8K and MATH than Minerva.

MAmmoTH (Yue et al., 2023) uses hybrid CoT and PoT to build a new dataset. This model is a fine-tuned version of LLaMA-2 and Code LLaMA, using a new dataset MathInstruct for instruction-tuning. MathInstruct is compiled from 13 math rationale datasets (7 existing), using both chain-of-thought (CoT) and program-of-thought (PoT) rationales. To create this dataset, GPT-4 is used to generate programs, and the values generated by programs are verified with ground truth. The dataset contains 260K (instruction, response) pairs with 72% using COT, and 28% using PoT. Example 6.5 shows CoT and PoT pairs. The authors report that PoT generally provides better results than CoT. Particularly for open-form questions, algorithmic reasoning was found to be more effective for complex math problems.

Example 6.5: Sample instructions and responses from the MathInstruct dataset. Grey boxes show PoT and CoT instructions, and white boxes show responses.

PoT

If circles x and y have the same area and circle x has a circumference of 12 π, half of the radius of circle y is : ? Let's write a Python program to solve it.

```
n0 = 12.0
t0 = n0 / 2.0
answer = t0 / 2.0
print(answer)
```

CoT

There are 3 boxes of cereal. One box holds 14 ounces of cereal. Another box holds half the amount of the first box and 5 ounces less than the third box. How much cereal is there in all 3 cereal boxes?

First = 14 oz Second = (1/2) * 14 = 7 oz Third = 7 + 5 = 12 oz 14 + 7 + 12 = 33 oz There are 33 ounces of cereal in those 3 boxes. The answer is 33

ToRA (Tool-integrated Reasoning Agents) (Gou et al., 2024) introduces a new reasoning representation interleaving natural language and the use of external libraries. Given a question, ToRA first generates reasoning in natural language, which continues until a program library is better suited (e.g., for equation solving). ToRA then generates a program from the natural language reasoning, and the output is further processed for adjustments, sub-task solving, and answer finalization. This process is continued until the final answer is represented using the LaTeX "\boxed{}" command, as illustrated in Example 6.6.[8] To fine-tune LLaMA-2, the TORA-CORPUS was produced using GPT-4, synthesizing reasoning trajectories for the GSM8k and MATH training sets.

Improving LLMs through fine-tuning on augmented data. A common technique for expanding and/or improving existing datasets involves transforming existing question/answer pairs to produce new ones (i.e., data augmentation).

For example, MetaMathQA (Yu et al., 2024) augments two earlier datasets, GSM8K and MATH. Three techniques are used for data aug-

[8]Note the similarity to the types of answers seen in CQA forums like Math Stack Exchange, where code is part of a larger answer narrative.

Example 6.6: ToRA reasoning interleaving natural language and program-based tool use. Adapted from Gou et al. (2024).

Problem

Suppose that the sum of the squares of two complex numbers 픎and 픔is 7 and the sum of their cubes is 10. List all possible values for 픎+ 픔, separated by commas.

Reasoning

Let's use `Sympy` to calculate and print all possible values for 픎+ 픔,

Program

```
from sympy import symbols, Eq, solve, simplify

def possible_values():
    x, y = symbols('x y')
    eq1 = Eq(x**2 + y**2, 7)
    eq2 = Eq(x**3 + y**3, 10)
    solutions = solve((eq1,eq2), (x, y))
    sums = [simplify(sol[0] + sol[1]) for sol in solutions]
    return sums
print(possible_values())
```

Output

```
>>> [-5, -5, 1, 1, 4, 4]
```

Reasoning

Removing duplicates, the possible values for 픎+ 픔are \boxed{-5, 1, 4}.

mentation. First, additional reasoning chains are generated for answers. This is done using few-shot CoT promoting, where questions are appended to QA pairs that include associated reasoning for answers. This few-shot prompt is then fed to an LLM to produce an additional answer with justification. The final answer is compared against the ground-truth, and incorrect answers are filtered out. The second augmentation rephrases questions. Examples of rephrased questions are used in a few-shot query along with the prompt: You are an AI assistant to help me rephrase questions. Fol low the given examples. Answers generated for rephrased questions had an accuracy of 76.30%, and 80.74% for the original questions.

The third augmentation uses backward reasoning in the MetaMath LLM to generate additional questions. Numeric values in questions are masked with an 'X', and the LLM is then asked to predict the value given the answer. The backward reasoning starts from the answer, and then generates reasoning steps to infer the masked numeric value in the question. The final augmented data set is used to fine-tune LLaMA-2.

While most LLMs achieve their highest effectiveness with a large number of parameters, Orca-Math (Mitra et al., 2024) instead fine-tunes smaller language models. Similar to MetaMath, a new dataset is introduced and then fine-tuning is done over three iterations. The Orca-Math-200K dataset is constructed from previous datasets such MathQA, and GSM8K with a total of 36,217 problems. This dataset was augmented by modifying the original word problems. GPT-4-Turbo is used with few-shot examples to generate new math word problems, and then prompted to convert the question into a statement using the answer to the question. Then, from the statement, it creates a new word problem. Solutions for the new problems are also generated by GPT4-Turbo. A total of 120,445 new problems were generated.

Another augmentation increases the difficulty level of existing questions, using two agents called the Suggester and Editor. The Suggester proposes increasing difficulty using techniques such as adding more variables or increasing certain values. The Editor modifies the original question based on the suggester's recommendation. This is done in two rounds, each time increasing the difficulty further. New problems that GPT4-Turbo answers in more than 1800 characters were dropped. 37,157 new problems were collected from this step.

After the data augmentation, Orca-Math's Mistral-7B LLM is fine-tuned in three steps. First, instruction-fine-tuning is applied using pairs of questions and answers. In the second step, a teacher-student learning model is used. For each question four answers are generated (as 'student' answers) by a supervised fine-tuned model. Student answers are compared to answers from GPT-4 Turbo (the teacher) using exact-match prompting. This is used to create positive and negative examples based on agreement with the 'teacher' answer for further fine-tuning. In the third step, training samples are generated using two approaches: Direct Preference Optimization (DPO) (Rafailov et al., 2023) and Kahneman-Tversky Optimization (KTO) (Ethayarajh et al., 2024). Both approaches are used for fine-tuning LLMs to align with preferences. With the fine-tuned model, Orca-Math archives an accuracy of 86.81%, which is higher than much larger models such as MAmmoTH and MetaMath.

LLM limitations. We will use a question from ARQMath-2 to illustrate some limitations of current LLM models. Prompting Llama-3-8B-Instruct with: “You are a math professor who will answer math questions, providing step-by-step solutions.” and the question “can prove it converges but I don’t know at what value it converges. $\sum_{x=1}^{\infty} \frac{(x-1)}{2^x}$”, leads to the answer shown in Example 6.7.

Example 6.7: Llama-3-8B-Instruct generated answer for a question from the ARQMath-2 Answer Retrieval task (see main text).

Let’s analyze the given series: $\sum_{x=1}^{\infty} \frac{(x-1)}{2^x}$
First, let’s rewrite the series as: $\sum_{x=1}^{\infty} \frac{x}{2^x} - \sum_{x=1}^{\infty} \frac{1}{2^x}$
Now, let’s focus on the first term: $\sum_{x=1}^{\infty} \frac{x}{2^x}$
This is known as the geometric series with a linear term:

$$\sum_{x=1}^{\infty} rx = \frac{r}{(1-r)^2}$$

In our case, $r = \frac{1}{2}$, so: $\sum_{x=1}^{\infty} \frac{x}{2^x} = \frac{\frac{1}{2}}{(1-\frac{1}{2})^2} = 2$

Now, let’s focus on the second term: $\sum_{x=1}^{\infty} \frac{1}{2^x}$

This is a geometric series with a common ratio of $\frac{1}{2}$: $\sum_{x=1}^{\infty} \frac{1}{2^x} = \frac{\frac{1}{2}}{1-\frac{1}{2}} = 1$

So, the original series becomes: $2 - 1 = \boxed{1}$

Therefore, the series converges to 1.

This answer is related to the question, but there is an error in the fourth expression, where rx should be xr_x. Moreover, for a “math professor”, splitting of the series as:

$$\sum_{x=1}^{\infty} \frac{(x-1)}{2^x} = \sum_{x=1}^{\infty} \frac{x}{2_x} - \sum_{x=1}^{\infty} \frac{1}{2_x}.$$

is not obvious, and needs verification/reasoning about convergence for each of those series. So, if we say it is a “step-by-step” solution, then the professor has ignored this crucial step.

The system prompt can guide the responses for LLMs. For the same ARQMath question, we change LLaMA-3 prompt to “You are a math professor who will answer math questions, providing a hint toward the solutions, not the whole solution.” The response is shown in Example 6.8.

Example 6.8: A different Llama-3-8B-Instruct answer when asked to provided a hint, but not the whole solution.

Here's a hint to help you get started:

You can try to recognize this series as a variant of a more well-known series. In particular, consider the geometric series $\sum_{x=1}^{\infty} r^x$, where $|r| < 1$. Can you see how the given series can be rewritten in a similar form?

Also, think about the fact that the series is a telescoping series, meaning that many of the terms cancel out when you sum them up. Can you identify which terms cancel out and how this might help you find the sum?

Take it from here and see if you can prove that the series converges and find its value!

The first part of the hint is a good start for the student: it narrows down the scope where the student can focus or get some ideas. However, the second part of the hint is completely wrong; this is not a telescoping series. A telescoping series is one where each term of the series can be written as a difference of two consecutive terms of a sequence. In the sense,

$$\sum_{n=1}^{\infty}(a_{n+1} - a_n) = (a_2 - a_1) - (a_3 - a_2) + \cdots .$$

So, the terms start canceling off and the sum is easy to find. But the series given in the query cannot be written in this form. So, this second part of the hint does not help, and is very misleading.

Currently, Chain-of-thought and Program-of-thought are commonly used to produce more reliable answers to math questions by LLMs. It is likely that more effective techniques will be used in the future. Current models need to individually create a dataset for fine-tuning their base LLM model (e.g., LLaMA-2), often created using a state-of-the-art LLM (e.g., GPT-4*). However, recently several researchers have warned against this approach, suggesting that model effectiveness may decrease when AI-generated data is used for training (Shumailov et al., 2024a; Shumailov et al., 2024b).

7

Final Thoughts and Research Opportunities

In this section we draw our discussion of mathematical information retrieval to a close, and identify some places where insights might be found in the future. For this purpose, we return to three key concerns that we have repeatedly returned to throughout the monograph:

1. the people that IR systems serve and interact with,
2. the information tasks that IR systems perform on behalf of people,
3. how systems and associated processes are evaluated.

Before we close, we also consider some limitations of MathIR systems from the perspective of individuals.

7.1 What is Next for MathIR?

To try and characterize the space of opportunities for future work in MathIR, we present two views summarizing topics discussed in the monograph where we could devote effort to strengthen existing approaches, and uncover new ones.

The first view is shown in Table 7.1, which presents the information needs for the people that play a role in the creation, use, and

Table 7.1: Information Needs for Mathematical Information Retrieval.

Group	Information Needs
Users	Query language operations and syntax
	Query/results interface usage
	How to organize and refind sources
	How to create, organize, and search annotations
	Data and tools for applying information in sources
	Tools for communicating information from sources
	Pragmatics: Stopping criteria for 'satisficing'
Assessors	Annotation interface operations and usage
	Relevance criteria and intended application
	Navigate, link, and search sources & annotations
	Pragmatics: 'reliable' annotation strategies
Designers & Researchers	Useful and novel:
	Data representations for formulas + text
	Retrieval interfaces
	Query/question mechanisms
	Annotation tools/protocols
	Machine learning models
	Retrieval models
	Data augmentation techniques
	Evaluation metrics, pooling, and protocols
	Conceptual frameworks / topic maps
	Test collections: creation, use, differences
	Implementation: planning and execution
	Pragmatics: 'good' end points for research/dev projects

evaluation of MathIR systems. These are the users, assessors, and designers/researchers for these systems, which we introduced in Section 1. Improving available information of the types listed, and/or making these information types more readily available or easy to use present research opportunities. Additional items that we note here that were not mentioned earlier include pragmatics. A critical requirement for productive knowledge work of the types shown in this table is knowing when to stop, and possibly reflect and report at that point. Improving our shared knowledge of actionable models and heuristics would be beneficial for people involved, and can help inform the use cases from which IR models and tools are designed and evaluated.

The second view shown in Table 7.2 is for the space of research opportunities addresses researchers and designers specifically. The rows and columns are organized by information task, and whether the opportunities are related to human interaction, system modules that perform information tasks, or evaluation of interaction and modules. We have indicated which sections present related material in the column headers. Please note that this is a first attempt at categorization along these dimensions. There are opportunities that we have undoubtably missed here.

Table 7.2: Future Directions for Mathematical Information Retrieval Research

Information Task	Human Interact. (Section 1)	System Modules (Sections 2, 4-6)	Evaluation (Section 3)
Retrieve			
R1. Query	Search interfaces Multi-modal input LLM prompts/RAG	Autocomplete Query suggestion Query perf. prediction	Log/User studies Module effectiveness Benefits of formulas?
R2. Consult	Formula/text navig. Math/text links Linked math/text	Math entity linking Text / formula tokenization	Log/User studies Math EL effects? Visual. effects?
Analyze			
A1. Annotate	User entity cards (*e.g.,* formulas) Passages in sources Source links/groups	Text + math represent. Formula represent.	Log/User studies Retrieval metrics User bandwidth? Storage requs.?
A2. Index	Org. cards, notes Org. passages, notes Org. searches Org. sources ('jar')	Sparse models Dense models Sparse + Dense (*e.g.,* per SPLADE)	Log/user studies Metrics: size, speed, effectiveness
Synthesize			
S1. Apply	Reusable formulas (*e.g.,* LaTeX chips) Tool integration (*e.g.,* CAS) Export sources & notes	Embedding models (Pre)training tasks Learning-to-Rank Model reduction (*e.g.,* for LLMs) Compression: Index, source, notes, etc.	Log/User studies Model effectiveness & efficiency
S2. Communicate	Math/text author. tools Summarization (*e.g.,* LLM)	Data augmentation Math-focused SERP Background-targeted answers	Log/User studies Test collections Research papers Shared task papers Surveys, task & concept models

Currently opportunities in machine learning (e.g., improved/compressed LLM models and prompting), improved formula tokenization and attention mechanisms for transformers, and finding new approaches to distilling contextual information into dense embedding vectors and other representations are important and widely recognized; these each realize improved applications of available statistical information. Some other important opportunities include improved query interfaces, annotation tools, and techniques for communication of search/answer results (e.g., sensitive to the searcher background). Mechanisms that improve a user's ability to organize, link, and quickly refind sources and previous results seem ripe for renewed exploration, provided that we are mindful of actual user needs, and not overwhelming the user. Integration of retrieval systems with mathematical tools such as Computer Algebra Systems (CAS) and theorem provers is another promising direction for interaction, and possibly also annotating queries and collections.

Another key area is the development of new representations for text and formulas, including formulas in isolation, but particularly representations for text, formulas, and possibly other graphics to better capture context for use in improved sparse and dense indexing.[1] Improved indexing and retrieval models remain an important opportunity as well, being fundamental to IR in general.

7.2 Limitations: What MathIR Cannot Provide

In our experiences working on MathIR, a frequently expressed concern or hope is that strong systems will remove the need for individuals to understand the math that they use. Instructors reasonably worry that some math problem sets can be completed by issuing prompts to LLMs or questions to search engines, with students receiving complete or near-complete answers. On the other side, math anxiety is common, and information from sources that help reduce mental effort and our risk of failure in addressing hard questions are appreciated, regardless of our math comfort level. So while copying retrieved answers is

[1]OPT and SLT variants, along with MathAMR and PHOC are described in Section 2, but many variations and other representations are unexplored.

clearly a problem, could technology instead accelerate our learning and understanding of new mathematical concepts?

For example, reliable MathIR systems could quickly retrieve or generate examples of questions and answers for similar problems without giving away answers. This is not cheating, and seems like an opportunity to save a lot of time. Unfortunately, after this retrieval step, we arrive at a fundamental limitation of technology well-known to math instructors.[2] Technology allows us to store, illustrate, organize, and retrieve information faster than in previous decades (National Council of Teachers of Mathematics, 2011). This can save significant amounts of human effort in terms of querying for and consulting sources, but related benefits for human learning have been modest (Cheung and Slavin, 2013). However sophisticated our technologies become, they reside outside of a person's mind, and there is evidence that deep understanding comes from effort exerted from inside a person's mind.

In one psychological study, students who constructed a formula themselves for the area of geometric shapes organized in a lattice were more likely to correctly adapt the formula for new shapes than students who were given the correct formula for the first case (Hallinen et al., 2021). Interestingly, the authors refer to constructing the correct formula as searching for the formula; the student must mentally create and compare alternative formulas as they work on the problem. Their study supports the idea that engaged exploration and comparison are needed for deep understanding. Naturally, such engaged thinking takes time.

To correctly use a source such as a search result hit or question answer, one must understand it first. Assessing the validity and applicability of an answer requires understanding of terminology, notation, and associated concepts. We see ample evidence for this on math CQA sites, where large numbers of posts and comments seek only clarification of questions and their context, with many asking for clarification of language and notation choices. Skimming a source online that we don't understand provides patterns that might suggest information we could

[2]These comments apply equally to technological resources for education in general, and more broadly, knowledge work.

use, and readily supports copying its contents. Using the same source to address a problem posed using different notation and terminology generally requires a deeper understanding of the material.

And so, students completing (non-trivial) problem sets using MathIR tools will still need to spend time and effort to evaluate returned answers, and to work through material on their own. For new and challenging topics, this process of working through the material will remain slow in comparison to issuing a query or question to a retrieval system. We do not anticipate that future MathIR systems will remove this requirement.[3]

As a more compact summary, here are some limitations for mathematical information retrieval systems:

1. Relevance is strongly influenced by searcher expertise and presentation in sources.
2. Retrieval provides sources, not the understanding of them.
3. Deep human understanding of information, e.g., for application in new contexts or problems, requires study, exploration, and experimentation – in other words, time.

That we feel this needs saying at all may surprise some readers, and perhaps some IR researchers more than anyone. Many other specialized search domains have similar challenges (e.g., law, medicine, chemistry). But our experience has been that mathematics, being both a powerful tool, and in many cultures a source of both pride and shame, sometimes leads to expressed hopes and concerns at odds with these limitations.

7.3 Some Parting Thoughts

The time at which this work is being written is an exciting time for mathematical information retrieval. One has a sense that we may be in for a very surprising leap forward soon, much as we saw for the game of Go, when AlphaGo applied monte carlo tree search, deep learning techniques, and reinforcement learning to play the game competitively

[3] Related to this, we believe that supporting, advancing, and sharing human understanding is fundamental for societally beneficial academic instruction and research of all kinds.

at the international level. Prior to this, many believed that competitive systems operating in such a vast search space would not be seen for a long time. AlphaGo was trained through simulating a very large number of games; the model parameters were learned via a somewhat brute force process of trial and error, in order to estimate reliable probabilities of success along different series of moves through empirical observation. Back to our information task model, finding efficient and effective ways to retrieve, analyze, and synthesize information about the game space and individual move configurations proved transformative.

With the advent of LLMs and other recent advances in machine learning, we might soon stumble upon effective algorithms that combine representations, statistics, and constraints in a way that allows complex mathematical information to be found relatively easily, and interactively using conversational search models. Related methods may automatically answer questions ably and in an audience-appropriate manner, and perhaps even produce proof strategies or complete proofs for mathematical conjectures of real sophistication. Such advances would finally realize some of the earliest goals of Artificial Intelligence research, and would have a large number of potential applications.

Even if such technologies do not come to pass in the near term, there is another important perspective for the future of MathIR. Even with incremental advances, there is an opportunity to create systems that help people by better aligning with their mathematical expertise and communication styles, and that make working with mathematical information just a bit more efficient, and just a bit more comfortable. We find this prospect equally exciting, and a challenging goal from the human, systems, and evaluation perspectives.

Acknowledgements

From Richard. Thanks to our colleagues from the MathSeer project (Douglas W. Oard, C. Lee Giles, Jian Wu, and Shaurya Rohatgi), and the MathSeer advisory board (Marti Hearst, Michael Kohlhase, and Frank Tompa) for providing helpful insights over many years. We also wish to thank the Alfred P. Sloan Foundation and National Science Foundation (USA) for supporting the MathSeer project (2017-2022).

Many students and staff have completed MIR-related research in the Document and Pattern Recognition Lab (dprl) at RIT. The approach taken in this monograph has been heavily influenced by their good work and good humour, and by their comments at over a decade of lab meetings. Before RIT, I had two excellent doctoral advisors at Queen's University in Canada, Dorothea Blostein and James R. Cordy. Their support and guidance were critical early in my career when I worked on recognizing mathematical notation, and then again when I began doing math retrieval work after I had arrived at RIT. The SLT and OPT formula representations used in this book were originally developed in collaboration with Dorothea and Jim.

Also, the influence of George Nagy can be seen in both the subject matter and form of this monograph: his mentorship profoundly impacted my career and approach to research, and without his utterly unique style of intervention this monograph would not exist.[4] George, I hope

[4]A research visit that opened with a promise to shop for a replacement house

that something in this work interests you, and ideally, also makes you laugh.

Another instructive experience was an invited talk that I gave at RMIT in Melbourne, at the invitation of Justin Zobel. Justin, thank you for the helpful discussions and support over the years, even when that talk turned out to be a sleep-deprived ramble. Thanks also for providing comments on the draft despite being tremendously busy.

In terms of bringing the topics of this monograph into better focus, we thank Jimmy Lin, who suggested adapting Broder's taxonomy to characterize mathematical information needs more concretely, and Susan Dumais, who helpfully reminded me that variable names carry little information without additional context. Her comment made us ask ourselves, 'exactly what information does a formula carry?' which led to the approach taken in Section 2.

The anonymous reviewers had numerous helpful comments, and their observations led to an improved organization of the monograph. Other students and colleagues offered helpful comments on drafts including Harold Mouchère, Bryan Amador, Patrick Philippy, Wei Zhong, and Nickvash Kani. Bryan in particular was instrumental in helping prioritize topics and improve the organization of the early sections.

I am deeply grateful to Aikiko Aizawa and Yiqun Liu for providing the opportunity to write this monograph, and particularly to Yiqun for his tremendous patience as I struggled to finish it. Thanks also to the good people at Now Publishers, including Alet Heezemans, Mike Casey, Mark de Jongh, and Anahita Gupta who were also helpful and patient despite repeated delays.

Behrooz and Anurag: thank you both for your many contributions to this work; the content is much more current, better illustrated, and better grounded mathematically as a result. Thanks also for hanging in there as what was supposed to be a 1 year project exceeded 3 years.

Most of all, I need to thank my wife and partner Katey, without whose reminders and support neither the time nor courage to finish this work would have ever been found. Thanks also to my father whose

fan if the first author's ideas were uninteresting. We later went shopping, and then discussed other research directions, of which math-aware search became the focus.

influence motivated me to write a monograph in the first place, and my mother and daughters Alix and Evalyn for their love and patience as I worked...a lot...on this monograph.

From Behrooz. I am grateful to Richard and Anurag for providing this wonderful opportunity and for their collaboration. I extend my thanks to my Ph.D. advisors, Richard Zanibbi and Douglas W. Oard, whose support and guidance have been instrumental in shaping my career. Their mentorship enabled me to contribute to this work. Lastly, I would like to thank my students at the Artificial Intelligence and Information Retrieval (AIIR) Lab at the University of Southern Maine for their valuable feedback.

From Anurag. The support and encouragement of many wonderful individuals has been instrumental in this journey. My deepest gratitude is to God for giving me an opportunity to learn mathematics and to my parents for instilling a love of knowledge in me. Nothing would have been the same had it not been for their infinite love and support in everything I did.

I would like to extend special thanks to Richard (the first author) who introduced me to the world of MIR. His support has been a constant source of strength. He offered invaluable advice, and was a constant source of inspiration throughout the writing process.

To Behrooz (the second author) and all the students who worked with us over the years on related projects, I am deeply grateful for your time, honesty, and thoughtful suggestions. You helped me see this story from perspectives I hadn’t considered.

Appendices

A

Online Resources

Links for some of the resources mentioned in this monograph may be found below in Table A.1.

Table A.1: Resources for Math Search and Question Answering

Name	Type	Link	Year
Tools			
LaTeXML	LaTeX-MathML Convertor	https://github.com/brucemiller/LaTeXML	2004
SnuggleTeX	LaTeX-MathML Convertor	https://github.com/davemckain/snuggletex	2008
LeanDojo	Theorem provers	https://github.com/lean-dojo/LeanDojo	2023
Test Collections			
NTCIR	Search Test Collection	https://ntcir-math.nii.ac.jp/data/	2013-16
ARQMath	Search Test Collection	https://www.cs.rit.edu/~dprl/ARQMath/	2020-22
AQUA-RAT	MWP Dataset	https://github.com/google-deepmind/AQuA	2017
MathQA	MWP Dataset	https://math-qa.github.io/	2019
ASDiv	MWP Dataset	https://github.com/chao-chun/nlu-asdiv-dataset.	2020
GSM8K	MWP Dataset	https://huggingface.co/datasets/openai/gsm8k	2021
MATH	MWP Dataset	https://github.com/hendrycks/math	2021
GeoQA	Geometry QA	https://github.com/chen-judge/GeoQA	2021
UniGeo	Geometry QA	https://github.com/chen-judge/UniGeo	2022
MathVista	Visual Math Problem	https://mathvista.github.io/	2024
MATH-Vision	Visual Math Problem	https://github.com/mathllm/MATH-V	2024
Systems			
MIAS	Formula Search	https://github.com/MIR-MU/MIaS	2011
Tangent-S	Formula Search	https://github.com/MattLangsenkamp/tangent-s	2017
Tangent-V	Formula Search	https://www.cs.rit.edu/~dprl/files/TangentV-source.zip	2019
Tangent-CFT	Formula Search	https://github.com/BehroozMansouri/TangentCFT	2019
NTFEM	Formula Search	https://github.com/NTFEM/Formulae-Embedding	2020
MathEmb	Formula Search	https://github.com/Franknewchen/MathEmb	2021
WikiMirs	Math-Aware Search	https://github.com/huxuan/WikiMirs	2016
Tangent-L	Math-Aware Search	https://github.com/fras2560/Tangent-L	2018
Approach0	Math-Aware Search	https://github.com/approach0/search-engine	2019
MathDowsers	Math-Aware Search	https://github.com/kiking0501/MathDowsers-ARQMath	2021
ALBERT-for-Math-AR	Math-Aware Search	https://github.com/AnReu/ALBERT-for-Math-AR	2021
MABOWDOR	Math-Aware Search	https://github.com/approach0/pya0	2023
CrossMath	Multilingual Math-Aware S.	https://github.com/jgore077/CrossMath	2024
LILA	QA Dataset/LLM	https://lila.apps.allenai.org/	2022
MAmmoTH	Math-LLM	https://github.com/TIGER-AI-Lab/MAmmoTH	2023
LLEMMA	Math-LLM	https://github.com/EleutherAI/math-lm	2024
ToRA	Math-LLM	https://github.com/microsoft/ToRA	2024
MetaMath	Math-LLM	https://github.com/meta-math/MetaMath	2024

B

Search and QA for Theorem Proving

In Sections 5 and 6 we discussed math-aware search and math question-answering. A topic closely related to these is theorem proving. Automated theorem proving aims to explore the application of computers in proving mathematical theorems. This is actually one of the earliest topics of interest in computer science. In the early 1960s, scientists started exploring the use of machines for theorem proving for the quantification theory (Davis and Putnam, 1960; Davis et al., 1962) using deduction rules to prove assertions.

Informal theorem proving refers to the way that humans approach proving theorems using reasoning through notation and natural language, likely with some missing details (e.g., assumed definitions) and skipped computational steps, for example. In contrast, formal theorem proving represents theorems in a machine-readable format, making verification by logical rules possible, at the cost of more information being explicitly stated (e.g., all variable types, definitions for all operators, etc.).

Converting informal to formal proof steps is referred to as Autoformalization. The Mizar[1] language is commonly used by mathemati-

[1]https://mizar.uwb.edu.pl/.

cians as a formal language for writing definitions and proofs. Using Mizar, Wang et al. (2018) explored converting (translation) of informal LaTeX-written text into formal Mizar language. This work explored different seq2seq architectures, with LSTM and attention providing the highest BLEU score for this translation.

Researchers have also explored other formalization languages, including applying large language models to generate statements in Codex (Chen et al., 2021b) has been studied recently (Wu et al., 2022). This translates statements in natural text into formalized theorems for the interactive proof assistant Isabelle (Wenzel et al., 2008). As an example, the system translated "Prove that there is no function f from the set of non-negative integers into itself such that f(f(n)) = n + 1987 for every n" perfectly to Codex as:

```
theorem
    fixes f :: "nat \<Rightarrow> nat"
    assumes "\<forall> n. f (f n) = n + 1987"
    shows False
```

Another task for theorem proving is retrieving useful lemmas that will help with proving steps, known as premise selection. This is a form of search problem, where relevance is defined in terms of suitability for proving a specific conjecture. The problem is defined as (Alama et al., 2014):

> Given an Automated Theorem Provers and a large number of premises, find premises that are useful to the prover for constructing a proof for a new conjecture.

DeepMath (Irving et al., 2016) is one of the earliest works to apply deep learning for this task. Conjecture and axiom sequences are embedded separately, concatenated, and then passed to a fully connected neural network for predicting the usefulness of the axiom. Embeddings are at character level for formulas, and word-level for statements defining symbols. The convolutional network model FormulaNet (Wang et al., 2017a) used a similar idea and applied graph neural networks. Using formulas in higher-order logic (Church, 1940), each formula is first parsed into an OPT: internal nodes represent a quantifier or a constant or variable function, and leaf nodes represent variable or constant values.

Edges connect a quantifier to all instances of its quantified variables. After creating the tree, a merging step is applied to merge leaf nodes representing the same constant/variable. Finally, a unification technique is used to replace variable names with ‘VAR’ and function names with ‘VARFUNC’. After building the graph, convolution or message passing is applied to get node embeddings. These embeddings are used with max-pooling to form an embedding for the graph.

As theorems are built upon existing mathematical knowledge, a graph representation of mathematical concept statements such as lemma, and definitions is a common approach for this task. One way of building this graph is to use statements as nodes and ordered edges from node s_1 to s_2 if there is statement 1 is a premise of statement 2 (Ferreira and Freitas, 2020a). With this definition of a graph, the problem can be viewed as link prediction, for which a Deep Graph Convolutional Neural Network (DGCNN) architecture was applied (Zhang et al., 2018). The textual content embedding of each node in this work is encoded using Doc2Vec (Le and Mikolov, 2014) model with mathematical concepts being encoded as linearized trees, with every sub-expression represented as a token.

For example, the formula $(x + y) \times c$ is represented as a sequence of tokens for subexpressions {‘x’, ‘y’, ‘(x + y)’, ‘(x + y) × c’} (similar to what is used to generate subexpression tokens for the WikiMirs search model discussed earlier). The authors later introduced STAtement Representation (STAR) cross-modal representation (Ferreira and Freitas, 2021), treating mathematical formulas as natural language. Each statement is viewed as a combination of words and formulas. However, for embedding, they proposed two separate self-attention layers, one for formulas and one for words. The output of the self-attention layers is then concatenated and passed to a Bi-LSTM to get the final representation of the statement. To find the relatedness score between conjecture and premises, a Siamese neural network was applied.

Generative and large language models have recently been applied for premise selection. LeanDojo (Yang et al., 2023) is an open-source toolkit based on the Lean[2] programming language, that introduces

[2]https://lean-lang.org/

ReProver (Retrieval-Augmented Prover). ReProver is a language model-based prover, augmented with retrieval for selecting premises. Given the initial state of proof, it retrieves a set of useful premises (set at 100) using a Dense Passage Retriever. These premises are concatenated to the initial state and passed to a fine-tuned ByT5 (Xue et al., 2022) model to generate steps toward the proof.

As the application of LLMs for theorem proving is gaining attention, new datasets are being introduced for theorem proving. NATURAL-PROOFS (Welleck et al., 2021) for example, is a multi-domain corpus of mathematical statements and their proofs, written in natural mathematical language. The main tasks in this dataset are: 1) mathematical reference retrieval: given a theorem, retrieve a set of references that occur in the theorem proof, and 2) mathematical theorem generation: generate the sequence of references that occur in a given theorem's proof. Table B.1 summarizes some of the existing resources for this task.

Table B.1: Resources for Math Search and Question Answering

Name	Type	Link	Year
NaturalProof	Proof Dataset	https://github.com/wellecks/naturalproofs	2021
FormulaNet	Theorem Proving	https://github.com/princeton-vl/FormulaNet	2017
STAR	Theorem Proving	https://github.com/ai-systems/crossmodal_embedding	2021

References

Ahmed, S., K. Davila, S. Setlur, and V. Govindaraju. (2021). "Equation Attention Relationship Network (EARN) : A Geometric Deep Metric Framework for Learning Similar Math Expression Embedding". In: 2020 25th International Conference on Pattern Recognition (ICPR). Milan, Italy: IEEE. 6282–6289. doi: 10.1109/ICPR48806.2021.9412619.

Aizawa, A., M. Kohlhase, and I. Ounis. (2013). "NTCIR-10 Math Pilot Task Overview". In: Proc. NTCIR-10. 1–8.

Aizawa, A., M. Kohlhase, and I. Ounis. (2014). "NTCIR-11 Math-2 Task Overview". In: Proc. NTCIR-11. 88–98.

Alama, J., T. Heskes, D. Kühlwein, E. Tsivtsivadze, and J. Urban. (2014). "Premise Selection for Mathematics by Corpus Analysis and Kernel Methods". Journal of Automated Reasoning. 52(2): 191–213. doi: 10.1007/s10817-013-9286-5.

Alexeeva, M., R. Sharp, M. A. Valenzuela-Escárcega, J. Kadowaki, A. Pyarelal, and C. Morrison. (2020). "MathAlign: Linking Formula Identifiers to their Contextual Natural Language Descriptions". English. In: Proceedings of the Twelfth Language Resources and Evaluation Conference. Marseille, France: European Language Resources Association. 2204–2212. url: https://aclanthology.org/2020.lrec-1.269.

Almazán, J., A. Gordo, A. Fornés, and E. Valveny. (2014). "Word Spotting and Recognition with Embedded Attributes". IEEE Trans. Pattern Anal. Mach. Intell. 36(12): 2552–2566. doi: 10.1109/TPAMI.2014.2339814.

Amador, B., M. Langsenkamp, A. Dey, A. K. Shah, and R. Zanibbi. (2023). "Searching the ACL Anthology with Math Formulas and Text". In: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023. ACM. 3110–3114. doi: 10.1145/3539618.3591803.

Amini, A., S. Gabriel, S. Lin, R. Koncel-Kedziorski, Y. Choi, and H. Hajishirzi. (2019). "MathQA: Towards Interpretable Math Word Problem Solving with Operation-Based Formalisms". In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics. 2357–2367. doi: 10.18653/v1/N19-1245.

Asakura, T., Y. Miyao, and A. Aizawa. (2022). "Building Dataset for Grounding of Formulae — Annotating Coreference Relations Among Math Identifiers". In: Proceedings of the Thirteenth Language Resources and Evaluation Conference. Marseille, France: European Language Resources Association. 4851–4858. url: https://aclanthology.org/2022.lrec-1.519.

Asakura, T., Y. Miyao, A. Aizawa, and M. Kohlhase. (EasyChair, 2021). MioGatto: A Math Identifier-Oriented Grounding Annotation Tool. EasyChair Preprint 6209.

Avenoso, R. (2021). "Spatial vs. Graph-Based Formula Retrieval". Master's thesis. Rochester, NY, USA: Rochester Institute of Technology.

Avenoso, R., B. Mansouri, and R. Zanibbi. (2021). "XY-PHOC Symbol Location Embeddings for Math Formula Retrieval and Autocompletion". In: Proceedings of the Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, Bucharest, Romania, September 21st - to - 24th, 2021. Vol. 2936. CEUR Workshop Proceedings. CEUR-WS.org. 25–35. url: https://ceur-ws.org/Vol-2936/paper-02.pdf.

Azerbayev, Z., H. Schoelkopf, K. Paster, M. D. Santos, S. McAleer, A. Q. Jiang, J. Deng, S. Biderman, and S. Welleck. (2024). "Llemma: An Open Language Model For Mathematics". url: https://arxiv.org/abs/2310.10631.

Baker, J. B., A. P. Sexton, and V. Sorge. (2010). "Faithful mathematical formula recognition from PDF documents". In: The Ninth IAPR International Workshop on Document Analysis Systems, DAS 2010, June 9-11, 2010, Boston, Massachusetts, USA. ACM International Conference Proceeding Series. ACM. 485–492. doi: 10.1145/1815330.1815393.

Banarescu, L., C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider. (2013). "Abstract Meaning Representation for Sembanking". In: Proceedings of the 7th linguistic annotation workshop and interoperability with discourse. 178–186.

Bates, M. J. (1989). "The Design of Browsing and Berrypicking Techniques for the Online Search Interface". Online review.

Belkin, N. J. (1980). "Anomalous States of Knowledge as a Basis for Information Retrieval". Canadian journal of information science. 5(1): 133–143.

Bender, E. M., T. Gebru, A. McMillan-Major, and S. Shmitchell. (2021). "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?" In: FAccT '21: 2021 ACM Conference on Fairness, Accountability, and Transparency, Virtual Event / Toronto, Canada, March 3-10, 2021. ACM. 610–623. doi: 10.1145/3442188.3445922.

Blei, D. M., A. Y. Ng, and M. I. Jordan. (2003). "Latent Dirichlet Allocation". J. Mach. Learn. Res. 3: 993–1022. url: http://jmlr.org/papers/v3/blei03a.html.

Bobrow, D. G. (1964). "A question-answering system for high school algebra word problems". In: Proceedings of the October 27-29, 1964, Fal l Joint Computer Conference, Part I. AFIPS '64 (Fal l, part I). San Francisco, California: Association for Computing Machinery. 591–614. doi: 10.1145/1464052.1464108.

Bojanowski, P., E. Grave, A. Joulin, and T. Mikolov. (2017). "Enriching Word Vectors with Subword Information". Transactions of the Association for Computational Linguistics. 5(Dec.): 135–146. doi: 10.1162/tacl_a_00051.

Borlund, P. (2003). "The IIR Evaluation Model: A Framework For Evaluation of Interactive Information Retrieval Systems". Inf. Res. 8(3). url: http://www.informationr.net/ir/8-3/paper152.html.

Bozzon, A. and P. Fraternali. (2010). "Chapter 8: Multimedia and Multimodal Information Retrieval". In: Search Computing: Challenges and Directions. Berlin, Heidelberg: Springer Berlin Heidelberg. 135–155. doi: 10.1007/978-3-642-12310-8_8.

Broder, A. Z. (2002). "A Taxonomy of Web Search". SIGIR Forum. 36(2): 3–10. doi: 10.1145/792550.792552.

Buckley, C. and E. M. Voorhees. (2004). "Retrieval Evaluation with Incomplete Information". In: Proceedings of the 27th annual international conference on Research and development in information retrieval - SIGIR '04. Sheffield, United Kingdom: ACM Press. 25. doi: 10.1145/1008992.1009000.

Chen, J., T. Li, J. Qin, P. Lu, L. Lin, C. Chen, and X. Liang. (2022a). "UniGeo: Unifying Geometry Logical Reasoning via Reformulating Mathematical Expression". In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics. 3313–3323. doi: 10.18653/v1/2022.emnlp-main.218.

Chen, J., J. Tang, J. Qin, X. Liang, L. Liu, E. Xing, and L. Lin. (2021a). "GeoQA: A Geometric Question Answering Benchmark Towards Multimodal Numerical Reasoning". In: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021. Online: Association for Computational Linguistics. 513–523. doi: 10.18653/v1/2021.findings-acl.46.

Chen, M., J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. (2021b). "Evaluating large language models trained on code". arXiv preprint arXiv:2107.03374.

Chen, W., X. Ma, X. Wang, and W. W. Cohen. (2022b). “Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks”. arXiv preprint arXiv:2211.12588.

Cheung, A. C. and R. E. Slavin. (2013). “The Effectiveness of Educational Technology Applications for Enhancing Mathematics Achievement in K-12 Classrooms: A Meta-Analysis”. Educational Research Review. 9(June): 88–113. doi: 10.1016/j.edurev.2013.01.001.

Church, A. (1940). “A Formulation of the Simple Theory of Types”. The journal of symbolic logic. 5(2): 56–68.

Cobbe, K., V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. (2021). “Training Verifiers to Solve Math Word Problems”. url: https://arxiv.org/abs/2110.14168.

Croft, W. B., D. Metzler, and T. Strohman. (2009). Search Engines - Information Retrieval in Practice. Pearson Education. url: http://www.search-engines-book.com/.

Dadure, P., P. Pakray, and S. Bandyopadhyay. (2024). “Mathematical Information Retrieval: A Review”. ACM Comput. Surv. 57(3). doi: 10.1145/3699953.

Dai, Y., L. Chen, and Z. Zhang. (2020). “An N-ary Tree-based Model for Similarity Evaluation on Mathematical Formulae”. In: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC). Toronto, ON, Canada: IEEE. 2578–2584. doi: 10.1109/SMC42975.2020.9283495.

Davila, K., R. Joshi, S. Setlur, V. Govindaraju, and R. Zanibbi. (2019). “Tangent-V: Math Formula Image Search Using Line-of-Sight Graphs”. In: Advances in Information Retrieval. Vol. 11437. Cham: Springer International Publishing. 681–695. doi: 10.1007/978-3-030-15712-8_44.

Davila, K., S. Ludi, and R. Zanibbi. (2014). “Using Off-line Features and Synthetic Data for On-line Handwritten Math Symbol Recognition”. In: 2014 14th International Conference on Frontiers in Handwriting Recognition. IEEE. 323–328.

Davila, K., F. Xu, S. Setlur, and V. Govindaraju. (2021). "FCN-LectureNet: Extractive Summarization of Whiteboard and Chalkboard Lecture Videos". IEEE Access. 9: 104469–104484. doi: 10.1109/ACCESS.2021.3099427.

Davila, K. and R. Zanibbi. (2017a). "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. Shinjuku Tokyo Japan: ACM. 1165–1168. doi: 10.1145/3077136.3080748.

Davila, K. and R. Zanibbi. (2017b). "Whiteboard Video Summarization via Spatio-Temporal Conflict Minimization". In: 14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, November 9-15, 2017. IEEE. 355–362. doi: 10.1109/ICDAR.2017.66.

Davila, K. and R. Zanibbi. (2018). "Visual Search Engine for Handwritten and Typeset Math in Lecture Videos and LATEX Notes". In: 16th International Conference on Frontiers in Handwriting Recognition, ICFHR 2018, Niagara Fal ls, NY, USA, August 5-8, 2018. IEEE Computer Society. 50–55. doi: 10.1109/ICFHR-2018.2018.00018.

Davis, M., G. Logemann, and D. Loveland. (1962). "A Machine Program for Theorem-proving". Commun. ACM. 5(7): 394–397. doi: 10.1145/368273.368557.

Davis, M. and H. Putnam. (1960). "A Computing Procedure for Quantification Theory". J. ACM. 7(3): 201–215. doi: 10.1145/321033.321034.

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova. (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics. 4171–4186. doi: 10.18653/v1/N19-1423.

Diaz, Y. (2021). "Introducing Handwriting into a Multimodal LaTeX Formula Editor". Master's thesis. Rochester Institute of Technology.

Diaz, Y., G. Nishizawa, B. Mansouri, K. Davila, and R. Zanibbi. (2021). "The MathDeck Formula Editor: Interactive Formula Entry Combining LaTeX , Structure Editing, and Search". In: Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems. Yokohama Japan: ACM. 1–5. doi: 10.1145/3411763.3451564.

Dmello, A. (2019). "Representing Mathematical Concepts Associated With Formulas Using Math Entity Cards". Master's thesis. Rochester Institute of Technology.

Dua, D., Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner. (2019). "DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs". url: https://arxiv.org/abs/1903.00161.

Duncan, D. (2021). Index, a History of the. New York: Norton.

Durgin, S., J. Gore, and B. Mansouri. (2024). "MathMex: Search Engine for Math Definitions". In: Advances in Information Retrieval - 46th European Conference on Information Retrieval, ECIR 2024, Glasgow, UK, March 24-28, 2024, Proceedings, Part V. Vol. 14612. Lecture Notes in Computer Science. Springer. 194–199. doi: 10.1007/978-3-031-56069-9_17.

Ethayarajh, K., W. Xu, N. Muennighoff, D. Jurafsky, and D. Kiela. (2024). "KTO: Model Alignment as Prospect Theoretic Optimization". url: https://arxiv.org/abs/2402.01306.

Faggioli, G., L. Dietz, C. L. A. Clarke, G. Demartini, M. Hagen, C. Hauff, N. Kando, E. Kanoulas, M. Potthast, B. Stein, and H. Wachsmuth. (2024). "Who Determines What Is Relevant? Humans or AI? Why Not Both?" Commun. ACM. 67(4): 31–34. doi: 10.1145/3624730.

Ferreira, D. and A. Freitas. (2020a). "Premise Selection in Natural Language Mathematical Texts". In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Online: Association for Computational Linguistics. 7365–7374. doi: 10.18653/v1/2020.acl-main.657.

Ferreira, D. and A. Freitas. (2021). “STAR: Cross-modal [STA]tement [R]epresentation for selecting relevant mathematical premises”. In: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume. Online: Association for Computational Linguistics. 3234–3243. doi: 10.18653/v1/2021.eacl-main.282.

Ferreira, D. and A. Freitas. (2020b). “Natural Language Premise Selection: Finding Supporting Statements for Mathematical Text”. url: https://arxiv.org/abs/2004.14959.

Formal, T., B. Piwowarski, and S. Clinchant. (2021). “SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking”. In: SIGIR ’21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021. ACM. 2288–2292. doi: 10.1145/3404835.3463098.

Fraser, D., A. Kane, and F. W. Tompa. (2018). “Choosing Math Features for BM25 Ranking with Tangent-L”. In: Proceedings of the ACM Symposium on Document Engineering 2018. Halifax NS Canada: ACM. 1–10. doi: 10.1145/3209280.3209527.

Freund, Y., R. Iyer, R. E. Schapire, and Y. Singer. (2003). “An Efficient Boosting Algorithm for Combining Preferences”. Journal of machine learning research. 4(Nov): 933–969.

Fuhr, N. (2017). “Some Common Mistakes In IR Evaluation, And How They Can Be Avoided”. SIGIR Forum. 51(3): 32–41. doi: 10.1145/3190580.3190586.

Gao, L., Z. Jiang, Y. Yin, K. Yuan, Z. Yan, and Z. Tang. (2017). “Preliminary Exploration of Formula Embedding for Mathematical Information Retrieval: can mathematical formulae be embedded like a natural language?” arXiv:1707.05154 [cs]. Aug. url: http://arxiv.org/abs/1707.05154.

Gao, L., K. Yuan, Y. Wang, Z. Jiang, and Z. Tang. (2016). “The Math Retrieval System of ICST for NTCIR-12 MathIR Task”. In: NTCIR.

Gao, L., A. Madaan, S. Zhou, U. Alon, P. Liu, Y. Yang, J. Callan, and G. Neubig. (2023). “PAL: Program-aided Language Models”. In: Proceedings of the 40th International Conference on Machine Learning. Vol. 202. Proceedings of Machine Learning Research. PMLR. 10764–10799. url: https://proceedings.mlr.press/v202/gao23f.html.

Gao, S. and Y.-K. D. Ng. (2023). “Recommending Answers to Math Questions Based on KL-Divergence and Approximate XML Tree Matching”. In: Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region. SIGIR-AP ’23. Beijing, China: Association for Computing Machinery. 21–31. doi: 10.1145/3624918.3625337.

Giacalone, B., G. Paiement, Q. Tucker, and R. Zanibbi. (2024). “Beneath the [MASK]: An Analysis of Structural Query Tokens in ColBERT”. In: Advances in Information Retrieval - 46th European Conference on Information Retrieval, ECIR 2024, Glasgow, UK, March 24-28, 2024, Proceedings, Part III. Vol. 14610. Lecture Notes in Computer Science. Springer. 431–439. doi: 10.1007/978-3-031-56063-7_35.

Gore, J., J. Polletta, and B. Mansouri. (2024). “CrossMath: Towards Cross-lingual Math Information Retrieval”. In: The 10th ACM SIGIR / The 14th International Conference on the Theory of Information Retrieval. url: https://openreview.net/forum?id=IYcJxgnJ9Q.

Gou, Z., Z. Shao, Y. Gong, Y. Shen, Y. Yang, M. Huang, N. Duan, and W. Chen. (2024). “ToRA: A Tool-Integrated Reasoning Agent for Mathematical Problem Solving”. url: https://arxiv.org/abs/2309.17452.

Graf, P. (1995). “Substitution Tree Indexing”. In: Rewriting Techniques and Applications. Springer Berlin Heidelberg. 117–131.

Greiner-Petter, A., M. Schubotz, C. Breitinger, P. Scharpf, A. Aizawa, and B. Gipp. (2023). “Do the Math: Making Mathematics in Wikipedia Computable”. IEEE Trans. Pattern Anal. Mach. Intell. 45(4): 4384–4395. doi: 10.1109/TPAMI.2022.3195261.

Grossman, J., Z. Lin, H. Sheng, J. T.-Z. Wei, J. J. Williams, and S. Goel. (2019). “MathBot: Transforming Online Resources for Learning Math into Conversational Interactions”. AAAI 2019 Story-Enabled Intelligence.

Grover, A. and J. Leskovec. (2016). “node2vec: Scalable Feature Learning for Networks”. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 855–864.

Guidi, F. and C. Sacerdoti Coen. (2016). "A Survey on Retrieval of Mathematical Knowledge". Mathematics in Computer Science. 10(4): 409–427. doi: 10.1007/s11786-016-0274-0.

Ha, J., R. Haralick, and I. Phillips. (1995). "Recursive XY Cut using Bounding Boxes of Connected Components". In: Proceedings of 3rd International Conference on Document Analysis and Recognition. Vol. 2. 952–955 vol.2. doi: 10.1109/ICDAR.1995.602059.

Hallinen, N. R., L. N. Sprague, K. P. Blair, R. M. Adler, and N. S. Newcombe. (2021). "Finding Formulas: Does Active Search Facilitate Appropriate Generalization?" Cognitive Research: Principles and Implications. 6(1): 50. doi: 10.1186/s41235-021-00316-y.

Hambasan, R., M. Kohlhase, and C. Prodescu. (2014). "MathWebSearch at NTCIR-11". In: Proc. NTCIR-11. 114–119.

Hamilton, W., Z. Ying, and J. Leskovec. (2017). "Inductive Representation Learning on Large Graphs". In: Advances in Neural Information Processing Systems. Vol. 30. Curran Associates, Inc. url: https://proceedings.neurips.cc/paper_files/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.

He, K., X. Zhang, S. Ren, and J. Sun. (2016). "Deep Residual Learning for Image Recognition". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Head, A., K. Lo, D. Kang, R. Fok, S. Skjonsberg, D. S. Weld, and M. A. Hearst. (2021). "Augmenting Scientific Papers with Just-in-Time, Position-Sensitive Definitions of Terms and Symbols". In: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems. Yokohama Japan: ACM. 1–18. doi: 10.1145/3411764.3445648.

Hearst, M. (2009). Search User Interfaces. Cambridge university press.

Hendrycks, D., C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. (2021). "Measuring Mathematical Problem Solving With the MATH Dataset". url: https://arxiv.org/abs/2103.03874.

Holmes, S., A. Moorhead, R. Bond, H. Zheng, V. Coates, and M. McTear. (2019). “Usability Testing of a Healthcare Chatbot: Can we use Conventional Methods to Assess Conversational user Interfaces?” In: Proc. 31st European Conference on Cognitive Ergonomics. 207–214.

Hopkins, M., R. Le Bras, C. Petrescu-Prahova, G. Stanovsky, H. Hajishirzi, and R. Koncel-Kedziorski. (2019). “SemEval-2019 Task 10: Math Question Answering”. In: Proceedings of the 13th International Workshop on Semantic Evaluation. Minneapolis, Minnesota, USA: Association for Computational Linguistics. 893–899. doi: 10.18653/v1/S19-2153.

Hosseini, M. J., H. Hajishirzi, O. Etzioni, and N. Kushman. (2014). “Learning to Solve Arithmetic Word Problems with Verb Categorization”. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar: Association for Computational Linguistics. 523–533. doi: 10.3115/v1/D14-1058.

Hu, X., L. Gao, X. Lin, Z. Tang, X. Lin, and J. B. Baker. (2013). “WikiMirs: A Mathematical Information Retrieval System for Wikipedia”. In: Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries - JCDL ’13. Indianapolis, Indiana, USA: ACM Press. 11. doi: 10.1145/2467696.2467699.

Huang, D., S. Shi, C.-Y. Lin, J. Yin, and W.-Y. Ma. (2016). “How well do Computers Solve Math Word Problems? Large-Scale Dataset Construction and Evaluation”. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Berlin, Germany: Association for Computational Linguistics. 887–896. doi: 10.18653/v1/P16-1084.

Irving, G., C. Szegedy, A. A. Alemi, N. Een, F. Chollet, and J. Urban. (2016). “DeepMath - Deep Sequence Models for Premise Selection”. In: Advances in Neural Information Processing Systems. Vol. 29. Curran Associates, Inc. url: https://proceedings.neurips.cc/paper_files/paper/2016/file/f197002b9a0853eca5e046d9ca4663d5-Paper.pdf.

Ito, T., T. Matsuzaki, and S. Sato. (2017). "Coreference Resolution on Math Problem Text in Japanese". In: Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers). Taipei, Taiwan: Asian Federation of Natural Language Processing. 373–377. url: https://aclanthology.org/I17-2063.

Jiang, A. Q., A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, et al. (2023). "Mistral 7B". arXiv preprint arXiv:2310.06825.

Jo, H., D. Kang, A. Head, and M. A. Hearst. (2021). "Modeling Mathematical Notation Semantics in Academic Papers". In: Findings of the Association for Computational Linguistics: EMNLP 2021. Punta Cana, Dominican Republic: Association for Computational Linguistics. 3102–3115. doi: 10.18653/v1/2021.findings-emnlp.266.

Joachims, T. (2006). "Training Linear SVMs in Linear Time". In: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. 217–226.

Jones, K. S. (1972). "A Statistical Interpretation of Term Specificity and its Application in Retrieval". J. Documentation. 28(1): 11–21. url: https://api.semanticscholar.org/CorpusID:2996187.

Kamali, S. and F. W. Tompa. (2013). "Retrieving Documents with Mathematical Content". In: Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval. Dublin Ireland: ACM. 353–362. doi: 10.1145/2484028.2484083.

Kane, A., Y. K. Ng, and F. W. Tompa. (2022). "Dowsing for Answers to Math Questions: Doing Better with Less". In: CLEF (Working Notes). 40–62.

Karpukhin, V., B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih. (2020). "Dense Passage Retrieval for Open-Domain Question Answering". In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). Online: Association for Computational Linguistics. 6769–6781. doi: 10.18653/v1/2020.emnlp-main.550.

Khattab, O. and M. Zaharia. (2020). "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. SIGIR '20. Virtual Event, China: Association for Computing Machinery. 39–48. doi: 10.1145/3397271.3401075.

Kirsch, D. (2010). "Detexify: Recognition of Hand-drawn LATEX Symbols". Master's thesis. University of Münster (in German).

Kohlhase, A. and M. Kohlhase. (2007). "Reexamining the MKM Value Proposition: From Math Web Search to Math Web ReSearch". In: Towards Mechanized Mathematical Assistants, 14th Symposium, Calculemus 2007, 6th International Conference, MKM 2007, Hagenberg, Austria, June 27-30, 2007, Proceedings. Vol. 4573. Lecture Notes in Computer Science. Springer. 313–326. doi: 10.1007/978-3-540-73086-6_25.

Kohlhase, M. and C. Prodescu. (2013). "MathWebSearch at NTCIR-10". In: Proc. NTCIR-10. 675–679.

Kohlhase, M. and I. Sucan. (2006). "A Search Engine for Mathematical Formulae". In: Artificial Intelligence and Symbolic Computation. Vol. 4120. Berlin, Heidelberg: Springer Berlin Heidelberg. 241–253. doi: 10.1007/11856290_21.

Koncel-Kedziorski, R., H. Hajishirzi, A. Sabharwal, O. Etzioni, and S. D. Ang. (2015). "Parsing Algebraic Word Problems into Equations". Transactions of the Association for Computational Linguistics. 3: 585–597.

Kristianto, G. Y. and A. Aizawa. (2014). "Extracting Textual Descriptions of Mathematical Expressions in Scientific Papers". D-Lib Magazine. 20(11): 9.

Kristianto, G. Y., G. Topic, and A. Aizawa. (2016a). "MCAT Math Retrieval System for NTCIR-12 MathIR Task": 323–330.

Kristianto, G. Y., G. Topic, and A. Aizawa. (2017). "Utilizing dependency relationships between math expressions in math IR". Inf. Retr. J. 20(2): 132–167. doi: 10.1007/S10791-017-9296-8.

Kristianto, G. Y., G. Topić, and A. Aizawa. (2016b). "Entity Linking for Mathematical Expressions in Scientific Documents". In: Digital Libraries: Knowledge, Information, and Data in an Open Access Society. Cham: Springer International Publishing. 144–149.

Krstovski, K. and D. M. Blei. (2018). "Equation Embeddings". arXiv:1803.09123 [cs, stat]. Mar. url: http://arxiv.org/abs/1803.09123.

Lai, V., A. Pouran Ben Veyseh, F. Dernoncourt, and T. Nguyen. (2022). "SemEval 2022 Task 12: Symlink - Linking Mathematical Symbols to their Descriptions". In: Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022). Seattle, United States: Association for Computational Linguistics. 1671–1678. doi: 10.18653/v1/2022.semeval-1.230.

Langkilde, I. and K. Knight. (1998). "Generation that Exploits Corpus-Based Statistical Knowledge". In: 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, COLING-ACL '98, August 10-14, 1998, Université de Montréal, Montréal, Quebec, Canada. Proceedings of the Conference. Morgan Kaufmann Publishers / ACL. 704–710. doi: 10.3115/980845.980963.

Langsenkamp, M., B. Amador, and R. Zanibbi. (2024). "A Study of PHOC Spatial Region Configurations for Math Formula Retrieval". Tech. rep. doi: 10.48550/ARXIV.2408.09283. arXiv: 2408.09283.

Langsenkamp, M., B. Mansouri, and R. Zanibbi. (2022). "Expanding Spatial Regions and Incorporating IDF for PHOC-Based Math Formula Retrieval at ARQMath-3". In: Proceedings of the Working Notes of CLEF 2022 - Conference and Labs of the Evaluation Forum, Bologna, Italy, September 5th - to - 8th, 2022. Vol. 3180. CEUR Workshop Proceedings. CEUR-WS.org. 63–82. url: https://ceur-ws.org/Vol-3180/paper-04.pdf.

Le, Q. and T. Mikolov. (2014). "Distributed Representations of Sentences and Documents". In: Proceedings of the 31st International Conference on Machine Learning. Vol. 32. Proceedings of Machine Learning Research. No. 2. Bejing, China: PMLR. 1188–1196. url: https://proceedings.mlr.press/v32/le14.html.

Lewkowycz, A., A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo, Y. Wu, B. Neyshabur, G. Gur-Ari, and V. Misra. (2022). "Solving Quantitative Reasoning Problems with Language Models". In: Advances in Neural Information Processing Systems. Vol. 35. Curran Associates, Inc. 3843–3857. url: https://proceedings.neurips.cc/paper_files/paper/2022/file/18abbeef8cfe9203fdf9053c9c4fe191-Paper-Conference.pdf.

Liang, Z., J. Zhang, L. Wang, W. Qin, Y. Lan, J. Shao, and X. Zhang. (2022). "MWP-BERT: Numeracy-Augmented Pre-training for Math Word Problem Solving". In: Findings of the Association for Computational Linguistics: NAACL 2022. Seattle, United States: Association for Computational Linguistics. 997–1009. doi: 10.18653/v1/2022.findings-naacl.74.

Libbrecht, P. and E. Melis. (2006). "Methods to Access and Retrieve Mathematical Content in ActiveMath". In: Mathematical Software - ICMS 2006. Vol. 4151. Berlin, Heidelberg: Springer Berlin Heidelberg. 331–342. doi: 10.1007/11832225_33.

Ling, W., D. Yogatama, C. Dyer, and P. Blunsom. (2017). "Program Induction by Rationale Generation: Learning to Solve and Explain Algebraic Word Problems". In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Vancouver, Canada: Association for Computational Linguistics. 158–167. doi: 10.18653/v1/P17-1015.

Líška, M., P. Sojka, M. Ruzicka, and P. Mravec. (2011). "Web Interface and Collection for Mathematical Retrieval : WebMIaS and MREC". In: Towards a Digital Mathematics Library. Bertinoro, Bertinoro, Italy: Masaryk University Press (Brno, Czech Republic). 77–84.

Liu, Y., K. Ding, and Y. Zhou. (2019). "AiFu at SemEval-2019 Task 10: A Symbolic and Sub-symbolic Integrated System for SAT Math Question Answering". In: Proceedings of the 13th International Workshop on Semantic Evaluation. Minneapolis, Minnesota, USA: Association for Computational Linguistics. 900–906. doi: 10.18653/v1/S19-2154.

Lu, P., H. Bansal, T. Xia, J. Liu, C. Li, H. Hajishirzi, H. Cheng, K.-W. Chang, M. Galley, and J. Gao. (2024). "MathVista: Evaluating Mathematical Reasoning of Foundation Models in Visual Contexts". url: https://arxiv.org/abs/2310.02255.

Lu, P., L. Qiu, K.-W. Chang, Y. N. Wu, S.-C. Zhu, T. Rajpurohit, P. Clark, and A. Kalyan. (2023). "Dynamic Prompt Learning via Policy Gradient for Semi-structured Mathematical Reasoning". url: https://arxiv.org/abs/2209.14610.

Lv, Y. and C. Zhai. (2011). "Lower-Bounding Term Frequency Normalization". In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management. 7–16.

Macina, J., N. Daheim, S. Chowdhury, T. Sinha, M. Kapur, I. Gurevych, and M. Sachan. (2023). "MathDial: A Dialogue Tutoring Dataset with Rich Pedagogical Properties Grounded in Math Reasoning Problems". In: Findings of the Association for Computational Linguistics: EMNLP 2023. Singapore: Association for Computational Linguistics. 5602–5621. doi: 10.18653/v1/2023.findings-emnlp.372.

Manning, C. D., P. Raghavan, and H. Schütze. (2008). Introduction to information retrieval. Cambridge University Press. doi: 10.1017/CBO9780511809071.

Mansouri, B. and Z. Jahedibashiz. (2023). "Clarifying Questions in Math Information Retrieval". In: Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval. ICTIR '23. Taipei, Taiwan: Association for Computing Machinery. 149–158. doi: 10.1145/3578337.3605123.

Mansouri, B. and R. Maarefdoust. (2024). "Using Large Language Models for Math Information Retrieval". In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '24. Washington DC, USA: Association for Computing Machinery. 2693–2697. doi: 10.1145/3626772.3657907.

Mansouri, B., V. Novotný, A. Agarwal, D. W. Oard, and R. Zanibbi. (2022a). "Overview of ARQMath-3 (2022): Third CLEF Lab on Answer Retrieval for Questions on Math". In: Experimental IR Meets Multilinguality, Multimodality, and Interaction. Cham: Springer International Publishing. 286–310.

Mansouri, B., V. Novotný, A. Agarwal, D. W. Oard, and R. Zanibbi. (2022b). "Overview of ARQMath-3 (2022): Third CLEF Lab on Answer Retrieval for Questions on Math (Working Notes Version)". In: Proceedings of the Working Notes of CLEF 2022 - Conference and Labs of the Evaluation Forum, Bologna, Italy, September 5th - to - 8th, 2022. Vol. 3180. CEUR Workshop Proceedings. CEUR-WS.org. 1–27. url: https://ceur-ws.org/Vol-3180/paper-01.pdf.

Mansouri, B., D. W. Oard, A. Agarwal, and R. Zanibbi. (2021a). "Effects of Context, Complexity, and Clustering on Evaluation for Math Formula Retrieval". CoRR. abs/2111.10504. arXiv: 2111.10504. url: https://arxiv.org/abs/2111.10504.

Mansouri, B., D. W. Oard, and R. Zanibbi. (2020). "DPRL Systems in the CLEF 2020 ARQMath Lab". CEUR Workshop Proceedings 2696. Ed. by L. Cappellato, C. Eickhoff, N. Ferro, and A. Névéol. url: https://ceur-ws.org/Vol-2696/paper%5C_223.pdf.

Mansouri, B., D. W. Oard, and R. Zanibbi. (2022c). "Contextualized Formula Search Using Math Abstract Meaning Representation". In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022. ACM. 4329–4333. doi: 10.1145/3511808.3557567.

Mansouri, B., D. W. Oard, and R. Zanibbi. (2022d). "Contextualized Formula Search Using Math Abstract Meaning Representation". In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. CIKM '22. Atlanta, GA, USA: Association for Computing Machinery. 4329–4333. doi: 10.1145/3511808.3557567.

Mansouri, B., S. Rohatgi, D. W. Oard, J. Wu, C. L. Giles, and R. Zanibbi. (2019a). "Tangent-CFT: An Embedding Model for Mathematical Formulas". In: Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval. Santa Clara CA USA: ACM. 11–18. doi: 10.1145/3341981.3344235.

Mansouri, B., R. Zanibbi, and D. W. Oard. (2019b). "Characterizing Searches for Mathematical Concepts". In: 2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL). Champaign, IL, USA: IEEE. 57–66. doi: 10.1109/JCDL.2019.00019.

Mansouri, B., R. Zanibbi, and D. W. Oard. (2021b). “Learning to Rank for Mathematical Formula Retrieval”. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. Virtual Event Canada: ACM. 952–961. doi: 10.1145/3404835.3462956.

Miao, S.-y., C.-C. Liang, and K.-Y. Su. (2020). “A Diverse Corpus for Evaluating and Developing English Math Word Problem Solvers”. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Online: Association for Computational Linguistics. 975–984. doi: 10.18653/v1/2020.acl-main.92.

Mikolov, T., K. Chen, G. Corrado, and J. Dean. (2013). “Efficient Estimation of Word Representations in Vector Space”. arXiv preprint arXiv:1301.3781.

Miller, B. R. and A. Youssef. (2003). “Technical Aspects of the Digital Library of Mathematical Functions”: 19.

Miller, B. R. (2013). “Three Years of DLMF: Web, Math and Search”. In: Intelligent Computer Mathematics. Berlin, Heidelberg: Springer Berlin Heidelberg. 288–295.

Miner, R. and R. Munavalli. (2007). “An Approach to Mathematical Search Through Query Formulation and Data Normalization”. In: International Conference on Mathematical Knowledge Management. Springer. 342–355.

Mishra, S., M. Finlayson, P. Lu, L. Tang, S. Welleck, C. Baral, T. Rajpurohit, O. Tafjord, A. Sabharwal, P. Clark, and A. Kalyan. (2022). “LILA: A Unified Benchmark for Mathematical Reasoning”. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics. 5807–5832. doi: 10.18653/v1/2022.emnlp-main.392.

Mišutka, J. and L. Galamboš. (2008). “Extending Full Text Search Engine for Mathematical Content”. In: Proc. Digital Mathematical Libraries 2008. 55–67. Birmingham, United Kingdom.

Mitra, A., H. Khanpour, C. Rosset, and A. Awadallah. (2024). “Orca-Math: Unlocking the potential of SLMs in Grade School Math”. url: https://arxiv.org/abs/2402.14830.

Nagy, G. and S. Seth. (1984). “Hierarchical Representation of Optically Scanned Documents”. In: Proc. International Conference on Pattern Recognition. Vol. 1. Montréal, Canada. 347–349.

National Council of Teachers of Mathematics. (2011). “Strategic Use of Technology in Teaching and Learning Mathematics - National Council of Teachers of Mathematics”. url: https://www.nctm.org/Standards-and-Positions/Position-Statements/Strategic-Use-of-Technology-in-Teaching-and-Learning-Mathematics/.

Ng, Y. K., D. J. Fraser, B. Kassaie, and F. W. Tompa. (2021). “Dowsing for Answers to Math Questions: Ongoing Viability of Traditional MathIR”. In: CLEF (Working Notes). 63–81.

Ng, Y. K., D. J. Fraser, B. Kassaie, G. Labahn, M. S. Marzouk, F. W. Tompa, and K. Wang. (2020). “Dowsing for Math Answers with Tangent-L”. CEUR Workshop Proceedings 2696. Ed. by L. Cappellato, C. Eickhoff, N. Ferro, and A. Névéol. url: https://ceur-ws.org/Vol-2696/paper%5C_167.pdf.

Nishizawa, G. (2020). “Visual Structure Editing of Math Formulas”. Master’s thesis. Rochester Institute of Technology.

Nogueira, R., Z. Jiang, and J. Lin. (2021). “Investigating the Limitations of Transformers with Simple Arithmetic Tasks”. url: https://arxiv.org/abs/2102.13019.

Norman, D. A. (1988). The Psychology of Everyday Things. Basic books.

Ohri, A. and T. Schmah. (2021). “Machine Translation of Mathematical Text”. IEEE Access. 9: 38078–38086. doi: 10.1109/ACCESS.2021.3063715.

Okamoto, M. and B. Miao. (1991). “Recognition of Mathematical Expressions by Using the Layout Structures of Symbols”. In: Prof. International Conference on Document Analysis and Recognition. Vol. 1. St. Malo, France. 242–250.

Papineni, K., S. Roukos, T. Ward, and W.-J. Zhu. (2002). “Bleu: a Method for Automatic Evaluation of Machine Translation”. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics. 311–318. doi: 10.3115/1073083.1073135.

Pavan Kumar, P., A. Agarwal, and C. Bhagvati. (2012). “A Structure Based Approach for Mathematical Expression Retrieval”. In: Multi-disciplinary Trends in Artificial Intelligence. Vol. 7694. Berlin, Heidelberg: Springer Berlin Heidelberg. 23–34. doi: 10.1007/978-3-642-35455-7_3.

Peng, S., K. Yuan, L. Gao, and Z. Tang. (2021a). “MathBERT: A Pre-Trained Model for Mathematical Formula Understanding”. arXiv:2105.00377 [cs]. May. url: http://arxiv.org/abs/2105.00377.

Peng, S., K. Yuan, L. Gao, and Z. Tang. (2021b). “Mathbert: A pre-trained model for mathematical formula understanding”. arXiv preprint arXiv:2105.00377.

Perozzi, B., R. Al-Rfou, and S. Skiena. (2014). “Deepwalk: Online Learning of Social Representations”. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. 701–710.

Petersen, F., M. Schubotz, A. Greiner-Petter, and B. Gipp. (2023). “Neural Machine Translation for Mathematical Formulae”. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Toronto, Canada: Association for Computational Linguistics. 11534–11550. doi: 10.18653/v1/2023.acl-long.645.

Pfahler, L. and K. Morik. (2020). “Semantic Search in Millions of Equations”. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Virtual Event CA USA: ACM. 135–143. doi: 10.1145/3394486.3403056.

Piękos, P., M. Malinowski, and H. Michalewski. (2021). “Measuring and Improving BERT’s Mathematical Abilities by Predicting the Order of Reasoning.” In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers). Online: Association for Computational Linguistics. 383–394. doi: 10.18653/v1/2021.acl-short.49.

Pirolli, P. and S. Card. (1999). “Information Foraging”. Psychological Review. 106(4): 643–675. doi: 10.1037/0033-295x.106.4.643.

Rafailov, R., A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn. (2023). "Direct Preference Optimization: Your Language Model is Secretly a Reward Model". In: Advances in Neural Information Processing Systems. Vol. 36. Curran Associates, Inc. 53728–53741. url: https://proceedings.neurips.cc/paper_files/paper/2023/file/a85b405ed65c6477a4fe8302b5e06ce7-Paper-Conference.pdf.

Reichenbach, M. S., A. Agarwal, and R. Zanibbi. (2014). "Rendering Expressions to Improve Accuracy of Relevance Assessment for Math Search". In: Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval. 851–854.

Reusch, A., J. Gonsior, C. Hartmann, and W. Lehner. (2024). "Investigating the Usage of Formulae in Mathematical Answer Retrieval". In: Advances in Information Retrieval. Cham: Springer Nature Switzerland. 247–261.

Reusch, A., M. Thiele, and W. Lehner. (2022). "Transformer-Encoder-Based Mathematical Information Retrieval". In: Experimental IR Meets Multilinguality, Multimodality, and Interaction. Cham: Springer International Publishing. 175–189.

Robertson, S. (2004). "Understanding Inverse Document Frequency: On theoretical arguments for IDF". J. Documentation. 60(5): 503–520. doi: 10.1108/00220410410560582.

Robertson, S. E. and S. Walker. (1994). "Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval". In: Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum). ACM/Springer. 232–241. doi: 10.1007/978-1-4471-2099-5_24.

Robertson, S. E. and H. Zaragoza. (2009). "The Probabilistic Relevance Framework: BM25 and Beyond". Found. Trends Inf. Retr. 3(4): 333–389. doi: 10.1561/1500000019.

Rohatgi, S., W. Zhong, R. Zanibbi, J. Wu, and C. L. Giles. (2019). "Query Auto Completion for Math Formula Search". arXiv: 1912.04115 [cs.IR].

Roy, S. and D. Roth. (2015). “Solving General Arithmetic Word Problems”. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Lisbon, Portugal: Association for Computational Linguistics. 1743–1752. doi: 10.18653/v1/D15-1202.

Ruzicka, M., P. Sojka, and M. Líška. (2016). “Math Indexer and Searcher under the Hood: Fine-tuning Query Expansion and Unification Strategies”. In: Proc. of the 12th NTCIR Conference on Evaluation of Information Access Technologies. Noriko Kando, Tetsuya Sakai, and Mark Sanderson,(Eds.) NII Tokyo. 331–337.

Sakai, T. (2007). “Alternatives to Bpref”. In: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR ’07. Amsterdam, The Netherlands: ACM Press. 71. doi: 10.1145/1277741.1277756.

Satpute, A., N. Gießing, A. Greiner-Petter, M. Schubotz, O. Teschke, A. Aizawa, and B. Gipp. (2024). “Can LLMs Master Math? Investigating Large Language Models on Math Stack Exchange”. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR ’24. Washington DC, USA: Association for Computing Machinery. 2316–2320. doi: 10.1145/3626772.3657945.

Schellenberg, T., B. Yuan, and R. Zanibbi. (2012). “Layout-Based Substitution Tree Indexing and Retrieval for Mathematical Expressions”. In: Document Recognition and Retrieval XIX, part of the IS&T-SPIE Electronic Imaging Symposium, Burlingame, California, USA, January 25-26, 2012, Proceedings. Vol. 8297. SPIE Proceedings. SPIE. 82970I. doi: 10.1117/12.912502.

Schubotz, M., A. Grigorev, M. Leich, H. S. Cohl, N. Meuschke, B. Gipp, A. S. Youssef, and V. Markl. (2016). “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. SIGIR ’16. Pisa, Italy: Association for Computing Machinery. 135–144. doi: 10.1145/2911451.2911503.

Schubotz, M., P. Scharpf, K. Dudhat, Y. Nagar, F. Hamborg, and B. Gipp. (2018). "Introducing MathQA: a Math-Aware Question Answering System". Information Discovery and Delivery. 46(4): 214–224. doi: 10.1108/IDD-06-2018-0022.

Shah, A. K., A. Dey, and R. Zanibbi. (2021). "A Math Formula Extraction and Evaluation Framework for PDF Documents". In: 16th International Conference on Document Analysis and Recognition, ICDAR 2021, Lausanne, Switzerland, September 5-10, 2021, Proceedings, Part II. Vol. 12822. Lecture Notes in Computer Science. Springer. 19–34. doi: 10.1007/978-3-030-86331-9_2.

Shen, Y. and C. Jin. (2020). "Solving Math Word Problems with Multi-Encoders and Multi-Decoders". In: Proceedings of the 28th International Conference on Computational Linguistics. Barcelona, Spain (Online): International Committee on Computational Linguistics. 2924–2934. doi: 10.18653/v1/2020.coling-main.262.

Shirahama, K. and M. Grzegorzek. (2016). "Towards Large-Scale Multimedia Retrieval Enriched by Knowledge about Human Interpretation". Multim. Tools Appl. 75(1): 297–331. doi: 10.1007/S11042-014-2292-8.

Shumailov, I., Z. Shumaylov, Y. Zhao, Y. Gal, N. Papernot, and R. Anderson. (2024a). "The Curse of Recursion: Training on Generated Data Makes Models Forget". url: https://arxiv.org/abs/2305.17493.

Shumailov, I., Z. Shumaylov, Y. Zhao, N. Papernot, R. Anderson, and Y. Gal. (2024b). "AI Models Collapse When Trained on Recursively Generated Data". Nature. July. doi: 10.1038/s41586-024-07566-y.

Sojka, P. and M. Líška. (2011). "Indexing and Searching Mathematics in Digital Libraries". In: Intelligent Computer Mathematics. Vol. 6824. Berlin, Heidelberg: Springer Berlin Heidelberg. 228–243. doi: 10.1007/978-3-642-22673-1_16.

Sojka, P., M. Růžička, and V. Novotný. (2018). "MIaS: Math-Aware Retrieval in Digital Mathematical Libraries". In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. Torino Italy: ACM. 1923–1926. doi: 10.1145/3269206.3269233.

Song, Y. and X. Chen. (2021). “Searching for Mathematical Formulas Based on Graph Representation Learning”. In: Intelligent Computer Mathematics. Vol. 12833. Cham: Springer International Publishing. 137–152. doi: 10.1007/978-3-030-81097-9_11.

Stathopoulos, Y., S. Baker, M. Rei, and S. Teufel. (2018). “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, Volume 1 (Long Papers). New Orleans, Louisiana: Association for Computational Linguistics. 303–312. doi: 10.18653/v1/N18-1028.

Stathopoulos, Y. and S. Teufel. (2015). “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 (Volume 2: Short Papers). 334–340.

Sudholt, S. and G. A. Fink. (2016). “PHOCNet: A Deep Convolutional Neural Network for Word Spotting in Handwritten Documents”. In: 2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR). IEEE Computer Society. 277–282.

Tan, M., L. Wang, L. Jiang, and J. Jiang. (2022). “Investigating Math Word Problems using Pretrained Multilingual Language Models”. In: Proceedings of the 1st Workshop on Mathematical Natural Language Processing (MathNLP). Abu Dhabi, United Arab Emirates (Hybrid): Association for Computational Linguistics. 7–16. doi: 10.18653/v1/2022.mathnlp-1.2.

Tausczik, Y. R., A. Kittur, and R. E. Kraut. (2014). “Collaborative problem solving: a study of MathOverflow”. In: Computer Supported Cooperative Work, CSCW ’14, Baltimore, MD, USA, February 15-19, 2014. ACM. 355–367. doi: 10.1145/2531602.2531690.

Thanda, A., A. Agarwal, K. Singla, A. Prakash, and A. Gupta. (2016). “A Document Retrieval System for Math Queries”. In: Proc. NTCIR-12. 346–353. Tokyo, Japan.

Truong, T.-N., C. Nguyen, R. Zanibbi, H. Mouchére, and M. Nakagawa. (2024). "A Survey on Handwritten Mathematical Expression Recognition: The Rise of Encoder-Decoder and GNN Models". Pattern Recognition: (to appear).

Tucker, Q. (2024). "Improving Histogram-Based Mathematical Formula Search with Local Matching and Learned Embeddings". Tech. rep. Rochester Institute of Technology. url: https://www.cs.rit.edu/~rlaz/projects_theses/Tucker_Local_PHOC_Project_May2024.pdf.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. (2017). "Attention Is All You Need". CoRR. abs/1706.03762. url: http://arxiv.org/abs/1706.03762.

Wallace, E., Y. Wang, S. Li, S. Singh, and M. Gardner. (2019). "Do NLP Models Know Numbers? Probing Numeracy in Embeddings". In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics. 5307–5315. doi: 10.18653/v1/D19-1534.

Wang, K., J. Pan, W. Shi, Z. Lu, M. Zhan, and H. Li. (2024). "Measuring Multimodal Mathematical Reasoning with MATH-Vision Dataset". arXiv: 2402.14804 [cs.CV]. url: https://arxiv.org/abs/2402.14804.

Wang, M., Y. Tang, J. Wang, and J. Deng. (2017a). "Premise Selection for Theorem Proving by Deep Graph Embedding". In: Advances in Neural Information Processing Systems. Vol. 30. Curran Associates, Inc. url: https://proceedings.neurips.cc/paper_files/paper/2017/file/18d10dc6e666eab6de9215ae5b3d54df-Paper.pdf.

Wang, Q., C. Kaliszyk, and J. Urban. (2018). "First Experiments with Neural Translation of Informal to Formal Mathematics". In: Intelligent Computer Mathematics. Cham: Springer International Publishing. 255–270.

Wang, X., C. Macdonald, N. Tonellotto, and I. Ounis. (2023). "Reproducibility, Replicability, and Insights into Dense Multi-Representation Retrieval Models: from ColBERT to Col". In: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023. ACM. 2552–2561. doi: 10.1145/3539618.3591916.

Wang, Y., X. Liu, and S. Shi. (2017b). "Deep Neural Solver for Math Word Problems". In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. Copenhagen, Denmark: Association for Computational Linguistics. 845–854. doi: 10.18653/v1/D17-1088.

Wang, Z., M. Zhang, R. G. Baraniuk, and A. S. Lan. (2021). "Scientific Formula Retrieval via Tree Embeddings". In: 2021 IEEE International Conference on Big Data (Big Data). 1493–1503. doi: 10.1109/BigData52589.2021.9671942.

Wangari, K. D. V., R. Zanibbi, and A. Agarwal. (2014). "Discovering real-world use cases for a multimodal math search interface". In: Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval. ACM. doi: 10.1145/2600428.2609481.

Wei, J., X. Wang, D. Schuurmans, M. Bosma, b. ichter brian, F. Xia, E. Chi, Q. V. Le, and D. Zhou. (2022). "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models". In: Advances in Neural Information Processing Systems. Vol. 35. Curran Associates, Inc. 24824–24837. url: https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

Welleck, S., J. Liu, R. L. Bras, H. Hajishirzi, Y. Choi, and K. Cho. (2021). "NaturalProofs: Mathematical Theorem Proving in Natural Language". In: Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1). url: https://openreview.net/forum?id=Jvxa8adr3iY.

Wenzel, M., L. C. Paulson, and T. Nipkow. (2008). "The Isabelle Framework". In: Theorem Proving in Higher Order Logics. Berlin, Heidelberg: Springer Berlin Heidelberg. 33–38.

White, R. W. (2016). Interactions with Search Systems. Cambridge University Press.

Wu, Y., A. Q. Jiang, W. Li, M. Rabe, C. Staats, M. Jamnik, and C. Szegedy. (2022). "Autoformalization with Large Language Models". In: Advances in Neural Information Processing Systems. Vol. 35. Curran Associates, Inc. 32353–32368. url: https://proceedings.neurips.cc/paper_files/paper/2022/file/d0c6bc641a56bebee9d985b937307367-Paper-Conference.pdf.

Xue, L., A. Barua, N. Constant, R. Al-Rfou, S. Narang, M. Kale, A. Roberts, and C. Raffel. (2022). "ByT5: Towards a Token-Free Future with Pre-trained Byte-to-Byte Models". Transactions of the Association for Computational Linguistics. 10: 291–306. doi: 10.1162/tacl_a_00461.

Yang, K., A. Swope, A. Gu, R. Chalamala, P. Song, S. Yu, S. Godil, R. J. Prenger, and A. Anandkumar. (2023). "LeanDojo: Theorem Proving with Retrieval-Augmented Language Models". In: Advances in Neural Information Processing Systems. Vol. 36. Curran Associates, Inc. 21573–21612. url: https://proceedings.neurips.cc/paper_files/paper/2023/file/4441469427094f8873d0fecb0c4e1cee-Paper-Datasets_and_Benchmarks.pdf.

Yasunaga, M. and J. D. Lafferty. (2019). "TopicEq: A Joint Topic and Mathematical Equation Model for Scientific Texts". Proceedings of the AAAI Conference on Artificial Intelligence. 33(July): 7394–7401. doi: 10.1609/aaai.v33i01.33017394.

Yu, L., W. Jiang, H. Shi, J. Yu, Z. Liu, Y. Zhang, J. T. Kwok, Z. Li, A. Weller, and W. Liu. (2024). "MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models". url: https://arxiv.org/abs/2309.12284.

Yu, M., G. Li, D. Deng, and J. Feng. (2016). "String Similarity Search and Join: A Survey". Frontiers Comput. Sci. 10(3): 399–417. doi: 10.1007/S11704-015-5900-5.

Yuan, K., D. He, Z. Jiang, L. Gao, Z. Tang, and C. L. Giles. (2020). "Automatic Generation of Headlines for Online Math Questions". Proceedings of the AAAI Conference on Artificial Intelligence. 34(05): 9490–9497. doi: 10.1609/aaai.v34i05.6493.

Yue, X., X. Qu, G. Zhang, Y. Fu, W. Huang, H. Sun, Y. Su, and W. Chen. (2023). “MAmmoTH: Building Math Generalist Models through Hybrid Instruction Tuning”. url: https://arxiv.org/abs/2309.05653.

Zanibbi, R., A. Aizawa, and M. Kohlhase. (2016a). “NTCIR-12 MathIR Task Overview”. In: Proc. NTCIR-12. 299–308.

Zanibbi, R. and D. Blostein. (2012). “Recognition and Retrieval of Mathematical Expressions”. International Journal on Document Analysis and Recognition (IJDAR). 15(4): 331–357. doi: 10.1007/s10032-011-0174-4.

Zanibbi, R., K. Davila, A. Kane, and F. W. Tompa. (2016b). “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. Pisa Italy: ACM. 145–154. doi: 10.1145/2911451.2911512.

Zanibbi, R. and L. Yu. (2011). “Math Spotting: Retrieving Math in Technical Documents Using Handwritten Query Images”. In: 2011 International Conference on Document Analysis and Recognition, ICDAR 2011, Beijing, China, September 18-21, 2011. IEEE Computer Society. 446–451. doi: 10.1109/ICDAR.2011.96.

Zeng, H., C. Luo, B. Jin, S. M. Sarwar, T. Wei, and H. Zamani. (2024). “Scalable and Effective Generative Information Retrieval”. In: Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024. ACM. 1441–1452. doi: 10.1145/3589334.3645477.

Zhang, A., A. Goyal, W. Kong, H. Deng, A. Dong, Y. Chang, C. A. Gunter, and J. Han. (2015). “adaQAC: Adaptive Query Auto-Completion via Implicit Negative Feedback”. In: Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval. 143–152.

Zhang, D., L. Wang, L. Zhang, B. T. Dai, and H. T. Shen. (2020a). “The Gap of Semantic Parsing: A Survey on Automatic Math Word Problem Solvers”. IEEE Trans. Pattern Anal. Mach. Intell. 42(9): 2287–2305. doi: 10.1109/TPAMI.2019.2914054.

Zhang, M., S. Baral, N. Heffernan, and A. Lan. (2022). “Automatic Short Math Answer Grading via In-context Meta-learning”. In: Proceedings from the 15th International Conference on Educational Data Mining.

Zhang, M., Z. Cui, M. Neumann, and Y. Chen. (2018). “An End-to-End Deep Learning Architecture for Graph Classification”. Proceedings of the AAAI Conference on Artificial Intelligence. 32(1). doi: 10.1609/aaai.v32i1.11782.

Zhang, T., V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. (2020b). “BERTScore: Evaluating Text Generation with BERT”. url: https://arxiv.org/abs/1904.09675.

Zhao, J., M.-Y. Kan, and Y. L. Theng. (2008). “Math Information Retrieval: User Requirements and Prototype Implementation”. In: Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries - JCDL ’08. Pittsburgh PA, PA, USA: ACM Press. 187. doi: 10.1145/1378889.1378921.

Zhong, L., J. Wu, Q. Li, H. Peng, and X. Wu. (2024). “A Comprehensive Survey on Automatic Knowledge Graph Construction”. ACM Comput. Surv. 56(4): 94:1–94:62. doi: 10.1145/3618295.

Zhong, W., S.-C. Lin, J.-H. Yang, and J. Lin. (2023). “One Blade for One Purpose: Advancing Math Information Retrieval using Hybrid Search”. In: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR ’23. Taipei, Taiwan: Association for Computing Machinery. 141–151. doi: 10.1145/3539618.3591746.

Zhong, W. and R. Zanibbi. (2019). “Structural Similarity Search for Formulas Using Leaf-Root Paths in Operator Subtrees”. In: Advances in Information Retrieval. Vol. 11437. Cham: Springer International Publishing. 116–129. doi: 10.1007/978-3-030-15712-8_8.

Zhong, W., X. Zhang, J. Xin, R. Zanibbi, and J. Lin. (2021). “Approach Zero and Anserini at the CLEF-2021 ARQMath Track: Applying Substructure Search and BM25 on Operator Tree Path Tokens”. Proc. CLEF 2021 (CEUR Working Notes).

Zhu, L., C. Zheng, W. Guan, J. Li, Y. Yang, and H. T. Shen. (2024). "Multi-Modal Hashing for Efficient Multimedia Retrieval: A Survey". IEEE Trans. Knowl. Data Eng. 36(1): 239–260. doi: 10.1109/TKDE.2023.3282921.

Zobel, J. and A. Moffat. (2006). "Inverted Files for Text Search Engines". ACM Comput. Surv. 38(2): 6–es. doi: 10.1145/1132956.1132959.