Effective Math-Aware Ad-Hoc Retrieval based on Structure Search and Semantic Similarities

by

Wei Zhong

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2023

Examining Committee Membership

The following served on the Examining Committee for this thesis.

Supervisor(s):	**Jimmy Lin** Professor David R. Cheriton School of CS, University of Waterloo
Internal Member:	**Frank Wm. Tompa** Distinguished Professor Emeritus David R. Cheriton School of CS, University of Waterloo
Internal Member:	**Charles Clarke** Professor David R. Cheriton School of CS, University of Waterloo
Internal Member:	**James Danckert** Professor and Cognitive Neuroscience Research Area Head Department of Psychology, University of Waterloo
External Member:	**Wolfgang Lehner** Professor and Chair of the Database Technology Group Technische Universität Dresden (TU Dresden)

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

The thesis includes contributions from three published and peer-reviewed papers. The thesis author Wei Zhong is the sole first author of all three articles, and is responsible for the most parts of idea, implementation, experimentation, and writing.

Other authors include:

- Chapter 3 (Zhong and Zanibbi [2019]): Richard Zanibbi;
- Chapter 4 (Zhong et al. [2020]): Shaurya Rohatgi, Jian Wu, C. Lee Giles, and Richard Zanibbi;
- Chapter 5 (Zhong et al. [2023]): Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin.

They are mainly responsible for sending feedback, discussing experimental details, and proof reading drafts.

Abstract

Despite the prevalence of digital scientific and educational contents on the Internet, only a few search engines are capable to retrieve them efficiently and effectively. The main challenge in freely searching scientific literature arises from the presence of structured math formulas and their heterogeneous and contextually important surrounding words. This thesis introduces an effective math-aware, ad-hoc retrieval model that incorporates structure search and semantic similarities. Transformer-based neural retrievers have been adopted to capture additional semantics using domain-adapted supervised retrieval.

To enable structure search, I suggest an unsupervised retrieval model that can filter potential mathematical formulas based on structure similarity. This similarity is determined by measuring the largest common substructure(s) in a formula tree representation, known as the Operator Tree (OPT). The structure matching is approximated by employing maximum matching of path-based structure features. The proposed structure similarity measurement can be tailored based on the desired effectiveness and efficiency trade-offs. It may consider various node types, such as operators and operands, and accommodate different numbers of common subtrees with varying weights. In addition to structure similarity, this unsupervised model also captures symbol substitutions through a greedy matching algorithm applied to the matched substructure(s).

To achieve efficient structure search, I introduce a dynamic pruning algorithm to the problem of structure retrieval. The proposed retrieval algorithm efficiently identifies the maximum common subtree among formula candidates and safely eliminates potential structure matches that exceed a dynamic threshold. To accomplish this, three rank-safe pruning strategies are suggested and compared against exhaustive search baselines. Additionally, more aggressive thresholding policies are proposed to balance effectiveness with further speed improvements. A novel hierarchical inverted index has been implemented. This index is designed to be compatible with traditional information retrieval (IR) infrastructure and optimization techniques.

To capture other semantic similarities, I have incorporated neural retrievers into a hybrid setting with structure search. This approach has achieved the state-of-the-art effectiveness in recent math information retrieval tasks. In comparison to strict and unsupervised matching, I have found that supervised neural retrievers are able to capture additional semantic similarities in a highly complementary manner. In order to learn effective representations in heterogeneous math contents, I have proposed a novel pretraining architecture that can improve the contextual awareness between math and its surrounding texts. This pretraining scheme generates effective downstream single-vector representations, eliminating the efficiency bottleneck from using multi-vector dense representations.

In the end, the thesis examines future directions, specifically the integration of recent advancements in language modeling. This includes incorporating ongoing exciting developments of large language models for improved math information retrieval. A preliminary evaluation has been conducted to assess the impact of these advancements.

Acknowledgements

I have had the great honor of being supervised by both Prof. Jimmy Lin and Prof. Richard Zanibbi during my Ph.D. program. While COVID-19 interrupted my Ph.D. journey midway, I consider myself extremely fortunate to have received the best advice and guidance possible from these two eminent figures in my field of study. I deeply appreciate their generous support in fostering the growth and success of students. Simultaneously, I am grateful for the freedom I had to explore the thesis topic of my choice.

I extend my heartfelt thanks to my thesis committee for their time, valuable comments, and insightful suggestions.

To my family; they are the unsung heroes who have made my academic journey possible. Their support provided me the peace of mind and perseverance needed to pursue my degree. To my wife, Karmen, I offer my gratitude for her encouragement, sacrifice, and companionship during my extended period of study abroad.

Working on this thesis topic has been an enlightening experience, and I am also grateful to another Waterloo group, which includes Dr. Tompa, Kiki Ng, Dr. Andrew Kane, and others, for their interests, feedback, and constructive critiques. Dr. Tompa, I appreciate your meticulous and insightful comments on my thesis. Also, I would like to express my gratitude to Professors Lee Giles and Douglas Oard, Dr. Jian Wu, Shaurya Rohatgi, Gavin Nishizawa, Mahshad, Puneeth, Parag, and others with whom I met and collaborated on the MathSeer project.

I am thankful to Dr. Pengcheng Shi, Prof. Linwei Wang, and others for the knowledge and support I received from RIT. And I want to mention Prof. Hui Fang from the University of Delaware, who introduced me to the field of information retrieval and encouraged me to work on this topic during a course project, marking the start of my wonderful academic journey.

I extend my thanks to everyone who contributed to the good memories and friendships I made in the DSG lab, DPRL, and other places during my graduate life. Among the many acts of kindness I received: Thank you, Yuqing Xie, for staying up together for the ARQMath submission and your various kinds of help. Thank you, Xinyi Fan, for assisting me in preparing for my interviews. Thank you, Mingxun Zhen, for hosting me and driving me to an early morning flight. Thank you, Yuqi Liu, for actively sending me suggestions for my new LinkedIn pages. Thank you, Xin Ji, Shi Peng, and Zhiying Jiang, for kindly mentioning me in your theses. I am also thankful to my landlord, Xiaoge Ye, for his warm-hearted kindness and offering me a low rent price in Waterloo.

Dedication

Dedicated to my parents, Zhong Jie and Hua Jing, as well as my wife, Karmen, and my entire family.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The significant progress of humanity in recent centuries can be largely attributed to the flourishing development of science and technology. Mathematical language plays a vital role in expressing scientific discoveries, serving as the primary and concise means of conveying information in the realm of scientific exploration. As a result, the capability of computer systems to comprehend mathematical language and retrieve information from these valuable human assets is of considerable interest and importance.

While we are currently experiencing an exciting era of rapid advancements in deep learning, the computer's understanding of mathematical language remains limited. Compared to natural language understanding, the understanding and reasoning of math language [Lu et al., 2022] from some of the most effective large language models is in fact lagging behind the *scaling law* [Kaplan et al., 2020]. As concluded Rae et al. [2021],

> "We find that scale has a reduced benefit for tasks in the Maths, Logical Reasoning, and Common Sense categories. Our results suggest that for certain flavors of mathematical or logical reasoning tasks, it is unlikely that scale alone will lead to performance breakthroughs."

Despite the limitations of advanced deep learning models in effectively handling math content, Information Retrieval (IR) can serve as a viable middle ground between fully comprehending mathematics and completely disregarding the presence of math language in documents. Information retrieval techniques, particularly through similarity search, allow for the highly efficient retrieval of existing documented math knowledge without the requirement for an in-depth understanding of the content.

In addition, utilizing IR systems for performing first-stage retrieval serves as an efficient recall mechanism for effective reranking [Nogueira and Cho, 2019, Nogueira et al., 2019b] and can also be employed as a Question-Answering (Q&A) reader in knowledge-intensive tasks [Izacard and Grave, 2020]. With the assistance of IR, we can also generate a relevant collection of factual and informative documents that can serve as valuable references or evidences for other systems to utilize and consume. This is crucial for answering math questions more effectively, as modern neural networks have shown greater power when they can avoid hallucination and factual inconsistency by consulting external tools [Ji et al., 2022, Schick et al., 2023, Shi et al., 2023].

1.1 Challenges

Unfortunately, conventional IR systems nowadays still lack many capabilities when it comes to searching math language, and this limitation arises due to various challenges.

1.1.1 Effectiveness Challenges

Mathematical languages, typically presented in structured markup languages like LaTeX, possess inherent properties that make them challenging for existing IR models to address. These properties include:

1. Math language often exhibits nested or structured patterns, and it is prone to permutations within substructures, such as under commutative operators. For an effective retrieval model, it becomes essential to capture the structure of mathematical expressions rather than treating them as individual tokens, as traditional retrieval models do. For instance, it is important to recognize that formulas $\frac{a}{a^2+b}$ and $\frac{a^2}{a+b}$ are different despite having identical tokens. Simultaneously, the model should also be able to identify that $\frac{a}{a^2+b}$ and $\frac{a}{b+a^2}$ are semantically equivalent, even if there is a permuted substructure. Importantly, indexing every subexpression and its permutations is not a practical solution.

2. In math formulas, most symbols can be freely substituted with another set of symbols without changing their semantic meaning. However, traditional search engines primarily relying on exact lexical matching are unable to capture such variations.

3. Lastly, the semantics of math content is highly abstract, contextual, and occasionally obscure. Achieving efficient and effective retrieval of such content may require a context-sensitive and powerful representation capable of capturing a wide range of variations and nuances in math topics. Additionally, the heterogeneous nature of math content presents another challenge in developing a generalized representation.

None of these challenges have been adequately addressed in traditional IR settings. In fact, traditional retrieval models could be 3 to 5 times less effective, compared to specialized math retrieval systems [Mansouri et al., 2022, Zhong et al., 2022b]. This gap is even larger when considering formula-centered retrieval [Mansouri et al., 2020a].

1.1.2 Efficiency and Complexity Issues

It is important to emphasize the efficiency aspect of information retrieval, particularly when dealing with real-world document collections comprising millions of entries. While effective but computationally expensive methods exist, their widespread adoption is hindered by impractical query run times. Considering the fundamental concern of balancing effectiveness and efficiency in the IR domain, I believe that this trade-off becomes even more crucial in the context of math retrieval. This is due to the existing systems for math retrieval facing challenges of either being ineffective or suffering from excessive slowness. These issues can be attributed to the following factors:

1. Inherently, the strict structure search problem is too complex to be considered by a real-time search engine on any practical real-world corpus. Even if we consider the math similarity search problem to be as simple as only matching substructures in tree representations, it is known that the problems of determining the largest common sub-tree and computing the edit distance between unordered trees are NP-hard [Zhang and Jiang, 1994]. To effectively retrieve math formulas, a search engine has to make hard trade-offs between efficiency and effectiveness.

2. Furthermore, incorporating an additional modality, such as math, into the retrieval process often necessitates the development of distinct pipelines or software stacks. The associated overheads and complexities can overshadow the commercial incentives for integrating math search functionality into existing search infrastructures. Without a cost-effective math IR model in place, the benefits of handling math content may struggle to outweigh the complexity and engineering overhead it entails.

Consequently, there is a significant interest in designing a math-aware retrieval model that tackles the aforementioned effectiveness challenges in a cost-effective manner.

1.2 Math-Aware Retrieval and Formula Search

Math-aware retrieval typically pertains to information retrieval involving mathematical formulas within the query. To assess the capability of formula retrieval more effectively, datasets or benchmarks often establish *formula search* tasks (as described in Section 2.4) where either an isolated formula or a particular formula extracted from a question is used to query a search engine. In this thesis, we will adhere to the above-mentioned definitions when describing a task and its corresponding query topics.

1.2.1 An Example

The field of Math Information Retrieval (MIR) [Zhao et al., 2008, Zanibbi and Blostein, 2012] is a relatively new sub-domain in the field of Information Retrieval (IR). One of the key tasks in MIR is to effectively and efficiently retrieve math language within scientific documents. In the context of retrieval, it refers to the ability to include mathematical expressions in a search query. This allows the search engine to assist in finding similar expressions and retrieve relevant documents or topics. In essence, it combines the functionality of a typical search engine with math-specific search capabilities.

Unlike traditional search engines, a math-aware search engine permits users to incorporate math expressions as keywords to aid in the identification of similar expressions and the retrieval of pertinent documents. A math-aware *ad-hoc* retrieval system goes beyond filtering the results, it incorporates a ranking mechanism based on user preferences (see Section 2.1 for a historical view of ad-hoc retrieval).

To provide a concrete example, let's consider a real question posted on Math StackExchange,[1] one of the most popular math Q&A websites:

> "While I'm able to prove that the series $u_n = \frac{\sqrt{(n-1)!}}{(1+\sqrt{1})...(1+\sqrt{n})}$ converges, I don't see the trick to compute the value of its sum starting at $n = 2$. Any clue on the way to compute the sum?"

To help users answer such questions, employing math-aware ad-hoc retrieval to find existing solutions, particularly in cases heavily reliant on a specific formula, can be an effective approach. A user has the option to use the math formula itself, along with optional text keywords, as the query to search for relevant documents. The search engine

[1] https://math.stackexchange.com/questions/1433826

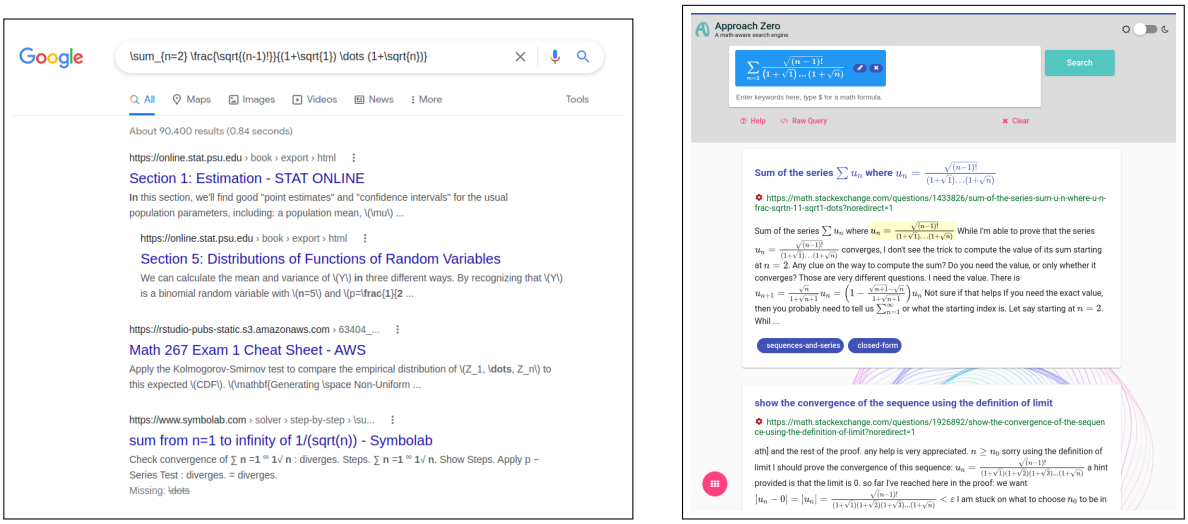


Figure 1.1: The search results of math formula query $\sum_{n=2} \frac{\sqrt{(n-1)!}}{(1+\sqrt{1})\dots(1+\sqrt{n})}$ returned from conventional and math-aware search engines. Left: Google search results. Right: Search results from https://approach0.xyz. (Screenshot taken on Jan. 27, 2023)

takes into account both the formula and the keywords to retrieve ranked documents that are most likely to contain the desired information.

Figure 1.1 illustrates a comparison between the user interface and search results of a commonly used commercial search engine and a math-aware ad-hoc retrieval demo that I have developed as the main contributor. For the former system, the input query and the returned documents in LaTeX format are not rendered or displayed nicely. Moreover, when compared to the math-aware search engine illustrated on the right side of the figure, even top-regarded search engines like Google may struggle to provide relevant results for math-related queries. On the other hand, to cater to math retrieval, a well-designed user interface and a potential hand-written recognition tool can play a crucial role in providing a positive user experience in a search engine [Labahn et al., 2008, Nishizawa et al., 2020]. Additionally, the query may include text keywords within the context of a question to meet the user's requirements [Mansouri et al., 2019b]. All of these specific needs within a math-aware ad-hoc retrieval system necessitate a distinct search scheme to ensure its usefulness and effectiveness.

1.3 Contributions and Outline

Here are some of the key contributions outlined by the chapters:

1. **Chapter 2** presents background knowledge on general IR and a survey of the MIR domain. It explores the historical systems in MIR and examines their relations to our proposed systems. The chapter establishes the context for our proposed system and provides a historical overview of the structure search in the landscape of MIR.

2. **Chapter 3** introduces a novel structure search model designed to effectively retrieve math formulas by matching their substructures in the Operator Tree (OPT) representation. The model adopts a search unit based on the prefixes of OPT leaf-root paths, ensuring that the structure matching remains invariant to commutative operators and symbol substitutions. It enables the model to differentiate between formulas like $E = mc^2$ and $y = ax^2$, which have similar structures but distinct symbols. By leveraging common substructure search and scoring, the proposed structure search has established a strong indication of formula relevance and achieved the state-of-the-art effectiveness among unsupervised MIR systems.

3. **Chapter 4** presents a novel dynamic pruning algorithm designed to accelerate the structure search model, resulting in significantly faster query execution without compromising effectiveness. The algorithm utilizes query structure properties to estimate upperbounds for structure matching, marking the initial incorporation of structure matching into inverted index optimization. The chapter introduces three different rank-safe pruning strategies, among which the GBP-LEN strategy emerges as the most efficient overall by modeling top-k retrieval as a binary linear programming problem. In addition, this chapter explores a more aggressive (unsafe) pruning method that further advances efficiency through the use of initial thresholds.

4. **Chapter 5** introduces the DPR and MAEs neural retrievers to the field of MIR. These models represent a significant advancement in leveraging deep learning for math retrieval tasks. In this chapter, a novel pretraining scheme is proposed, which takes advantage of the rich formula-text context information in math documents. Through extensive evaluation, it is discovered that deep neural retrievers complement structure search models by effectively reducing the false positive rate in top search results. Building upon this insight, two highly effective hybrid search combinations are proposed under the name *MABOWDOR*. These hybrid search models integrate efficient search components to achieve highly effective retrieval results. Notably, the MABOWDOR-base hybrid search model stands out by maintaining a state-of-the-art

level of effectiveness while employing a minimal number of hybrid components, with sub-second average query run times against a million-level corpus in a single CPU environment.

5. **Chapter 6** concludes the thesis and provides an outlook on the future of MIR. It highlights the challenges that remain to be addressed in this field and discusses the potential impact of using large language models to rerank high-quality first-stage retrieval results.

Chapter 2

Background

2.1 Traditional Ad-Hoc Retrieval

The field of Information Retrieval (IR) commences with the exploration of unstructured *documents* in order to meet an information need [Schütze et al., 2008]. In the early stages, the main search options for vast amounts of data were predominantly boolean retrieval, which only offers filtered results without any ranking. However, such systems generally exhibit divergent extremes in the trade-off between precision and recall [Lee and Fox, 1988]. With the emergence of the WEB era in the early 1990s and the subsequent exponential expansion of data collection, the field of information retrieval has undergone substantial evolution. Rather than filtering documents based on specific conditions, information retrieval now utilizes scoring functions and various optimizations to effectively and efficiently address user requirements by locating pertinent documents. This is accomplished by generating a prioritized list that ranks documents based on their similarities to the query, a technique commonly employed in popular search engines today. This methodology is commonly known as *ad-hoc retrieval*.

2.1.1 Ranking

In traditional ad-hoc retrieval, the evaluation of similarity often involves treating a document and a query as *bags of words*. This approach entails assessing the similarity between two texts by comparing their vocabulary vectors, disregarding the original word order within the document. It is worth noting that in proximity search [Tao and Zhai, 2007],

where the closest matching keywords in the documents are utilized to measure proximity similarity, there is a partial consideration of word sequence.

This search paradigm facilitates efficient query processing through the utilization of the *inverted index*. The inverted index is constructed using *inverted lists* or *posting lists*, which often include the positions of terms within the documents. This approach has been proven effective and well-suited for retrieving a relevant document d based on a given query q, employing the *tf–idf* scoring schemes [Jones, 1972, Zobel and Moffat, 1998]:

$$S(q,d) = \sum_{t \in q,d} tf_{t,d} \cdot idf_t \tag{2.1}$$

where $tf_{t,d}$ is the term frequency of a term t occurring in document d, and idf_t is the inverse document frequency which usually is inversely proportional to the log of the global frequency of t.

This formulation of similarity can be viewed as a dot product in the query-document term vector space, known as the *vector space model*. Alternatively, it can be interpreted as the mutual information between a document and a query, where the idf factor represents the conditional information entropy of the document given the query [Aizawa, 2003].

One of the most popular tf–idf scoring variance is derived from the Okapi-BM25 [Robertson et al., 1995, Kamphuis et al., 2020], formally,

$$\sum_{t \in q} \log\left(1 + \frac{N - df_t + 0.5}{df_t + 0.5}\right) \cdot \frac{tf_{t,d}}{tf_{t,d} + k_1(1 - b + b(L_d/L_{avg}))} \tag{2.2}$$

where k_1 and b are hyperparameters, and N, df_t, $tf_{t,d}$, L_d, L_{avg} refer to the total number of documents, the document frequency of the term t, the term frequency of term t in the document d, the length of document d, and the average document length respectively. While the tf factor has designed heuristically from combining empirical BM11 and BM15 functions [Robertson et al., 1995], the weight from the idf factor has some theory interpretations [Croft and Harper, 1979, Aizawa, 2003, Robertson, 2004, Lee, 2007, Croft et al., 2010], e.g., it can be viewed as an approximated likelihood ratio of a seeing a document given its relevance, $P(D \mid \text{relevance})/P(D \mid \text{irrelevance})$.

2.1.2 Efficiency

One of the key challenges encountered by search engines is to ensure efficiency, particularly in terms of query latencies. This is due to the fact that even a single-thread retrieval

system is commonly required to handle a substantial volume of documents, varying from hundreds of thousands to millions. In order to improve space, inverted lists within the inverted index are frequently heavily compressed. In fact, commercial search engines may choose to cache the entire index in memory. However, compression techniques employed in IR must strike a balance between achieving a high compression ratio for integers (e.g., document IDs) and enabling fast decompression during decoding [Mallia et al., 2019].

In addition to compression, another widely used and frequently combined optimization technique is *dynamic pruning*. Dynamic pruning methods involve utilizing an estimated upperbound of the score for each candidate during retrieval. This allows for the pruning of a significant number of candidate documents that have a score lower than the lowest score among a dynamically maintained set of top-k documents. Dynamic pruning is also applied to skip reading index items or avoid computing full similarity scores whenever possible. Pruning methods can be based on various query processing schemes [Shan et al., 2012]. The Document-at-a-time (DAAT) scheme necessitates merging all relevant posting lists simultaneously. On the other hand, the Term-at-a-time (TAAT) or Score-at-a-time (SAAT) schemes process one posting list at a time for each term where additional memory (referred to as *accumulators*) is required to store partial scores, and the posting lists are usually sorted by document importance, with potentially promising documents positioned at the front of the inverted lists. For instance, the impact score [Anh and Moffat, 2006] can be used to determine the ordering of documents in the posting lists. Pruning strategies are *rank-safe* (or *safe up to rank k*) if they guarantee that the final top-k documents are ranked in the same order before and after pruning. The best approach at this time for SAAT is JASS [Trotman and Crane, 2019], and the most well-known rank-safe pruning strategies for DAAT are MaxScore [Turtle and Flood, 1995, Strohman et al., 2005, Jonassen and Bratsberg, 2011] and WAND variants [Broder et al., 2003, Ding and Suel, 2011]. Shan et al. [Shan et al., 2012] demonstrated that MaxScore variants, such as BMM (Block MaxScore) and LBMM (Late Block MaxScore), outperform other dynamic pruning strategies when it comes to long queries. More recently, Mallia et al. [Antonio Mallia and Suel, 2019] reported similar findings across various popular index encodings, further supporting the superiority of MaxScore variants in long queries.

In MaxScore dynamic pruning (illustrated in Figure 2.1), the top-k scored candidate hits are kept throughout the querying process dynamically and the lowest score in top-k candidates is defined as *threshold* θ. Since at most k candidates will be returned as search results, dynamic pruning strategies work by estimating a score upperbound early before knowing the precise score of a hit. If it is less or equal to θ, the associated document can be pruned safely because it can not appear in the final results. Moreover, if a subset of hit posting lists alone cannot produce a top-k result from their upperbounds, they

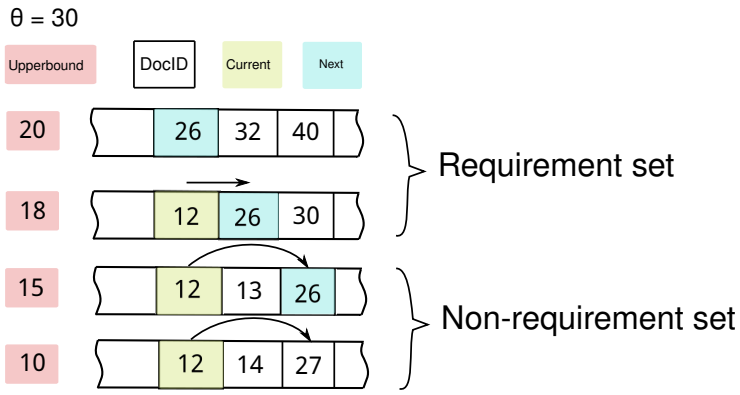


Figure 2.1: Dynamic pruning illustration. A dynamic threshold θ is the lowest score in the top-k results. Items in the non-requirement set alone which cannot produce a score higher than θ are safely skipped.

are called a *non-requirement set*, the opposite being the *requirement set*. Posting lists in the non-requirement with IDs less than the current IDs in the requirement set can be skipped safely, because posting lists in the former set alone will not produce a top-k candidate. The threshold used for pruning can be rescaled artificially by a factor of F to gain further efficiency, denoted as $\theta' = F \cdot \theta$, which means the pruning strategy is no longer guaranteed to be rank-safe. However, dynamically adjusting this factor F based on the perceived difficulty or *easiness* of a query can further enhance retrieval speed with minimal degradation in effectiveness, as demonstrated by Tonellotto et al. [2013].

2.2 Supervised Retrieval

Traditional IR methods commonly employ tf–idf variants for similarity scoring, which rely on exact lexical matching to retrieve relevant documents. While tf–idf takes into account corpus statistics, specifically in the *idf* factor, to determine the importance of terms within the corpus, a drawback of conventional unsupervised lexical retrieval is its inability to identify documents that use synonyms to the query. This issue, known as the *vocabulary mismatch* or *lexical gap* problem [Furnas et al., 1987], hinders the retrieval of documents that contain equivalent meaning but different terminology. Traditionally, the Latent Semantic Indexing (LSI) [Dumais, 2004, Hofmann, 1999] approach has attempted to address the vocabulary mismatch issue by employing techniques such as Singular Value Decomposition (SVD) compression or topic modeling to construct dense representations and index them

for retrieval. However, LSI has been found to exhibit only marginal improvements compared to BM25 when using a sufficient number of orthonormal vector dimensions [Atreya and Elkan, 2011]. Furthermore, LSI is prone to generating false positive results [Ai et al., 2016] and has demonstrated limited relevance matching capability [Guo et al., 2016]. To overcome these drawbacks, recent studies by Nalisnick et al. [2016] and Huang et al. [2013] have shown the importance of including relevance signals during training, such as utilizing large query logs, in order to achieve improved retrieval effectiveness. This highlights the significance of incorporating relevance information during the training process to enhance the retrieval performance using data-driven methods.

Prior to the rise of the Transformer architecture (see below), methods based on deep learning attempted to infer relevance signals from large click-through data or query logs [Huang et al., 2013, Xiong et al., 2017, Dai et al., 2018, Zamani et al., 2018], or through weak supervision by strong unsupervised relevance model such as BM25 [Dehghani et al., 2017]. However, these studies do not show empirically stronger and more robust retrieval results compared to traditional retrieval. The emergence of Transformer models has been a milestone in information retrieval, as their capacity to capture higher-level semantics through non-linear transformations has significantly enhanced in-domain effectiveness.

2.2.1 Transformer Encoder

The Transformer architectures [Vaswani et al., 2017, Devlin et al., 2019] have spurred the development of deep encoder models for passage retrieval. A Transformer-based encoder can be efficiently trained end to end, enabling it to capture contextual and higher-level semantics, thereby naturally bridging the lexical gap. At the core of the architecture lies a *self-attention mechanism* placed at each layer. This mechanism computes attention scores for every pair of tokens at positions i and j:

$$o_j = \frac{1}{Z} \sum_{i}^{n} f_s(q_j, k_i) g(v_i), \quad j = 1, 2...n \tag{2.3}$$

where a similarity or alignment function f_s is learned to generate a scaler score from an input query $q_j \in \mathbb{R}^d$ and each input key $k_i \in \mathbb{R}^d$, and it is then applied onto each input value $v_i \in \mathbb{R}^p$ to produce an output $o_j \in \mathbb{R}^p$ using a normalization factor Z. The attention mechanism introduced originally from machine translation [Bahdanau et al., 2014, Luong et al., 2015] usually takes q_j from hidden state generated from previous steps in an auto-aggressive way, while in self-attention, or intra-attention [Lin et al., 2017], the query is directly taken from input signals similar to query and key.

Transformer uses *multi-head self attention* where it computes multiple parallel heads of dimension $m = d/M$ indexed by $h = 1, 2, ..., M$. In this case, the key and value have the same dimension as the input values $X \in \mathbb{R}^{n \times d}$ $(d = p)$. The query, key, and value vectors are obtained through linear transformations $W_v^{(h)}, W_k^{(h)}, W_q^{(h)} \in \mathbb{R}^{d \times m}$ for each head.

$$O^{(h)} = \text{softmax}\left(\frac{(XW_q^{(h)}) \cdot (XW_k^{(h)})^T}{\sqrt{d}}\right) \cdot XW_v^{(h)} \tag{2.4}$$

$$\text{Att}(X) = [O^{(1)}...O^{(M)}] \cdot W_o \tag{2.5}$$

where the softmax function is applied to the last dimension by defining the similarity function as embedded Gaussian $f_s(q_j, k_i) = \exp(\theta(q_j)^T \phi(k_i))$ [Wang et al., 2018] with scaled dot-product [Vaswani et al., 2017]. Functions θ, ϕ, g are all linear transformations, and the output for this attention operation is also pooled from a linear transformation $W_o \in \mathbb{R}^{d \times d}$ of all heads. Furthermore, it can be shown that the self attention function is permutation equivariant [Ji et al., 2019, Yun et al., 2019].

A Transformer block at layer l is further stacked with elements such as dropouts [Hinton et al., 2012], residual connection [He et al., 2016], and fully-connected layers that are applied to each position separately. If we denote fully-connected feed-forward layers as parameterized non-linear function F and ignore dropouts, we can write down the Transformer layer $T^{(l)}$ at layer l as

$$Y^{(l)} = \text{LayerNorm}(X^{(l-1)} + \text{Att}^{(l)}(X^{(l-1)})) \tag{2.6}$$

$$T^{(l)}(X) = \text{LayerNorm}(Y^{(l)} + F^{(l)}(Y^{(l)})). \tag{2.7}$$

where the LayerNorm is a special variant of layer normalization [Ba et al., 2016] applied to the hidden dimension for each token position independently. The abundant residual connections used in the Transformer are potentially essential in gradient-based iterative optimization for compressing input token sets [Ma et al., 2022b, Yu et al., 2023]. Also, with more shortcut connections, the loss function landscape has more convex regions [Li et al., 2018]. Unlike recurrent neural networks [Hochreiter and Schmidhuber, 1997, Cho et al., 2014], which generate the query in an autoregressive manner, the Transformer encoder enables more direct information propagation. This language modeling approach also mitigates the *vanishing gradient* problem by utilizing *positional encodings* instead of relying on recurrence. On the other hand, the feed-forward layers are shown to store key-value memories [Geva et al., 2020, Dai et al., 2021], and the so-called Add & Norm operations shown in Eq. 2.6 and 2.7 can be altered to a pre-normalization to gain training efficiency [Baevski and Auli, 2018, Xiong et al., 2020b, Geiping and Goldstein, 2022].

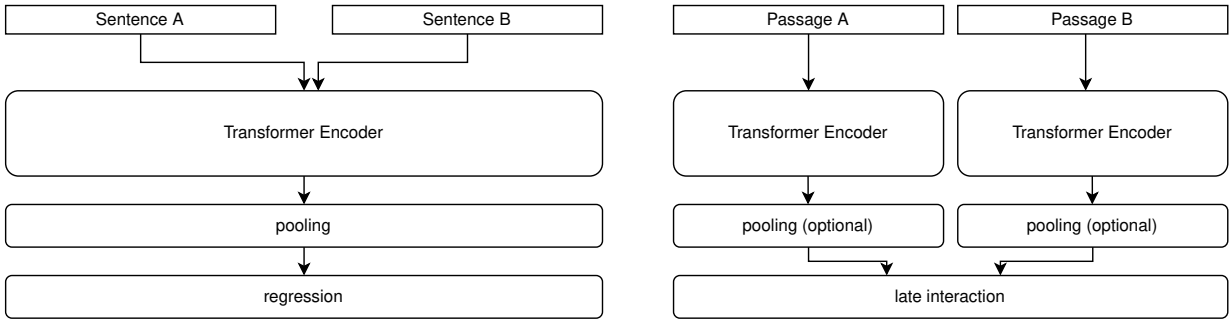


Figure 2.2: The architectures of a cross encoder (left) and a bi-encoder (right). The bi-encoder architecture generates separate representations for different passages. This enables the indexing of these representations offline, reducing the computational load during inference. Consequently, only a less expensive late interaction needs to be computed online.

Based on the encoder architecture of Transformer, the BERT model [Devlin et al., 2019] proposed Masked Language Model (MLM) and Next Sentence Prediction (NSP) self-supervising objectives, usually trained on large corpora of text datasets. The BERT model and its descendants, e.g., RoBERTa [Liu et al., 2019] and DeBERTa [He et al., 2020], have achieved the state-of-the-art effectiveness in numerous down-stream tasks.

2.2.2 Neural Retriever

The introduction of the MS-MARCO dataset [Nguyen et al., 2016] has established a standardized and user-friendly benchmark for deep neural retrievers that process large amounts of data. Evaluations on MS-MARCO using large pretrained language models such as BERT [Devlin et al., 2019] or T5 [Raffel et al., 2020, Ni et al., 2021] have showcased unparalleled effectiveness by simply reranking the top-1000 candidates generated from traditional retrieval systems [Nogueira and Cho, 2019, Nogueira et al., 2019b]. Conventionally, Transformer-based rerankers as such are called *cross encoder* or *interaction-focused architectures* [Guo et al., 2020, Lin, 2022], because they estimate similarity through full pair-wise token interactions between both query and candidate passages. Another successful application demonstrated on MS-MARCO is the expansion of document passages to incorporate synonyms or semantically relevant words [Nogueira et al., 2019c,a], aiming to address the vocabulary mismatch problem. These pivotal advancements have sparked extensive research into leveraging the power of Transformer models to construct more effective neural retrievers in the field of information retrieval.

However, performing direct inference with standard Transformers can be computationally intensive, as it involves passing the input through the entire network with hundreds of millions of parameters. Consequently, alternative two-stage inference approaches have been proposed. In these approaches, the first stage generates similarity representations offline for each query or document passage independently, which are then indexed using efficient coding methods such as HNSW [Malkov and Yashunin, 2018]. This approach defers the calculation of similarity to a later stage, significantly reducing online computational costs. Encoders designed in this manner are referred to as *bi-encoders* or *dual encoders* [Humeau et al., 2019]. Figure 2.2 showcases the distinctions between a cross encoder and a bi-encoder. Bi-encoders are well-suited for first-stage retrieval as they can offload most of the computation to the indexing stage, thereby accelerating query processing.

Over the past decade, the utilization of deep neural network bi-encoders to generate more effective representations or to improve retrieval efficiency has given rise to various techniques [Guo et al., 2020, Lin, 2022]. In the context of this thesis, I will summarize a few relevant directions below.

Representations

The `[CLS]` token embedding trained from the NSP task [Devlin et al., 2019] is for capturing sentence-level semantics, making it a natural choice of representation for passage retrieval applications. Several approaches have utilized this embedding for retrieval purposes. DPR [Karpukhin et al., 2020] employs the last-layer `[CLS]` embedding directly for retrieval. CEDR [MacAvaney et al., 2019] employs multi-layer `[CLS]` token embeddings. Additionally, some other sentence encoders [Reimers and Gurevych, 2022, Zhan et al., 2020b, Carlsson et al., 2020, Izacard et al., 2021] utilize an aggregated embeddings from the last layer outputs of BERT. However, without fine-tuning or domain adaptation with relevance labels, the `[CLS]` embedding has been shown to be weaker than traditional embeddings [Reimers and Gurevych, 2022].

Moreover, the fixed dimension of `[CLS]` embedding cannot capture the semantics for long documents well [Luan et al., 2021], and single-vector representation may suffer from bad cross-domain generation [Santhanam et al., 2021, Formal et al., 2021a]. As a result, *multi-vector* or *token-level* embeddings from all BERT contextual outputs are also explored. In this case, the late interaction will need to be computed in a more expensive way. ColBERT models [Khattab and Zaharia, 2020, Santhanam et al., 2021] are notable examples in this direction. However, their practical adoption has been hindered by either the computational expenses related to indexing token-level dense representations or the complexity of the accelerated query processing pipeline [Santhanam et al., 2022a].

To address the efficiency challenge of using multiple dense representations, sparse lexical codes have been proposed and they allow for the utilization of inverted index structures and potentially leverages existing optimizations [Mallia et al., 2022, Mackenzie et al., 2021]. Early work in this direction involved representing passage tokens by assigning learned scale importance weights in the lexical space [Dai and Callan, 2020, Mallia et al., 2021, Lin and Ma, 2021]. Later advancements expanded the sparse representation approach to include semantically related words. Noteworthy models in this line of research include SparTerm/SPARTA [Bai et al., 2020, Zhao et al., 2020], TILDE [Zhuang and Zuccon, 2021a,b], and SPLADE [Formal et al., 2021b,a, 2022]. These models leverage the MLM outputs from the Transformer encoder to identify synonyms for expansion.

Contrastive learning and hard negatives

The use of contrastive learning has significantly enhanced the performance of neural retrievers [Karpukhin et al., 2020, Xiong et al., 2020a]. Within the domain of information retrieval, encoders are commonly trained using both relevant (positive) and irrelevant (negative) samples to effectively acquire similarity representations. In general, gradient descent-based contrastive learning exhibits advantages with larger batch sizes [Chen et al., 2020], as employing smaller "mini-batches" can lead to increased learning variance [Qian and Klabjan, 2020] and smaller sample sizes compared to full batch gradient descent. Consequently, by predominantly relying on large batch sizes, neural retrievers can significantly enhance their effectiveness, as exemplified by approaches like RocketQA [Qu et al., 2020]. In the case of fixed memory capacity, when training dense retrievers using contrastive learning, including training samples within the same batch as *in-batch negatives* [Karpukhin et al., 2020] can further augment negative samples during training. This widely adopted strategy is employed by highly effective retrievers such as ColBERT-v2 [Santhanam et al., 2021] and SimLM [Wang et al., 2022]. To overcome hardware limitations, a commonly employed technique called *gradient cache* has been used to alleviate GPU memory consumption. This technique involves performing an additional forward pass of the encoder and caching the gradients of the representations computed from another forward pass without the need for a computational graph [Chen et al., 2016, Gao et al., 2021b].

Xiong et al. [2020a] demonstrated that the convergence of retrieval model training depends on the informativeness of constructed negatives. To enhance the informativeness, they employ *dynamic hard negative* mining from top retrieved documents and periodically update the model, requiring re-indexing. In contrast, Zhan et al. [2020a] employ top-ranked results as hard negatives but adopt a different approach by only altering the query representation, thereby eliminating the need for re-indexing. Gao et al. [2021a] investigated the

utilization of lower-ranked BM25 results as false positive samples, referred to as *static hard negatives*, independently of the neural model. Zhan et al. [2021] examine both dynamic and static hard negatives and provide an explanation for their superior effectiveness compared to random negative sampling.

To further improve retrieval effectiveness, GPL [Wang et al., 2021b] uses a stronger cross encoder to improve the quality in hard negative mining before training encoders. And given good pretraining objectives, Ram et al. [2021] has demonstrated a small set of 128 hard negative samples can create sufficiently effective retrievers. Lastly, SimLM [Wang et al., 2022], a recent dense retriever, has gained effectiveness by combining BM25 hard negatives and self-mined hard negatives in a multi-stage distillation pipeline.

Further pretraining

Recently, there has been a rise in the adoption of newly proposed pretraining schemes that mimic information retrieval (IR) tasks. These schemes are commonly referred to as *further pretraining* [Gururangan et al., 2020]. For retrieval purposes, Chang et al. [2020] have introduced the ICT pretraining objective. The ICT objective feeds passage-level inputs to each of the bi-encoders directly, rather than separating them using a `[SEP]` delimiter as seen in the NSP objective. Similarly, Ram et al. [2021] use an intersected span between two sample passages as positive pairs.

Instead of changing pretraining objectives, Gao et al. [2021c], Carlsson et al. [2020] use the siamese network with dropout applied to the same sentence inputs. An alternative approach by changing the model architecture involves the use of *auto-encoding* techniques where passages are encoded from the `[CLS]` embedding, serving as a bottleneck during pretraining. These models are commonly referred to as *Masked Auto Encoding* (MAE) models [Lu et al., 2021, Gao and Callan, 2021a, Wu et al., 2022, Xiao and Liu, 2022].

In the studies conducted by Ma et al. [2021a,b], words are sampled as pseudo queries from the training passage, and the encoder-decoder architecture is trained to predict this set of words. Gao and Callan [2021b] introduced an auxiliary Masked Language Modeling (MLM) objective within the decoder. This objective, in addition to relying on the `[CLS]` token, utilizes lower-layer token embeddings to recover masked words. In subsequent work, Gao and Callan [2021a] further expanded on this approach by incorporating a contrastive learning objective that incorporates co-occurring span pairs and in-batch negatives. Wang et al. [2021a] instead model the pretraining using a de-noise autoencoder, injecting "noise" to a sentence by deleting or swapping words. COSTA [Ma et al., 2022a] incorporates a contrastive span prediction task where the `[CLS]`-generated projector is tasked with

recovering the index of randomly sampled spans from the encoder. In contrast, Wang et al. [2022] use ELECTRA-style objectives on an MAE architecture by altering tokens using a generator and predicting original tokens at all positions.

2.3 Math Information Retrieval

Math Information Retrieval (MIR) is a field of study that encompasses various research interests. It includes areas such as math content detection, math-aware retrieval, handwritten formula recognition, and more. Several surveys have been conducted to provide overviews of the field, including works by Zhao et al. [2008], Zanibbi and Blostein [2012], and Meadows and Freitas [2022]. At the core of Math Information Retrieval (MIR) lies the fundamental component of information retrieval, which enables the effective and efficient mining and searching of mathematical content. Unlike conventional IR, math-aware retrieval specifically addresses the handling and consideration of structured math formulas in both queries and documents.

While studies on handwritten formula recognition [Labahn et al., 2008, Nishizawa et al., 2020] play a crucial role in facilitating user interfaces for the input of highly structured math formulas, this thesis primarily emphasizes the information retrieval aspect of Math Information Retrieval (MIR). Consequently, it provides a historical overview of related work and the evolution of MIR from the perspective of retrieval systems, following the approach outlined by Novotnỳ [2021].

2.3.1 Early Work (2003 – 2013)

The published research of MIR dates back to the DLMF project from NIST two decades ago [Miller and Youssef, 2003]. This line of early explorations has touched on many important and unique aspects of this domain, including problem definition [Youssef, 2005], detecting equivalence in expressions [Shatnawi and Youssef, 2007], relevance ranking [Youssef, 2007], search highlighting [Miller and Youssef, 2008]. Some early and demonstrative research systems are developed, including ActiveMath [Melis et al., 2006], MathFind [Munavalli and Miner, 2006], Whelp [Asperti et al., 2006], MathDex [Miner and Munavalli, 2007], EgoMath [Mišutka and Galamboš, 2008], MathGO! [Adeel et al., 2008], etc.

In the early stages of Math Information Retrieval (MIR), many researchers employed a technique known as "textualization" to convert structured mathematical formulas into plain words. This textualization process facilitated the integration of mathematical content

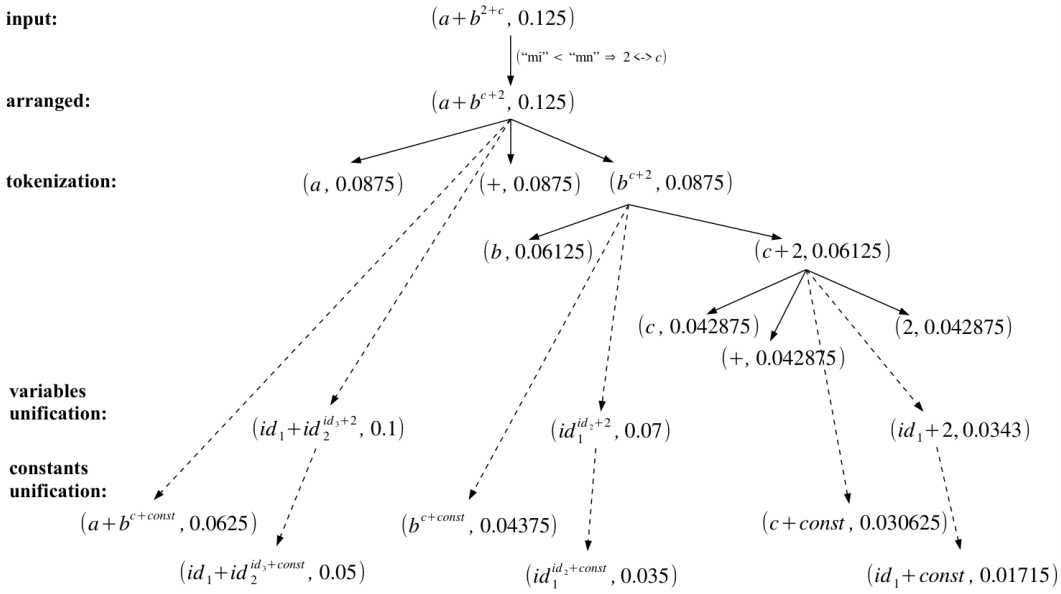


Figure 2.3: The textualization process employed by the MIaS system [Sojka and Líška, 2011b] involves multiple levels of decomposition and unification on a simple mathematical expression.

into traditional search engines by representing formulas as text. Studies such as Youssef [2005] and Kim et al. [2012] explored this approach to enable the retrieval of mathematical information using existing search engine infrastructure. However, to enable the recall of sub-expressions or permuted versions of a mathematical formula, it is necessary to decompose the formula into multiple "canonicalized" forms, to account for properties like commutativity. The process of decomposing the formula into these variations is often referred to as *augmentation* [Mišutka and Galamboš, 2008, Sojka and Líška, 2011b]. An example system, MIaS [Sojka and Líška, 2011b,a], generates a large number of sub-expressions for a simple query input (shown in Figure 2.3). In the case of formulas with a large number of operands and levels, it becomes impractical to enumerate all possible augmentations due to the exponential growth of potential variations. As a result, a common practice is to limit the maximum number of augmentations that are indexed [Mišutka and Galamboš, 2008]. However, it is important to note that restricting the number of augmentations can result in reduced recall during the retrieval process.

To evaluate formula similarity, systems often employ techniques beyond traditional text search scoring. One approach is to assign empirical weights to each decomposed subtree based on factors such as height and n-gram length, as demonstrated in studies like Miner

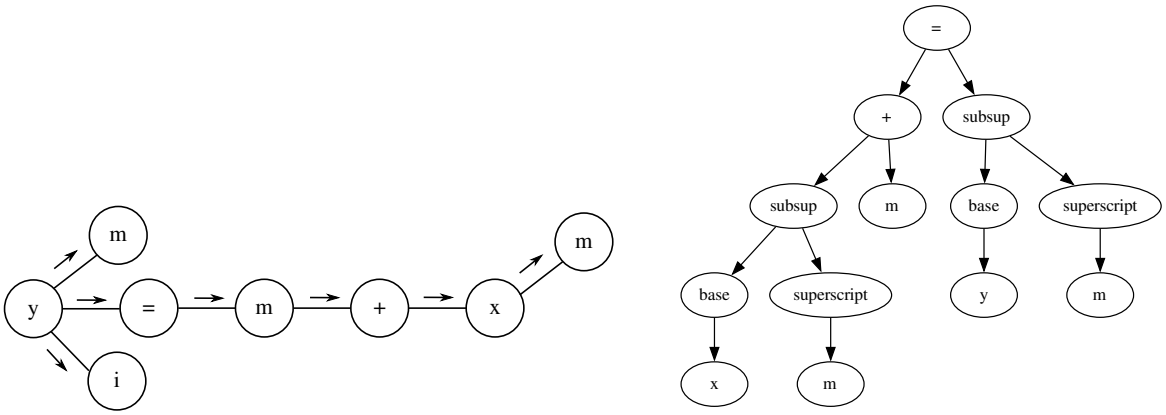


Figure 2.4: From left to right: Example SLT (Symbol Layout Tree) and OPT (Operator Tree) representations, respectively. Both represent the same math formula $y_i^m = m + x^m$. SLT focuses on the visual structure, while OPT emphasizes operand relations and operations.

and Munavalli [2007] or solely based on height as explored in Sojka and Líška [2011b]. However, using n-gram length alone may not effectively capture the similarity especially when mathematical markup languages use multiple characters to represent a single variable.

Systems like Whelp [Asperti et al., 2006] and MWS [Normann and Kohlhase, 2007] take a different approach by converting mathematical language into first-order clauses in a formal proof language. Rather than evaluating similarity, these systems focus on filtering results based on equivalence or provability relationships between expressions. While this formal evaluation can provide rigorous results, it poses challenges when applied to real-world datasets. Maintaining an index for such systems requires significant memory resources, as highlighted in Kohlhase et al. [2012]. Additionally, user-generated math expressions often cannot be parsed correctly, which compromises the cost-effectiveness of these systems compared to simpler alternatives [Aizawa et al., 2014].

As a result of previous explorations, a structure-based matching metric becomes necessary. There are two major structure representations in early work, i.e., SLT [Stalnaker, 2013] and OPT [Shatnawi and Youssef, 2007, Hijikata et al., 2007]. Figure 2.4 provides a visual representation of an instance tree for each representation. The Symbol Layout Tree (SLT) captures the lower-level structural semantics of math formulas, resembling the layout and topology of a formula in LaTeX representation. SLT focuses on preserving the visual layout and topology, offering the advantage of minimal ambiguity during parsing. On the other hand, the Operator Tree (OPT) representation identifies operators and operands in

the math formula and constructs a recursive tree structure. Each internal node represents an operator, while the children correspond to its operands.

OPT has been widely used in the early development of math information retrieval (MIR) systems [Miner and Munavalli, 2007, Shatnawi and Youssef, 2007, Hijikata et al., 2007]. The initial application of the Operator Tree (OPT) representation for search purposes involved manually constructing trees and detecting equivalence between math formulas using context-free grammars [Shatnawi and Youssef, 2007]. However, subsequent works [Miner and Munavalli, 2007, Hijikata et al., 2007, Mišutka and Galamboš, 2008, Adeel et al., 2008] adopted OPT-like tree representations using existing MathML markups.

In the context of this thesis, there is a specific line of early work that focuses on using leaf-root paths from the Operator Tree (OPT) representation for formula search. Several studies, such as Hijikata et al. [2007, 2009], Yokoi and Aizawa [2009], have explored this approach, leveraging paths from the leaves to the root of the tree to derive commutativity-invariant features for retrieval purposes. These path-based features require $O(N \cdot I)$ space to index a formula, where N represents the number of leaves or operands in the OPT and I represents the number of internal nodes which is considered manageable (especially when OPTs have a limited height in the math documents). The use of leaf-root paths for retrieval was first proposed by Hijikata et al. [2007] as *XPaths*. However, this approach had a high recall but lacked precision due to the absence of whole structure comparison. Hijikata et al. [2009] improved precision by incorporating horizontal sibling nodes, but this led to a lower recall. To address this, Yokoi and Aizawa [2009] and Hagino and Saito [2013] expanded the leaf-root path set by extracting prefixes and suffixes, creating a subpath set. This enhancement improved recall and allowed retrieval of both sub-expressions and subtree-wildcards in formulas.

Early work in the field of MIR has explored various structure distances to measure similarity. Kamali and Tompa [2009] introduced a similarity measurement called *structural n-similarity*, which computes the ratio of matched nodes to all nodes being compared. In subsequent work, Kamali and Tompa [2010] proposed a structural inverted index that treats subtree hash signatures as words, enabling substructure matching through cached dynamic programming. To further enhance efficiency, Kamali and Tompa [2013] introduced early termination techniques for accelerated similarity computation. In a different approach, Zanibbi and Yu [2011] utilized dynamic time warping (DTW) to identify similarity in pixel-level columns from image inputs. These approaches aimed to address the computational complexity of formula structure search by transforming the problem into more efficient setups. However, their matching metrics may not be generally effective as they often overlook holistic substructure matches or structural variations.

2.3.2 Better Structure Search (2014 – 2019)

Comparing the effectiveness of MIR systems in a standardized way was challenging until the introduction of relevant tasks by two major evaluation platforms: NTCIR [Aizawa et al., 2013, 2014, Zanibbi et al., 2016a] and ARQMath [Mansouri et al., 2020a, 2021c, 2022], along with their datasets. Since the pilot MIR task in NTCIR-10 [Aizawa et al., 2013], better structure search methods by matching substructure(s) on different intermediate structure representations have gained popularity [Gao et al., 2016, Kristianto et al., 2016, Zanibbi et al., 2016b, Fraser et al., 2018, Zhong and Zanibbi, 2019] due to their remarkable effectiveness in these evaluations.

WikiMirs/ICST Hu et al. [2013] build conventions about basic similarity axiomatic rules and generalized matching of wildcards. It proposes a primitive parser rule to construct structure representation for LaTeX markup, and extract normalized expression strings at each level for indexing and modified tf–idf retrieval weighted by subexpression level offset in tree representations. In WikiMirs v2 [Lin et al., 2014], an enhanced OPT is constructed with semantic enrichment, e.g., lexicographically sorted children under a commutative operator. The scoring of similarity is generalized to consider multiple relevant formulas in a document by weighted sum. In WikiMirs v3 [Wang et al., 2015], a hybrid search approach is introduced that combines formula search and text retrieval. Furthermore, formula importance is incorporated, similar to idf, but also considering other factors such as whether a formula occupies a whole line by itself. By applying learning to rank techniques on Wikipedia data, the ICST team achieves the top-performing results for Wikipedia retrieval in NTCIR-12 [Zanibbi et al., 2016a].

MCAT The MCAT system [Topić et al., 2013, Kristianto et al., 2014, 2016] leverages path features generated from OPT, specifically focusing on content MathML leaf-root paths. These path features include ordered paths, unordered paths, and additional sibling patterns to enhance search precision. This combination allows the system to retrieve formula subexpressions within both commutative and non-commutative operators. MCAT employs an indexing approach that captures leaf-root paths from all subtrees, similar to the approach described in Yokoi and Aizawa [2009]. Additionally, MCAT utilizes hashed substructure using the *SIGURE* variable name hashing [Ohashi et al., 2016]. These techniques contribute to MCAT achieving the highest precision scores in various formula search tasks, as demonstrated in NTCIR-12 [Zanibbi et al., 2016a]. Despite its effectiveness, the efficiency of MCAT is hindered by an extended scoring procedure required to unify structure wildcards in larger math formulas. Reportedly, MCAT has a median query execution time of approximately 25 seconds when using a server machine and multi-threading [Kristianto et al., 2016]. Furthermore, MCAT's matching of formula structures relies on text

words and utilizes default scoring in Lucene/Solr. As a result, their system may not detect partial structure matches with different variable names.

Tangent systems The original Tangent system [Stalnaker, 2013] is built upon symbol pairs from SLT. It retrieves formulas using a "bag-of-pairs" boolean search model. The first version named Tangent-1 [Pattaniyil and Zanibbi, 2014] is evaluated in NTCIR-11. It supports the retrieval for math matrices by considering them as symbols separated by rows. Additionally, Tangent-1 uses a standard tf–idf scoring model offered by Lucene. Tangent-2 [Stalnaker and Zanibbi, 2015] follows the same approach, but it is rewritten in Python and offers a more comprehensive search interface. It utilizes a de-coupled in-memory index using Redis and implements a ranking function for improved performance and efficiency. Tangent-3 [Zanibbi et al., 2015, Davila et al., 2016, Zanibbi et al., 2016b], proposed for the NTCIR-12, employs a two-stage cascading ranking where a coarse but efficient core engine based on an iterator tree index is used to recall formulas ranked by Dice's coefficient; and a second stage reranking is done with expensive maximum structure alignment (i.e., *Maximum Subtree Similarity*, or MSS) and variable unification. The first stage is more efficient due to a constrained symbol pair window and a faster core engine written in C++. Both stages incorporate the SLT representation and employ a weighted sum score for ranking. In subsequent work, Davila and Zanibbi [2017] introduces Tangent-S, which significantly improves effectiveness. Building upon these insights, Tangent-V [Davila et al., 2019] is proposed, where the method is extended to handle visual representations such as PDFs and images. Without requiring explicit parsing of the image content, Tangent-V leverages a *line-of-sight graph* that simulates symbol pairs. This approach outperforms concurrent systems in terms of partial relevance score, demonstrating the general effectiveness of symbol pairs in dealing with visual representations.

Tangent-L [Fraser et al., 2018, Dallas, 2018] deviates from Tangent-3 and introduces *math tuples*, which are carefully curated orthogonal features[1]. The MathDowsers system [Ng et al., 2021, Ng, 2021], and the more recent μTextSearch system [Kane et al., 2022] employ ongoing improvements to the representation of math tuples, making continuous progress in subsequent ARQMath tasks [Mansouri et al., 2021c, 2022].

Approach0 (Chapters 3 and 4) Drawing on prior works exploring the use of leaf-root paths for formula search [Hijikata et al., 2007, 2009, Yokoi and Aizawa, 2009], an early approach is introduced to identify substructures while simultaneously matching a comprehensive set of paths [Zhong and Fang, 2016, Zhong, 2015]. By incorporating pseudo rank nodes, this method can also distinguish non-commutative matches, akin to the ordered path effects utilized by MCAT [Topić et al., 2013]. Subsequently, Zhong et al. [Zhong

[1]For examples, refer to the open-source parser package: https://github.com/fwtompa/mathtuples

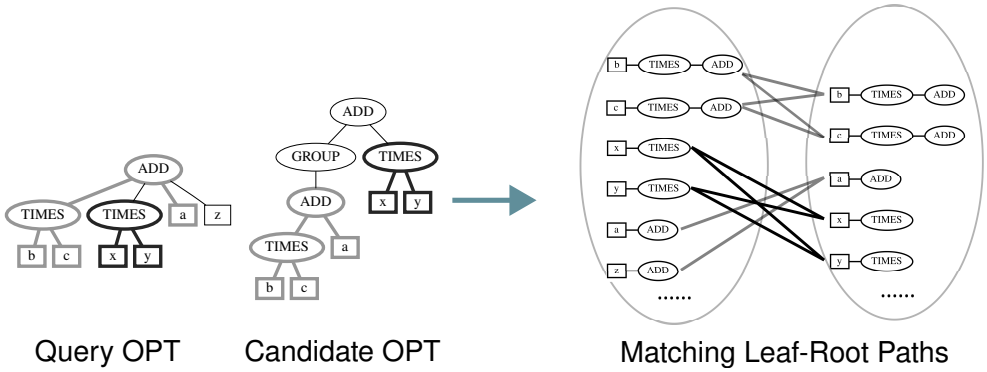


Figure 2.5: Structure search based on leaf-root paths. Matching substructures between OPT representations of the query formula $bc+xy+az$ and a candidate formula $(bc+a)+xy$.

and Zanibbi, 2019] expand on the OPT leaf-path matching technique by including path prefixes and proposing an approximate structure matching method that dynamically groups the root-end nodes of leaf-root paths. As depicted in Figure 2.5, the proposed structure matching metric identifies the maximum common subtree(s) between two mathematical formula OPTs from their leaf-root path set. This can be viewed as an extension of the MSS metric from Tangent-S [Davila and Zanibbi, 2017], but applied to only bottom-up subtrees within the OPTs. By employing structure-based dynamic pruning techniques [Zhong et al., 2020], structure search in OPT representations can be made efficiently enough to support practical first-stage retrieval. More recently, evaluations [Mansouri et al., 2021c, 2022] demonstrate that this approach achieves high precision in candidate selection without the need for a subsequent fine-grained structure matching stage.

2.3.3 Data-Driven Retrieval (2017 – Now)

Data-driven methods can complement structure searches by revealing connections among mathematical formulas with diverse structures. Previous MIR research in this area has explored self-supervised clustering algorithms and effective supervised methods that leverage Transformer models. Below, I provide a summary of advancements in this direction.

Learning embeddings from linear math tokens Initially, data-driven methods have been used for the clustering of math formulas [Samarasinghe and Hui, 2009, Ma et al., 2010]. As highlighted by Gao et al. [2017], the straightforward linear clustering of formula symbols has limited success. They emphasize that mathematical formulas exhibit structural properties where an operator is typically dissimilar to its local neighbors, for instance,

the addition operator and its operands are often incorrectly assigned to the same cluster.

Learning embeddings from structural representations To address the issue above, structure representations are used to explicitly model the clustering of math semantics. For example, Tangent-CFT learns word embeddings by treating operators, SLT, and OPT features as words. Zhang et al. [2021] study operator embeddings based on the slant of transformation between steps, while Dai et al. [2020] learn word embeddings from n-ary tree SLT representations. More recently, Song and Chen [2021] and Pfahler and Morik [2022] construct graph embeddings directly from OPT or MathML structures using Graph Neural Networks (GNNs). Similarly, Wang et al. [2021c] generates tree embeddings for OPT representations in an auto-encoding setting. However, employing GNNs to measure structure similarity is considered less appropriate compared to solely determining graph isomorphism, as regular GNNs do not possess more power than the Weisfeiler-Lehman test [Xu et al., 2018], which is often regarded as overly restrictive in defining similarities [Schulz et al., 2022]. So far, the effectiveness of GNNs is still relatively inferior to (unsupervised) structure search [Song and Chen, 2021, Wang et al., 2021c].

Capturing contextualized semantics, including the surrounding text Krstovski and Blei [2018] treat SLT features as words but additionally consider the surrounding text in a topic modeling setting. Yasunaga and Lafferty [2019] use a variational autoencoder to parameterize the topic distribution and explicitly model the sequence generation of formula tokens using a topic-extended LSTM with context information. However, these work have not been evaluated on common retrieval datasets. In a recent attempt [Ferreira and Freitas, 2021], math contextual embeddings are learned through unsupervised learning based on a siamese network. However, the retrieval effectiveness (measured by F1-score) is similar to a vanilla BERT model which is not explicitly trained for the math domain.

Tuning with relevance signals The problem with self-supervised clustering for retrieval is that learned representation may not contain sufficient similarity signals for effective retrieval [Greiner-Petter et al., 2020]. Without using a Transformer, Allamanis et al. [2017], Gangwar and Kani [2022] have applied recursive neural network and seq-to-seq model (with attention mechanism) to learn equivalent math expressions in a supervised objective. To boost ranking effectiveness, the Tangent-CFTED system [Mansouri et al., 2020b] – an upgraded version of Tangent-CFT – uses tree edit distance with carefully tuned edit weights to rerank candidates retrieved from unsupervised embedding representations.

With the surge of the Transformer architecture [Vaswani et al., 2017], the use of a Transformer encoder with supervised retrieval signals to capture contextualized and highly abstract math semantics has become commonly practiced in the domain of MIR [Peng et al., 2021, Reusch et al., 2021a, Mansouri et al., 2021c, Zhong et al., 2022a, Novotnỳ

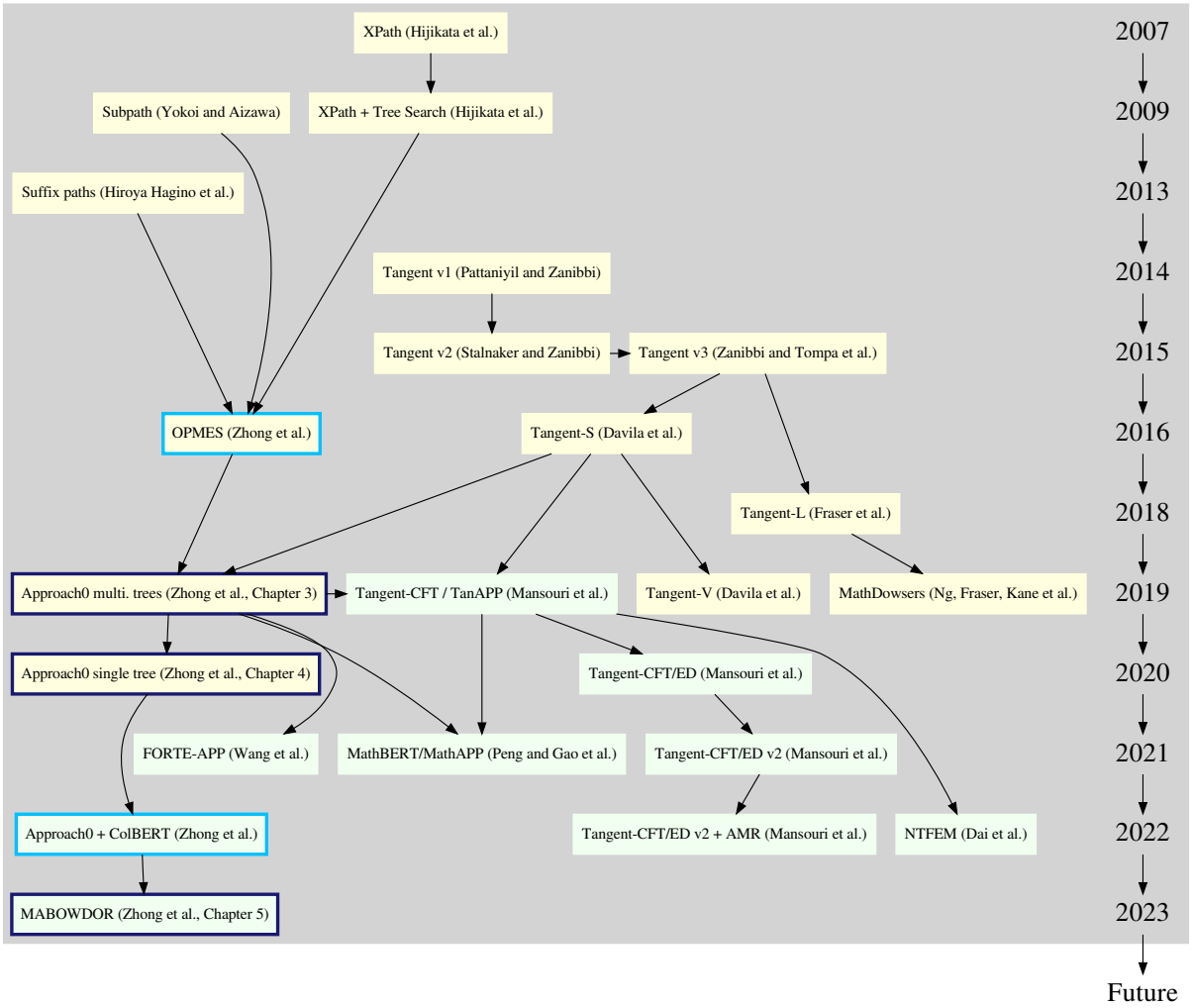


Figure 2.6: The ontology of our systems and related retriever families in chronological order. Dark blue border for our work discussed in this thesis, and light blue border for our work not discussed in this thesis. Light yellow nodes denote unsupervised or semi-supervised systems, and light green nodes denote supervised systems. The arrow denotes a dependency or is inspired by.

and Štefánik, 2022, Geletka et al., 2022]. Indeed, recent research has demonstrated that combining specialized structure search for scoring structure similarity with neural retrievers for capturing other semantic aspects can lead to highly complementary results [Zhong et al., 2022b,a, Kane et al., 2022].

The MathBERT model [Peng et al., 2021], evaluated on the NTCIR-12 collection [Zanibbi et al., 2016a], represents the first successful neural math retriever based on a Transformer encoder. It incorporates structure mask pretraining in addition to the pretraining objectives employed in BERT. For reranking, MathBERT utilizes the feature vectors from the last two layers and operates on a high-quality candidate pool generated by Tangent-CFT.

The creation of ARQMath datasets [Mansouri et al., 2020a, 2021c, 2022] has demonstrated the success of Transformer-based models, leading to wider adoption of deep models in MIR. One notable example is the work by Novotnỳ et al. [2020, 2021], which utilizes a bi-encoder based on SentenceBERT [Reimers and Gurevych, 2022] to predict the percentage of upvotes for a question-answer pair within its original MSE thread. Rohatgi et al. [2020, 2021] rerank conventional ad-hoc search results using the full token embeddings averaged from a pretrained RoBERTa Transformer. The DPRL QASim method [Mansouri et al., 2021b] employs two Transformers as similarity assessors. The first one is a question-question SentenceBERT assessor, which is pretrained on the Quora website and fine-tuned using related/duplicate links from the MSE website; the second assessor is a question-answer model using TinyBERT [Jiao et al., 2019], pretrained on the MS-MARCO dataset [Bajaj et al., 2016] and fine-tuned on the ARQMath-1 training data. The similarity score generated by QASim is the product of the scores provided by these two assessors. In the question-question model, the similarity is evaluated between the topic question and the question associated with the document answer. The TU_DBS systems [Reusch et al., 2021b,a, 2022] has been the first to demonstrate that neural retrievers can achieve state-of-the-art precision in math-aware retrieval on the ARQMath datasets. Their approach during the ARQMath 2021 competition utilizes a cross encoder as the primary model [Reusch et al., 2021b,a], while also exploring a less-effective ColBERT bi-encoder [Khattab and Zaharia, 2020] for the initial retrieval stage. Notably, they were also pioneers in applying deep models to formula search (Task 2). The backbone model of TU_DBS is based on the ALBERT Transformer [Lan et al., 2019], which is pretrained directly on the ARQMath corpus, with a maximum token input length of 512.

In our recent work [Zhong et al., 2022b,a, 2023], I extensively explore the combination of an efficient structure search method with other finely tuned bi-encoder Transformer alternatives. Figure 2.6 provides an overview of the ontology related to our system, offering insights into its historical development and various related systems.

However, hiding behind the great modeling power, issues like deployment complexity, software pipeline cost, and efficiency are becoming more dominant considerations in Transformer-based language modeling [Santhanam et al., 2022b]. Existing effective neural retrievers in MIR, suffering from excessive input tokens and low budgets or resources, are no exception in this regard – top effectiveness is often obtained from multi-vector representations [Zhong et al., 2022a], heavy ensembles [Geletka et al., 2022, Kane et al., 2022], or running cross-encoder rerankers on a large set of candidates [Reusch et al., 2021b, 2022].

2.4 Experiment Datasets

Compared to the general information retrieval tasks, MIR dataset remains scarce. There are two evaluation platforms that have hosted MIR tasks, namely NTCIR[2] and more recently, CLEF.[3] In this thesis, the evaluation and comparison of systems and their effectiveness are conducted exclusively on these datasets. To evaluate the efficiency and to pretrain the supervised retriever models, which require a larger dataset, I utilize a self-hosted corpus obtained by crawling data from an additional data sources, i.e., the Art of Problem Solving (AoPS) community WEB pages.[4]

NTCIR-12 Wiki Math Browsing

The NTCIR-12 Wiki Math Browsing (WMB) [Zanibbi et al., 2016a] is a formula search task made from math-related pages in Wikipedia. It is widely compared by different systems and it is also the most recent MIR dataset from the NTCIR series [Aizawa et al., 2013, 2014, Zanibbi et al., 2016a]. In contrast to earlier NTCIR MIR tasks [Aizawa et al., 2013, 2014], the WMB task focuses on evaluating similarity retrieval and employs more popular query formulas [Mansouri et al., 2021a].

The documents contain around 591,000 isolated formulas, and the query can either be a concrete formula or can contain wildcards specified by the special LaTeX command **\qvar** to match arbitrary subexpression or symbols, e.g., query \qvar{a}^{2} + \qvar{b}^{3} may match $x^2 + (y + 1)^3$. Judgment rating(s) are summed into a scale of 0 – 4, for each judged formula, the ratings are mapped to fully relevant (if grade ≥ 3), partially relevant (if grade ≥ 1), or irrelevant (if grade $= 0$). Top precisions are the official metrics, however, bpref [Buckley and Voorhees, 2004] is often used in post-hoc evaluations.

[2] https://research.nii.ac.jp/ntcir/index-en.html
[3] https://www.clef-initiative.eu
[4] https://artofproblemsolving.com/community

CLEF ARQMath

The Conference and Labs of the Evaluation Forum (CLEF) has hosted three MIR retrieval tasks over consecutive years [Mansouri et al., 2020a, 2021c, 2022]. ARQMath datasets offer a more comprehensive evaluation and contain five times more concrete math queries compared to NTCIR. Given ARQMath-3 Task 1 for example, the judgment pool consists of an average of 464 answer posts per topic, with a total of 80 assessed topics. All ARQMath datasets share a common corpus, which comprises approximately 1 million questions and 28 million formulas extracted from the Math StackExchange (MSE) website[5] between 2010 and 2018. Evaluation topics in each year are sampled from real-user questions posted after year 2018. Relevance levels during judgment have *High*, *Medium*, *Low*, and *Irrelevant* ordered from highest score 3 to lowest 0.

The initial two ARQMath collections (2020 and 2021, or ARQMath-1 and ARQMath-2) encompass two tasks. Task 1 is about Community Question and Answer (CQA) and entails retrieving relevant answer posts from the corpus using full-text queries sampled from MSE user-generated questions. Task 2 focuses on formulas and aims to identify pertinent formulas within the documents, considering their context, given a specific query formula provided in a Task 1 topic. To ensure formula diversity, this task necessitates returning a maximum of 5 visually distinct formulas; otherwise, the extra results will not be evaluated. In the most recent ARQMath 2022 collection (ARQMath-3), an additional Task 3 has been introduced. Task 3 is about Open Domain QA and requires participants to provide a single answer for each topic in Task 1. The answer can be automatically generated or extracted from existing data, potentially from outside the ARQMath collection.

ARQMath Task 2 and NTCIR-12 WMB share similarities as both are formula search tasks. However, NTCIR-12 uses isolated formula for each query without considering its context, while ARQMath Task 2 asks to retrieve a pertinent formula provided in math a question. In addition, ARQMath-3 Task 2 encompasses 76 assessed topics, with an average evaluation of 152.3 visually distinct formulas per topic. In contrast, the NTCIR-12 WMB task consists of approximately 60 assessed topics per topic.

MSE and AoPS (self hosted)[6]

By extracting user-generated content from the MSE and AoPS math Q&A websites, two math-content corpora were obtained. The size of these corpora ranges from 1 million to

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

[6] Can be downloaded from https://www.cs.rit.edu/~dprl/data/mse-corpus.tar.gz and from https://vault.cs.uwaterloo.ca/s/G36Mjt55HWRSNRR.

1.7 million documents. The larger corpus containing both MSE and AoPS data is intended for data-consuming pretraining purposes, while the other corpus is adjusted to a smaller size by only using MSE data. This adjustment ensures that the exhaustive search does not take an excessive amount of time to run, additionally, it allows for noticeable differences in query run times when comparing run times in Chapter 4.

2.5 Evaluation Measurements

The primary goal of information retrieval evaluation metrics is to measure the effectiveness of a retrieval system in terms of how well it retrieves the most relevant documents in the top-ranked results while minimizing the retrieval of non-relevant or irrelevant ones [Voorhees, 2019].

For ranked results at position i (default maximum $m = 1000$), a non-negative integer score rel_i is the judged relevance level given by human expert(s). Due to limited resources, usually only a small set of documents are judged, and selected documents are often contributed by participant systems; this is called pooling. In a post-hoc evaluation, which needs to consider those with unknown relevance (unjudged), the standard approach is to assume them as irrelevant. ARQMath adopts the prime-version metrics as official measurements which exclude unjudged search results, thus ensuring a fairer comparison for systems evaluated in a post-hoc experiment.

NDCG NDCG (Normalized Discounted Cumulative Gain) is defined by

$$\text{NDCG} = \frac{1}{Z} \sum_{i=1}^{m} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{2.8}$$

where Z is the normalization factor computed from a perfect ranking. The prime version NDCG' is proposed by Sakai and Kando [2008].

Precision The P@k (precision at position k) [Schütze et al., 2008] is a binary metric defined by the number of relevant items in top k results divided by k.

MAP MAP (Mean Average Precision) [Schütze et al., 2008] is a binary metric, calculated from averaged query precision (AP) scores where

$$\text{AP} = \frac{1}{m} \sum_{i=1}^{m} P@i. \tag{2.9}$$

bpref bpref [Buckley and Voorhees, 2004] is a binary metric that naturally excludes unjudged documents:

$$\text{bpref} = \frac{1}{|R|} \sum_{i \in [R]}^{|R|} \left(1 - \frac{\left| \{n \in N : n < i\} \right|}{\min(R, N)} \right) \tag{2.10}$$

where R and N are the indices of judged relevant and irrelevant documents, respectively.

For binary metrics (Precision, MAP, and bpref), the H+M binarization is used in ARQMath collections by collapsing High and Medium relevance to 1, and Low and Irrelevant to 0. For the NTCIR WMB collection, binarization is applied depending on the context, evaluating either fully relevant or partially relevant results.

Chapter 3

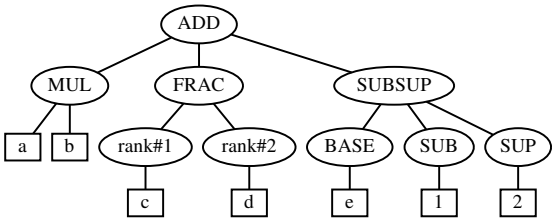
Effective Structure Search

When evaluating semantic similarities in math documents, it is crucial to consider a broad range of aspects. Among these aspects, structure similarity is relatively easier to define compared to similarities in natural language, in addition, results show that the presence of common (sub-)expressions is a strong indicator for the relevance of the formula or the document in which it is contained. This observation provides an effective yet simple approach to retrieving relevant math documents by comparing structure similarities in formulas. In this thesis, I will adopt an *axiomatic approach* [Fang and Zhai, 2005] to capture structure similarities in an unsupervised manner. This will involve explicitly designing a set of rules that the scoring function should adhere to.

3.1 Structure Matching

The Operator Tree (OPT) representation captures the structure of a mathematical formula by recursively constructing a tree that identifies operators and operands in each sub-expression. In this representation, internal nodes correspond to operators and their subtrees represent the operands (refer to Figure 3.2 for examples). I utilize this tree structure as an intermediate representation to assess similarity between formulas, as described in our previous studies [Zhong and Fang, 2016, Zhong, 2015].

At a high level, the identification of the largest common substructures between two OPT representations of formula pairs is accomplished by analyzing their *leaf-root paths* (see Figure 3.1 for an illustration). These paths, intrinsic to the operator tree, remain unaffected by the permutation of operand positions. Consequently, the path feature in



Unique leaf-root paths (after tokenization)
`VAR/MUL/ADD`
`VAR/rank1/FRAC/ADD`
`VAR/rank2/FRAC/ADD`
`VAR/BASE/SUBSUP/ADD`
`VAR/SUB/SUBSUP/ADD`
`VAR/SUP/SUBSUP/ADD`
`VAR/MUL`
`VAR/rank1/FRAC`
`VAR/rank2/FRAC`
...

Figure 3.1: Left: An example OPT representing a hypothetical formula $ab + \frac{c}{d} + e_1^2$. Leaf symbols are left untokenized for enhanced visibility. Right: Unique leaf root paths for this example OPT (including prefixes). Note: A non-commutative operator such as fraction (`FRAC`) will be attached by rank nodes [Zhong and Fang, 2016] to differentiate its actual operands. And a sub- or super-scripted symbol might be constructed in the above way so that it can be matched to a non-sub-scripted operand for a better recall.

Figure 3.2: Matching multiple formula subtrees between the OPT representations of formulas $a+bc+xy+z$ and $(a+bc)+xy$ as shown in the left and right of the figure respectively.

the OPT representation captures both the commutativity of operands and the existence of common substructures, the latter is captured by matching the prefixes of leaf-root paths. On the other hand, they can also provide an efficient approximation of structural overlaps. These statements are further illustrated in this chapter.

3.1.1 Structure Similarity

To formally define the structure matching approach, I incorporate graph or subtree isomorphism definitions [Shamir and Tsur, 1999]. The notation for a tree T is denoted as $T = T(V, E, r)$ which implies a labeled rooted tree with root r and edges $(v_1, v_2) \in E(T)$, moreover, each vertex $v \in V(T)$ (or $v \in T$) is associated with a label or *symbol* $\ell(v)$ which is not necessarily unique in a tree. Here I further define *formula subtree* and *common formula subtree* [Zhong and Zanibbi, 2019].

Definition 3.1.1. Given rooted trees S and T, we say S is a *formula subtree* of T, denoted as $S \preceq_l T$, if there exists an injective mapping $\phi : V(S) \to V(T)$ satisfying:

1. $\forall\ (v_1, v_2) \in E(S)$, we have $(\phi(v_1), \phi(v_2)) \in E(T)$;
2. $\forall\ v \in V(S)$, we have $\ell(v) = \ell(\phi(v))$;
3. If $v \in V(S)$ is a leaf vertex in S, then $\phi(v)$ is also a leaf in T.

Definition 3.1.2. A *common formula subtree* of two trees T_q and T_d consists of two corresponding formula subtrees $\hat{T}_q$ of T_q and $\hat{T}_d$ of T_d where they are isomorphic and also subgraphs of T_q and T_d respectively. Let $\mathrm{CFS}(T_q, T_d)$ denote the set of all such common formula subtrees of T_q, T_d, i.e., $\mathrm{CFS}(T_q, T_d) = \{\hat{T}_q, \hat{T}_d : \hat{T}_q \preceq_l T_q, \hat{T}_d \preceq_l T_d, \hat{T}_q \cong \hat{T}_d, \hat{T}_q \subseteq T_q, \hat{T}_d \subseteq T_d\}$ where "$\cong$" and "$\subseteq$" indicate graph isomorphism and subgraph relations respectively.

In addition to the regular common subtree isomorphism, a common formula subtree requires leaves in a subtree to be mapped to the leaves in the counterpart. In other words, formula subtrees are matched bottom up from the leaves (or operands).

Similar to the definition of a common forest [Valiente, 2002], I use disjoint common subtrees to describe multiple subexpression matches. Figure 3.2 illustrates two matching common subexpressions ($a + bc$ and xy), with the matches highlighted in grey and bold. We call these matches a *common formula forest*. It consists of common formula subtree(s) identified by disjoint $(\hat{T}_q^i, \hat{T}_d^i)$ pairs:

Definition 3.1.3. A set of common formula subtrees π is called a *common formula forest* of two formula trees T_q and T_d,

$$\pi = \{(\hat{T}_q^1, \hat{T}_d^1), (\hat{T}_q^2, \hat{T}_d^2), ...(\hat{T}_q^n, \hat{T}_d^n)\} \in \Pi(T_q, T_d) \tag{3.1}$$

if for $i = 1, 2, ...n$:

(1) $\hat{T}_q^i, \hat{T}_d^i \in \text{CFS}(T_q, T_d)$

(2) $\hat{T}_q^1, \hat{T}_q^2, ...\hat{T}_q^n$ are disjoint, and $\hat{T}_d^1, \hat{T}_d^2, ...\hat{T}_d^n$ are disjoint.

where $\Pi(T_q, T_d)$ denote all possible common formula forests of T_q and T_d.

For the structural similarity metric, I want to find the "largest" common formula forest to represent the most similar parts of two math expressions. In order to define "large" generally, the similarity function between two formula trees is parameterized by some scoring function γ of $\pi \in \Pi(T_q, T_d)$:

Definition 3.1.4. The *formula tree similarity* of T_q and T_d given scoring function γ is

$$\Gamma_\gamma(T_q, T_d) = \max_{\pi \in \Pi(T_q, T_d)} \gamma(\pi) \tag{3.2}$$

which converts the similarity assessment of a pair of formulas into a structure matching problem. Obviously, this similarity only captures one aspect of math formula semantic similarity, i.e., their structure similarity. Nevertheless, extensive studies [Davila and Zanibbi, 2017, Zhong and Zanibbi, 2019] have demonstrated that a strong structure match serves as a reliable indicator and is highly effective in estimating the relevance of a match.

To measure the common structure "size", intuitively, I choose the number of matched tree nodes to define the scoring function. Since the similarity contribution of different nodes (i.e. operands and operators) may be non-uniform, I propose the *multi-tree matching similarity* scoring function γ:

$$\gamma(\pi) = \sum_{i:(\hat{T}_q^i, \hat{T}_d^i) \in \pi} \beta_i \cdot \Big(\alpha \cdot \text{internals}(\hat{T}_d^i) + (1 - \alpha) \cdot \text{leaves}(\hat{T}_d^i)\Big) \tag{3.3}$$

where internals(T) is the number of internal nodes or operators in T, leaves(T) is the number of leaves or operands in T, and $\alpha \in [0, 1]$ adjusts the similarity contribution by operators and operands. Lastly, $\beta_i \geq 0$ are the contribution weights for different common formula subtrees.

For the convenience of later discussion, I assume the summation in Eq. (3.3) is performed from the widest common formula tree in descending order, indexed by i, i.e., $(\hat{T}_q^i, \hat{T}_d^i)$ is the *i-th widest* match (in terms of number of matched leaves) in π. And I set $\beta_1 \geq \beta_2 \geq ... \geq \beta_n$ heuristically to weight "wider" subexpressions higher.

In practice, it is wasteful to compute all terms in Eq. (3.3) if the largest M matched subexpressions cover most of the total matched size, therefore, the scoring function in Eq. (3.3) can be approximated by fixing $\beta_i = 0$ for $i > M$. Essentially, it becomes measuring top-n common formula subtrees with the greatest number of matched operands.

Considering multiple common subtrees and evaluating all types of nodes for all common trees can still be a computationally expensive process. A simplified scoring function is further introduced, it only measures the number of matched leaf nodes from the widest common tree, i.e., when $\alpha = 0, \beta_1 = 1, M = 1$. Formally, the *widest formula tree similarity*

$$w^* = \max_{(\hat{T}_q, \hat{T}_d) \in \pi} \text{leaves}(\hat{T}_d), \qquad \pi \in \Pi(T_q, T_d) \tag{3.4}$$

Presumably, different parts of the structure contribute differently to the similarity. For example, the left hand side of the equation "$i^2 = -1$" is a better signature than the right hand side *not* because it contains more symbols but because the squared power of i is a more rarely seen substructure, and as such, its presence in a document results in a greater reduction of the conditional information entropy [Robertson, 2004, Aizawa, 2003]. As a result, I further introduce a *path-weighted widest formula tree similarity*, i.e., $\bar{w}^*$, to make a distinction among different paths associated to different operands:

$$\bar{w}^* = \max_{(\hat{T}_q, \hat{T}_d) \in \pi} \sum_i \text{idf}_i, \qquad \pi \in \Pi(T_q, T_d) \tag{3.5}$$

where idf_i is the classic idf weight (see Section 3.1.2) associated with each leaf or operand indexed by $i = 1, 2, ..., \text{leaves}(\hat{T}_d)$.

3.1.2 Path Weighting

The rationale behind applying idf weighting to path-based structure matching is akin to its use in text search methodologies. By incorporating this weighting scheme, path-based structure search becomes highly compatible with existing retrieval models based on inverted indexes [Croft et al., 2010]. However, unlike the idf weight in text search where each token match is assigned an idf weight, in structure search, the summation of path

idf weights is associated with each structure match. In the following, I will present a few initial assumptions that can help justify the assignment of path weights.

If we were going to rank a structure match by a reciprocal probability of a *given* matched path set $\mathcal{P}$ being non-relevant (i.e., event $\bar{R}$), then

$$\frac{1}{P(\bar{R} \mid \mathcal{P})} = \frac{P(\mathcal{P})}{P(\mathcal{P} \mid \bar{R}) \cdot P(\bar{R})} \propto \frac{1}{P(\mathcal{P} \mid \bar{R})} \tag{3.6}$$

where the posterior is estimated by a proportional likelihood without affecting the ranking.

Let u_i be the probability of a path i occurring in a non-relevant maximum match $\bar{R}$, and assume u_i is independent and non-uniform among paths, then

$$\frac{1}{P(\bar{R} \mid \mathcal{P})} \propto \prod_{i \in \mathcal{P}} \frac{1}{u_i} \prod_{i \notin \mathcal{P}} \frac{1}{1-u_i} \tag{3.7}$$

$$= \prod_{i \in \mathcal{P}} \frac{1}{u_i} \prod_{i \in \mathcal{P}} \frac{1-u_i}{1} \prod_{i \in \mathcal{P}} \frac{1}{1-u_i} \prod_{i \notin \mathcal{P}} \frac{1}{1-u_i} \tag{3.8}$$

$$= \prod_{i \in \mathcal{P}} \frac{1}{u_i} \cdot \frac{1-u_i}{1} \prod_{i} \frac{1}{1-u_i} \tag{3.9}$$

$$\Rightarrow \log \frac{1}{P(\bar{R} \mid \mathcal{P})} \propto \sum_{i \in \mathcal{P}} \log \frac{1-u_i}{u_i} \tag{3.10}$$

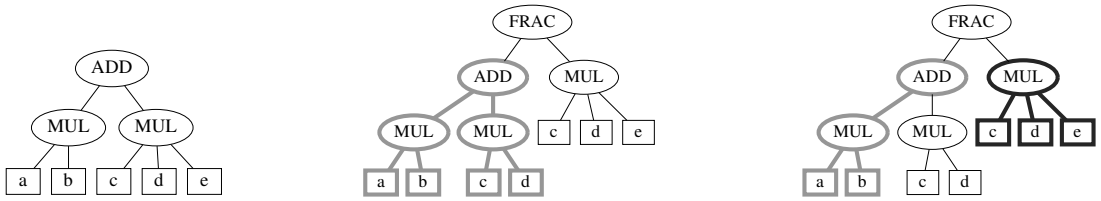
Because most of the paths are likely to occur in a non-relevant maximum match, it is reasonable to approximate $u_i = \mathrm{DF}_i / N$ where DF_i is the document frequency of path i, and N is the total number of paths in the index. As a result,

$$\log \frac{1}{P(\bar{R} \mid \mathcal{P})} \propto \sum_{i \in \mathcal{P}} \log \frac{N - \mathrm{DF}_i}{\mathrm{DF}_i} \approx \sum_{i \in \mathcal{P}} \log \frac{N}{\mathrm{DF}_i} = \sum_{i \in \mathcal{P}} \mathrm{idf}_i \tag{3.11}$$

Eq. 3.11 can be plugged into Eq. 3.5. Similar to text search, the idf weight of a path can be pre-computed during the construction of a search index. As a result, Eq. 3.5 can be viewed as a generalized form of the scoring function described in Eq. 3.4, while retaining the efficiency of only needing to calculate matched leaves.

3.1.3 Approximated Matching

The problems of determining the largest common sub-tree and computing the edit distance between unordered labelled trees are NP-hard [Zhang and Jiang, 1994]. There are other



(a) Query OPT. **(b)** Greedy match. **(c)** Optimal match.

Figure 3.3: An counterexample when the greedy widest match does not yield the defined formula tree similarity. OPT in (a) is the matching subject, OPTs in (b) and (c) show two different possible common formula subtrees highlighted in bold colors. The non-optimal match in (b) has fewer nodes matched but is found using the greedy widest matching by identifying the widest common tree as the first tree.

tree matching algorithms that are linear-time [Valiente, 2002] but would require unlabeled nodes and matching vertex out degree (i.e., matching the entire bottom-up subtree at some node), which are too restrictive for retrieval purpose, e.g., $x + y$ will not match $x + y + z$ due to out degree mismatch. To practically compute formula tree similarity, I propose a few assumptions that approximate the equality of a match for substructures using only leaf-root paths.

Assumption 1. If $\pi^* \in \Pi(T_q, T_d)$ is the maximizer of defined formula tree similarity given by Eq. 3.2 for $\alpha = 0$ and $\beta_1 \gg \beta_2 \gg ... \gg \beta_n$, then π^* is assumed to be also the maximizer for any $\alpha > 0$ and $\beta_1 \geq \beta_2 \geq ... \geq \beta_n$.

Under Assumption 1, it is hypothesized that finding multiple widest common formula subtrees greedily will yield the defined formula tree similarity, while this may not be the case in reality (see Figure 3.3).

On the other hand, I propose to use leaf-root paths from OPT as the basic search unit for efficient retrieval. There are two reasons: (1) In the OPT representation, "vertical" leaf-root paths are inherently immune to positional permutations in operands, this makes it naturally a good set of features for math formula similarity assessment even if the operands are under a commutative operator. (2) We can extract the prefixes of leaf-root paths to find subexpressions. See Figure 3.3 for instance, the `{c,d,e}/MUL` paths in (a) are prefixes of the leaf-root paths `{c,d,e}/MUL/FRAC` in (c), and they can be used for matching the subtree structure they represent without matching the entire original tree.

To maximize recall, two paths can match if they are identical after applying tokenization (or unification) on each of the symbols along the path (e.g., the OPTs representing $a + b$

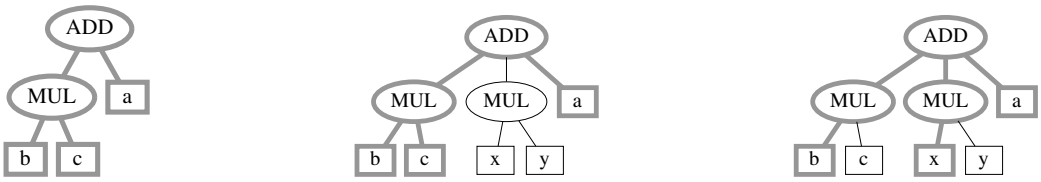


Figure 3.4: OPT representations for formula $a+bc$ shown in (a), and for formula $a+bc+xy$ shown in (b) and (c). Highlighted in bold grey are the same set of leaf-root paths in each tree, only in (a) and (b) are they corresponding to isomorphic substructures.

and $x + y$ have the same set of tokenized leaf-root path `VAR/ADD`, they can be matched regardless of having different symbols), and the symbol match is left to be performed after identifying a structure match (see Section 3.2.1).

To efficiently test for identical structures using only leaf-root paths, let $\mathcal{P}(T)$ denote the set of all leaf-root paths from a rooted tree T. Another assumption is made:

Assumption 2. For rooted trees T_q and T_d, let $S_q = \mathcal{P}(T_q), S_d = \mathcal{P}(T_d)$. If there exists perfect matching $M(S_q, S_d, E)$, then assume $T_q \cong T_d$,

where a matching between path sets S_1, S_2 is defined as bipartite graph $M(S_1, S_2, E)$ where E is the set of edges representing assigned matches. Assumption 2 hypothesizes that two subtrees are structurally identical if their leaf-root paths match, however, this is also not necessarily true (see Figure 3.4).

Under Assumptions 1 and 2, it can be shown that given a pair of OPTs T_q and T_d, if we can find $S_q^m, S_d^m = \arg\max |E|$ for any perfect matching $M(S_q^m \subseteq S_q, S_d^m \subseteq S_d, E)$ where $S_q = \mathcal{P}(\hat{T}_q), S_d = \mathcal{P}(\hat{T}_d)$ and $\hat{T}_q \preceq_l T_q, \hat{T}_q \subseteq T_q, \hat{T}_d \preceq_l T_d, \hat{T}_d \subseteq T_d$, then the widest match of $\pi^* \in \Pi(T_q, T_d)$ has $\text{leaves}(\hat{T}_q^1) = \text{leaves}(\hat{T}_d^1) = |S_q^m| = |S_d^m|$. In other words, matching greedily the leaf-root paths from query and document OPT subtrees can be used to get the number of leaves of the widest matched tree in a common formula forest π^* to obtain the defined formula tree similarity given by Eq. 3.2.

3.1.4 Multi-Tree Matching

In many cases we find multiple subtree matches are beneficial to assess formula structure similarity. As an example, consider the following two variations of the *Vandermonde*

identity that should be regarded as similar:

$$\sum_{k=0}^{r-1} \binom{n}{2k+1} \binom{n}{2r-2k-1} \quad \text{and} \quad \sum_{k=0}^{r} \binom{n}{2k} \binom{n}{2r-2k} \tag{3.12}$$

It is easy to observe that the multiple subexpressions highlighted in different colors are better capturing the similarity of these expressions than only considering the single largest matched subexpression.

We can obtain multiple matches recursively. To find out the next maximally matched subtree, assuming that leaves($\hat{T}_d^1$) is obtained, we can exclude already matched paths and similarly compute other leaves($\hat{T}_d^i$) for $i = 2, 3, ..., n$ specified by multi-tree matching similarity in Eq. 3.3. On the other hand, to get the number of operators associated with matched leaves, assume we have greedily found the set of leaves that belongs to the maximizer π^* for $\alpha = 0$ and $\beta_1 \gg \beta_2 \gg ... \gg \beta_n$, then under assumption 1, we can obtain the best multi-tree match in Eq. 3.2 by the following steps: first go through all subtree pairs (T_q^x, T_d^y) rooted at $x \in T_q, y \in T_d$, and examine if it joins with any pair of already matched trees in π^* by looking at whether their leaves intersect. If true, we will count x, y as additional matched operators.

Algorithm 1 proposed by Zhong and Zanibbi [2019] depicts in detail the multi-tree matching procedure described above. The algorithm of multi-tree matching first does greedy leaf matching in OPERANDMATCH, finding the k widest matched trees in terms of the number of leaves. Then it discovers the matched operators in OPERATORMATCH by examining if an internal node shares any leaf with the already matched ones. Note that only visible operators are counted, this is because some internal OPT nodes are abstract and do not appear in the rendered math expression (e.g., the `SUB` or `SUP` nodes), counting them will bias the similarity measurement in the model. Algorithm 1 avoids counting those nodes by consulting a pre-built "visibility" mapping for operators (i.e., the VISIBLE function).

Figure 3.5 illustrates an imaginary table that helps to understand Algorithm 1. Each path decomposed from their OPT representation is inserted into a corresponding table cell which groups a set of matching candidates. Each cell also represents an element of input list L in Algorithm 1. The highlighted cells in dark colors (blue and green) represent the widest matched trees. The highlighted cell in light color (light blue) represents the matched operator belonging to some already matched tree.

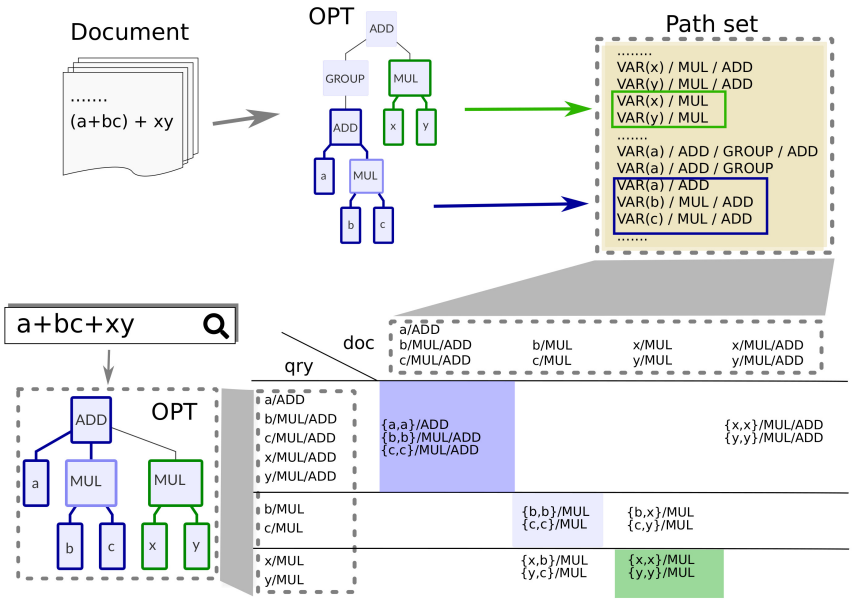


Figure 3.5: Matching multiple formula subtrees between the OPT representations of formulas $a+bc+xy+z$ and $(a+bc)+xy$ as shown in the left and right of the figure respectively. Path set consists of tokenized leaf-root paths with all prefixes.

Algorithm 1 Multi-tree matching for formula trees

Let (S_q^m, S_d^m) be the maximum matching path set of given path set (S_q, S_d).
Define $\ell(S)$ to be all the leaf nodes (equivalently, path IDs) for path set S.
function OPERANDMATCH(Q^m, D^m, L, k, leavesCounter)
  $Q^X := \{\ \}, D^X := \{\ \}$ ▷ Excluded path set
  for $i = 0, 1, ..., k$ **do**
    $Q^{\max}$, $D^{\max}$, max := 0 ▷ Best matched tree records
    for (S_q, S_d) from L **do**
      if $Q^X \cap\ \ell(S_q) = \emptyset$ and $D^X \cap\ \ell(S_d) = \emptyset$ **then** ▷ Disjoint tree pairs
        if $|S_q^m| >$ max **then** ▷ Greedily find widest matches
          max := $|S_q^m|$
          $Q^{\max}, D^{\max} := \ell(S_q^m),\ \ell(S_d^m)$
    if max > 0 **then**
      $Q^X, D^X := Q^X \cup\ Q^{\max}, D^X \cup\ D^{\max}$
      $Q_i^m, D_i^m := Q^{\max},\ D^{\max}$
      leavesCounter[i] = max
    else ▷ No more possible operand matchings
      break
  return Q^m, D^m, leavesCounter
function OPERATORMATCH(Q^m, D^m, L, k, operatorsCounter)
  Let Q^{map}, D^{map} be maps of matched internal nodes, initially empty.
  for (S_q, S_d) from L **do**
    for $i = 0, 1, ..., k$ **do**
      if $Q_i^m \cap\ \ell(S_q) \neq \emptyset$ and $D_i^m \cap\ \ell(S_d) \neq \emptyset$ **then** ▷ Joint tree pairs
        Let n_q, n_d be the root-end nodes of S_q, S_d respectively.
        if $Q^{\text{map}}[n_q], D^{\text{map}}[n_d]$ are both empty **then**
          $Q^{\text{map}}[n_q], D^{\text{map}}[n_d] := n_d, n_q$
          if visible(n_q) **then**
            operatorsCounter[i] := operatorsCounter[i] + 1
        break
  return operatorsCounter
function FORMULATREEMATCH(T_q, T_d, k)
  for $i = 0, 1, ..., k$ **do**
    $Q_i^m := \{\ \}, D_i^m := \{\ \}$ ▷ Matched path set for *i-th* largest matched tree
    leavesCounter[i] := 0
    operatorsCounter[i] := 0
  L := List of (S_q, S_d) where $S_q, S_d \in \mathcal{P}(T_q^x), \mathcal{P}(T_d^y)$ for each node $x \in T_q, y \in T_d$.
  Q^m, D^m, leavesCounter := OPERANDMATCH(Q^m, D^m, L, k, leavesCounter)
  operatorsCounter := OPERATORMATCH(Q^m, D^m, L, k, operatorsCounter)
  return leavesCounter[i], operatorsCounter[i] for $i = 1, 2, ...k$

3.2 Other Unsupervised Similarities

3.2.1 Symbol Similarity

Aside from structure similarity, math formula similarity is also reflected in symbols. For example, $E = mc^2$ and $y = ax^2$ have the same structure representation but they differ greatly in semantics due to the symbol mismatch. Intuitively, an exact match of symbols should be prioritized, however, we need to allow symbol substitution as well, e.g., formulas $x + y + y^2$ and $y + x + x^2$ are considered "nearly identical" due to their symbol equivalence after substitutions (this is also called α-*equivalence*, see Kohlhase and Sucan [2006]).

To address symbol similarity, I have defined a list of axiomatic rules to rank formulas heuristically. Let the unique instances of each symbol in a formula be named *bound variables*, for example, there are 3 bound variables x, y and 2 in the example formula of $x + y + y^2$. Now, assume we have already obtained the structural matching results for all bound variables, the axiomatic rules require:

1. **Exact match exceeds non-exact match**: Exact symbol match of bound variables exceeds the match of different symbols if their number of matches are the same. For example, the similarity between x and x exceeds that of y and x.

2. **Uniform similarity for different symbols**: The match of bound variables with different symbols should have equal similarity if their number of matches are the same. For example, the similarity between y and x is equally similar to a and b.

3. **Bound variable greediness**: If the number of matched variables are not the same, the match with greater bound variable cardinality should exceed the match of a smaller cardinality. For example, $x + x$ and $y + y$ are considered more similar than $x + x$ and $x + z$ even if the latter one has an exact match for letter x.

4. **Compositionality**: The overall symbol similarity is composed of its optimal individual bound variable matches. For example, we can greedily accumulate the similarity (sub)scores between the matches for different bound variables to obtain the overall formula similarity.

Algorithm 2 is used to apply the rules listed above. These rules are heuristic because the symbol similarity are inherently subjective; however, an axiomatic approach allows an algorithm to assess similarity between math formulas in a more principled way.

Algorithm 2 Mark and Cross (Hashmap implementation)

Input: The established path match m.
Output: Symbol similarity score (allowing symbol substitutions).

```
 1: function MARK(m)
 2:     bound_scores := an empty hashmap with zero as default value.
 3:     for each query and document path pair (q, d) in m do
 4:         δ := 1 if q, d match completely, b_1 if operands match, otherwise b_2 where 0 < b_2 < b_1.
 5:         s_q, s_d := leaf symbol of q and d, respectively.
 6:         bound_scores[s_q][s_d] := bound_scores[s_q][s_d] + δ
 7:     return bound_scores
 8:
 9: function CROSS(bound_scores)
10:     score := 0                                          ▷ Symbol similarity score.
11:     cross := an empty hashmap          ▷ Excluded (already matched) document variables.
12:     for each s_q in bound_scores do          ▷ In desc. order of query bound variable sizes.
13:         max_subscore, max_s_d := 0, ∅
14:         for each s_d in bound_scores[s_q] do
15:             if cross[s_d] then
16:                 continue
17:             if max_subscore < bound_scores[s_q][s_d] then
18:                 max_subscore := bound_scores[s_q][s_d]
19:                 max_s_d := s_d
20:         if max_subscore = 0 and any path of s_q requires exact match then
21:             return 0          ▷ Abort early if specified exact match cannot be fulfilled.
22:         else if max_s_d ≠ ∅ then
23:             cross[max_s_d] := true
24:             score := score + max_subscore
25:     return score
26:
27: function SYMBOLSIMILARITY(m)
28:     bound_scores := MARK(m)
29:     return CROSS(bound_scores)
```

Aside from leaf symbols, the *fingerprint* of each path is also compared. We will assign the highest path-match weight if both values agree, a medium weight if leaf symbols match but the fingerprints do not, and a lower path-match score if otherwise. The fingerprint captures a local symbol appearance for operators on a path, it can be used to differentiate formulas of the same structure but with different math operator(s). The fingerprint is generated for each path by computing the Fowler-Noll-Vo HASH [Noll, 2023] from the leaf node up to a number of operator nodes above.

The process of Algorithm 2 is illustrated in Table 3.1. The algorithm has two phases, a MARK phase that associates the match scores for individual path after their structurally matched candidates have been obtained, and a CROSS phase to greedily accumulate scores from larger bound variable grids and to exclude them from being matched further. More specifically, paths in the common structure are first paired by their symbols, then, the operand symbol and other similarity features associated with each path will determine the symbol score by summing the partial match scores δ from bound variable grids with the highest scores to the lowest, excluding previously matched bound variables from the document. Since the MARK phase can be accelerated by indexing bound variables, Algorithm 2 has a time complexity of $O(B_q \cdot B_d)$ where B_q and B_d are the number of bound variables from query and candidate formulas respectively.

In Algorithm 2, if neither the leaf symbol nor any feature matches, a base score b is assigned (i.e., b_1 or b_2, depending on their symbol match degree) to count for structure match. The first heuristic rule implies $b < 1$; and due to the 3rd rule, we have $k \times b > (k-1) \times 1$ for $k \in \mathbb{N}^+$. Accordingly, the base scores are chosen to be non-zeros or closed to 1. The base scores b_1 and b_2 are given as hyperparameters in Algorithm 2.

3.2.2 Context Lexical Similarity

Math document similarity, despite potentially relying on key formulas, cannot disregard the presence of textual content. To incorporate context awareness into math document retrieval, I propose to utilize unsupervised scoring functions from traditional IR.

Unsupervised lexical similarity has a long list of options,[1] but traditional fulltext search commonly adopts tf–idf or its variants [Kamphuis et al., 2020] because they incorporate global word distribution and tend to perform robustly against many datasets. Among them, the BM25+ scoring [Lv and Zhai, 2011] stands out in effectiveness for long documents. For

[1] See this Python package README for a list of well-summarized text similarity measurements: https://pypi.org/project/textdistance

Table 3.1: Illustration of the Mark and Cross scoring (Algorithm 2) for formulas $x + y + y^2$ and $-y + x + x^2$. The set of leaf-root paths for $x + y + y^2$ are x, y: `VAR/ADD`; y: `VAR/BASE/SUBSUP/ADD`; 2: `NUM/SUP/SUBSUP/ADD`. And the set of leaf-root paths for $-y+x+x^2$ are y, x: `VAR/ADD`; x: `VAR/BASE/SUBSUP/ADD`; 2: `NUM/SUP/SUBSUP/ADD`. Bound variables (variables of the same symbol) are separated by double borders. For better visualization, the box notation is meant to give some structure context. Matching candidates in each cell are *marked* by their associated partial similarity scores, and the committed matches are *crossed* greedily top down, highlighted in grey. In this example, $b_1 = 0.9$ and $b_2 = 0.8$. The (unnormalized) symbol similarity score is $(0.8 + 0.8) + 0.8 + 1.0 = 3.4$.

Bound Var.	x		y	2
	x	$\boxed{x}^2$	$-y$	$x^{\boxed{2}}$
$\boxed{y}^2$		0.8		
y	0.8		0.9	
x	1.0		0.8	
$y^{\boxed{2}}$				1.0

BM25 variances (Section 2.1.1), BM25+ is often used for math formula context similarity scoring. More specifically, the BM25+ scoring function is defined as:

$$BM25^{+}(q,d) = \sum_{t \in q} \log\left(1 + \frac{N - df_t + 0.5}{df_t + 0.5}\right) \cdot \frac{tf_{t,d}}{tf_{t,d} + k_1(1 - b + b(L_d/L_{avg}))} \tag{3.13}$$

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

3.2.3 Overall Similarity

A scalar value representing the overall similarity score is needed to determine the priority of a search result when returning it to a user in an ad-hoc retrieval. To combine different similarities, relevant score factors are multiplied by the structure similarity score $\gamma(\pi^*)$ (see Section 3.1.1), after obtaining the best-matched substructure(s) π^*.

First, the symbol similarity (see Section 3.2.1) conditioned on the matched structure π is normalized to S'_{sym}. Then, for query formula Q and document formula D, the final symbol score factor S_{sym} is a further rescaled version that contributes more distinctions for formulas of similar S'_{sym} scores:

$$S_{sym}(Q, D \mid \pi) = \frac{1}{1 + (1 - S'_{sym})^2} \tag{3.14}$$

Second, a penalty factor is introduced to penalise long formulas in a document. Assume the original formula length is L_D, how much the penalty $P(D;\eta)$ will be assigned to an inverted log rescaler is determined by a hyperparameter $\eta \in [0, 1]$:

$$P(D;\eta) = 1 - \eta + \eta \cdot \frac{1}{\log(1 + L_D)}. \tag{3.15}$$

Finally, the overall score for math formula similarity $S(Q, D)$ is given by

$$S(Q, D) = \gamma(\pi^*) \cdot S_{sym}(Q, D \mid \pi^*) \cdot P(D;\eta) \tag{3.16}$$

where $\pi^* = \text{argmax}_\pi\, \gamma(\pi)$ is calculated by performing structure matching, as described in Sections 3.1.3 and 3.1.4.

There is an old form used in Zhong and Zanibbi [2019] where we apply F-score – or equivalently, harmonic mean – onto structure and symbol similarities:

$$S^F(Q, D) = \text{F-Score}\left(\frac{\gamma(\pi^*)}{|\text{leaves}(T_q)|},\ S_{sym}(Q, D \mid \pi^*)\right) \cdot P(D; \eta). \tag{3.17}$$

Compared to the F-score form, the product form is more efficient and similar to tf–idf – if the structure similarity is represented by the path-weighted widest formula tree defined in Eq 3.5, it resembles the tf–idf scoring except that the "term frequency" here counts for common structure "width", i.e., the number of matched leaves in the common subtrees. Inspired by this insight, it is natural to combine tf–idf and structure search scores in a single and uniform weighted summation.

As a result, the final score consists of the tf–idf score for text and scores for each math formula similarity defined by Eq 3.16, weighted by a *math path weight* scaler λ_m [Zhong et al., 2021], when using BM25+,

$$S_{\text{all}}(Q, D) = \lambda_m \cdot S(Q_{\text{math}}, D_{\text{math}}) + BM25^+(Q_{\text{text}}, D_{\text{text}}) \tag{3.18}$$

where $Q_{\text{text}}, D_{\text{text}}$ are query and document text tokens extracted from query Q and document D. Similar meaning applies to $Q_{\text{math}}, D_{\text{math}}$ except the math tokens are isolated formulas (e.g., in LaTeX representations). The above pipelines unleash an unsupervised math-aware retrieval model that we name Approach Zero. [2]

3.3 Evaluation

3.3.1 Experimental Setup

For formula search evaluation, I consider MIR tasks including the NTCIR-12 WMB [Zanibbi et al., 2016a] and the ARQMath-3 Task 2 benchmark [Mansouri et al., 2022].

For the NTCIR-12 dataset, with approximately only 600,000 formulas available, it is practical to perform a comprehensive evaluation of the multi-tree matching. We consider all 20 concept queries in this dataset. Unless specified in Section 3.3.3, the formula length penalty weight, denoted as η, is set to 0.05.

[2] *Approach Zero* or *approach0* gets its name from the math vocabulary "asymptotics", the core concept to define a limit.

In the ARQMath dataset, with tens of millions of formulas, only the widest formula tree is evaluated due to efficiency constraints. Additionally, to ensure proper parsing of LaTeX markup into the expected OPT, 20 of the query formulas are manually refined to adhere to conventional mathematical expression notations. For example, we inserted a comma in B.336 query formula to separate two conditions in a set expression (where the original expression uses a long space for separation). Without manual corrections, it would be challenging to automatically suggest these changes to ensure that every user-generated formula can be converted into a clear OPT representation for effective retrieval. The formula length penalty η is set to 0.3 in this dataset.

In Eq. 3.18, the math path weight λ_m is set to 2.5 and the default BM25+ hyper parameters $b = 0.75, k_1 = 2.0$. Unless specified otherwise, an optimal set of parameters where $\beta_1 = 0.9, \beta_2 = 0.06, \beta_3 = 0.04, \alpha = 0.4$ is applied for Eq. 3.3 [Zhong and Zanibbi, 2019]. Additionally, the symbol score thresholds b_1, b_2 in Algorithm 2 are set to 0.94 and 0.9 (the choice of these parametric values are explained in Section 3.3.3). When evaluating parameters independent of the number of matched trees, experiments were performed using a single-tree matching setting.

3.3.2 Main Results

The formula search evaluation results on NTCIR-12 and ARQMath-3 are summarized in Table 3.2 and 3.3 respectively (systems highlighted in bold font refer to the Approach Zero system discussed in this thesis).

We can see the proposed unsupervised formula search is competitive on the NTCIR-12 dataset; the evaluation scores are even comparable to those of learned deep models (e.g., MathBERT [Peng et al., 2021] and ColBERT [Zhong et al., 2022b]). In addition, the current state-of-the-art system, i.e., MathAPP, is an ensemble system combining our run to generate search results. It is noteworthy that MathAPP, when incorporating our structure search run into its baseline Transformer-based retriever MathBERT, achieves a 15% improvement in the effectiveness of fully relevant search results. This observation suggests that the proposed structure search model is highly effective in identifying fully relevant formulas. It also highlights the significance of structure similarity as a crucial prior relevance signal when evaluating the semantic similarity of formulas.

On the larger-scale ARQMath dataset, where running expensive models efficiently is impractical, our more efficient single-tree matching approach surpasses other state-of-the-art models. Specifically, the proposed structure matching model achieves a 12% higher precision in the top search results compared to another effective unsupervised retriever,

Table 3.2: Effectiveness evaluation in the NTCIR-12 Wiki Formula Browsing Task. Our unsupervised retriever is compared to the most effective existing systems. The systems can be categorized into three groups: expensive, cost-effective, and ensemble.

	System	Part. Rel.	Fully Rel.	Note
Expensive	(1) MCAT (Kristianto et al. [2016])	0.57	0.57	Avg run time $\geq$ 25 secs
	(2) Tangent-S (Davila and Zanibbi [2017])	0.59	0.64	High variance in run times
	(3) Our ColBERT (Zhong et al. [2022b])	0.48	0.55	High resource demand
Cost-effective	(4) Our DPR (Zhong et al. [2022b])	0.43	0.52	Transformer bi-encoder
	(5) GNN (Song and Chen [2021])	0.54	0.63	Using GNN
	(6) Tangent-CFT (Mansouri et al. [2019a])	**0.71**	0.60	FastText Embedding
	(7) **Ours (Zhong et al. [2020])**	0.54	0.63	Using **single-tree** match
	(8) **Ours (Zhong and Zanibbi [2019])**	0.60	**0.67**	Using **3-tree** match
Ensemble	(9) TanAPP (Mansouri et al. [2019a])	0.73	0.70	Tangent-CFT + Ours
	(10) MathBERT (Peng et al. [2021])	0.74	0.61	2-layer BERT embeddings
	(11) MathAPP (Peng et al. [2021])	**0.76**	**0.72**	M.BERT + T.-CFT + Ours

Table 3.3: Effectiveness evaluation in the ARQMath-3 (2022) Formula Search Task (Task 2). Our retriever is evaluated against the top-performing results from other participant teams where the Tangent-CFTED system utilizes a supervised model with tree distance reranking. The Judged metrics measures the percentage of judged hits.

Runs	ARQMath-3 Task 2				
	NDCG'	MAP'	P'@10	bpref	Judged
Most effective systems					
(1) Tangent-CFTED [Mansouri et al., 2020b]	**0.694**	0.480	0.611	0.471	61.7
(2) MathDowsers [Kane et al., 2022] †	0.640	0.451	0.549	0.443	60.3
Ours using single-tree matching					
(3) **Ours [Zhong et al., 2022a] †**	0.639	**0.501**	**0.615**	**0.505**	45.9

"†": using unsupervised method.

MathDowsers [Kane et al., 2022]. It is worth noting that while MathDowsers also employs a structure matching approach, it extracts over five types of features from SLT paths. On the other hand, Approach Zero explicitly models structure similarity from the matched OPT leaf-root paths, requiring to index tokens only proportional to the number of operands.

Our proposed structure match is also on par with Tangent-CFTED [Mansouri et al., 2020b], an upgraded version of Tangent-CFT [Mansouri et al., 2019a]. Tangent-CFTED uses tree distance to rerank results retrieved from *learned* structure embeddings. The evaluation on the ARQMath benchmark demonstrates that our single-tree matching alone can obtain effective formula search results comparable to learned and more complex models. The efficiency of a single-tree matching structure search model enables its application on real-world scale datasets. Moreover, there is significant potential for further improvement in efficiency in single-tree matching, as discussed in Chapter 4.

3.3.3 Ablation Study

Number of matched trees and types of matched nodes First, we have explored the impact of matching different number of substructures. Figure 3.6 shows representative parameter values that we have tried. We started with a single tree match (rows 1–4), finding that weighting operator symbol matches slightly lower than operands ($\alpha = 0.4$) have produced the best results (we tried α in $[0, 1]$ using an increment of 0.1). We then fixed $\alpha = 0.4$, $\sum_i^n \beta_i = 1$, and tried uniform weights (rows 5–7) and non-uniform weights for two trees (rows 8–10) and three trees (rows 11–13). We examined uniform β weights for multiple matches up to 3 trees. For non-uniform weights in two-trees, we consider β_1 in $[0.5, 0.99]$ using increments of 0.05 or 0.01; for three-tree matching, we considered β_1 in $[0.5, 0.95]$ and β_2 in $[0.05, 0.45]$ using increments of 0.05.

Results in Figure 3.6 shows uniform weights generally yield worse results than non-uniform ones. And two runs from non-uniform weights obtain the best partial and full relevance scores respectively. This observation is intuitive because our setting of non-uniform weights emphasizes larger subexpressions, which arguably have more visual impact. However, even if multi-tree matching does improve effectiveness, the retrieval based on single tree matching (even with only operand matching) can also generate competitive fully relevance results with minimal computational overhead.

Second, to illustrate the effect of matching multiple subexpressions, Figure 3.7 shows changes in fully relevant bpref scores for different queries, when changing the maximum number of matched trees with uniform weights. For space, Figure 3.7 omits queries whose score remains unchanged or differs negligibly across the maximum number of common trees

Run	Parameters			
	α	β_1	β_2	β_3
(1) opt-only	1.0	1.00		
(2) opd-opt-a6	0.6	1.00		
(3) opd-opt-a4	0.4	1.00		
(4) opd-only	0.0	1.00		
(5) uni-beta-1	0.4	1.00		
(6) uni-beta-2	0.4	0.50	0.50	
(7) uni-beta-3	0.4	0.34	0.33	0.33
(8) 2-beta-98	0.4	0.98	0.02	
(9) 2-beta-80	0.4	0.80	0.20	
(10) 2-beta-60	0.4	0.60	0.40	
(11) 3-beta-90-4	0.4	0.90	0.06	0.04
(12) 3-beta-75-2	0.4	0.75	0.15	0.10
(13) 3-beta-60-3	0.4	0.60	0.25	0.15

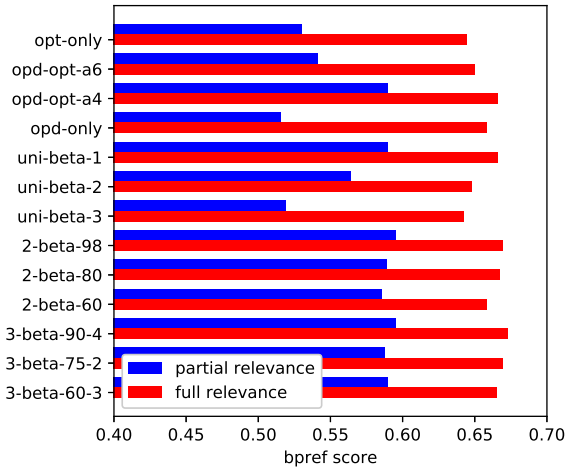


Figure 3.6: Evaluation results on the NTCIR-12 Wiki Formula Browsing Task from different number of maximum matching subtrees and the α parameter (table and bar graph).

being considered. We observe that different queries have different behaviours as the number of considered trees increases, e.g., introducing secondary matching into queries 4 and 6 improves results, while multi-tree matching hurts performance noticeably in queries 16, 18 and 20. To explain this, a further case study is conducted. Looking at the previously mentioned query examples in Figure 3.8, we can observe that due to the differences in their structural complexity, extracting partial components in queries 4 and 6 produces better similarity than matching partial components in more complex queries (e.g query 16 and 18). In other words, being able to be decomposed into small parts without losing too much informative structure will make queries (such as 4 and 6) benefit from multiple-tree scoring. On the other hand, queries 16 and 18 perform better using a single tree because their structure as a whole is essential to identify them.

Symbol similarity and path idf weighting In addition to evaluating structure similarity through tree matching, I have also examined the effects of enabling or disabling symbol similarity scoring and path idf weighting. These two enhancements can be viewed as re-weighting factors applied to the structurally matching paths. It is important to note that if these re-weightings are disabled, the retrieval process will revert to solely relying on structure matching, where the maximum size of common structures is sought. The ablations of these two enhancements for structure search are presented in Table 3.4.

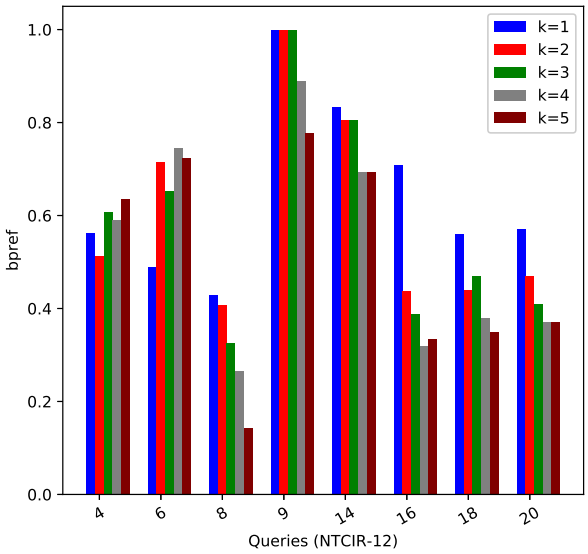


Figure 3.7: Full relevance bpref scores for matching uniformly-weighted subtrees (1 to 5 trees) on the NTCIR-12 WMB dataset.

No.	Query Formula
4)	$\nabla \times \mathbf{B} = \mu_0 \mathbf{J} + \underbrace{\mu_0 \epsilon_0 \frac{\partial}{\partial t} \mathbf{E}}_{\text{Maxwell's term}}$
6)	${}^{238}_{92}\mathrm{U} + {}^{64}_{28}\mathrm{Ni} \rightarrow {}^{302}_{120}\mathrm{Ubn}^{*} \rightarrow \ \ldots$
16)	$\tau_{\mathrm{rms}} = \sqrt{\frac{\int_0^\infty (\tau - \overline{\tau})^2 A_c(\tau) d\tau}{\int_0^\infty A_c(\tau) d\tau}}$
18)	$P_i^x = \frac{N!}{n_x!(N-n_x)!} p_x^{n_x} (1-p_x)^{N-n_x}$

Figure 3.8: Example queries in Fig 3.7.

Table 3.4: Ablations on the symbol scoring and path idf, evaluated on the ARQMath-3 Task 2 data. Enabling both scoring is more effective compared to ablated versions except for a minor drop in the precision score when path idf is disabled.

Run	NDCG'	MAP'	P@10	bpref
Full scoring	**0.6394**	**0.5007**	0.6158	**0.5052**
Disabled symbol match	0.5335	0.4060	0.5667	0.4237
Disabled path idf	0.6260	0.4864	**0.6197**	0.4872
All disabled	0.4858	0.3571	0.5573	0.3787

We can see both symbol similarity scoring and path idf weighting have a notable boost to the effectiveness of formula search. The greatest improvement is from enabling symbol similarity awareness where around 18 % to 19 % advancements are observed in NDCG' and MAP'. This observation is intuitive because math symbols, specifically leaf nodes or operands, contribute to similarity in a distinct dimension compared to structure similarity. The path idf weighting does not improve top precision but there are 5% gains on MAP' and bpref due to the ability to further distinguish path contributions to the structure matches. Interestingly, when both re-weighting enhancements are disabled (only performing uniform weight tree matching), we see more obvious effectiveness degradation which means only using structure matching is suboptimal to capture formula semantic similarities.

Symbol similarity base scores Given the significance of symbol similarity scoring, I proceeded to conduct further evaluations using a grid search of the base score weights b_1 and b_2 ($b_1 > b_2$). These weights represent the partial scores obtained from (1) matching both the path fingerprint and operand symbol, or (2) matching only the operand symbol, as outlined in Algorithm 2. The result is shown in Figure 3.9.

The NDCG' metric captures the overall effectiveness, taking into account formulas with partially matched symbols. On the other hand, P@10 (precision at 10) gives more weight to highly relevant hits in the top results. It is worth noting that these two metrics exhibit different trends in Figure 3.9, when the symbol score weights are varied. The overall effectiveness reaches its peak when there is a large difference between the operand symbol match score (b_1) and the pure structure score (b_2). This result emphasizes the importance of symbol match over structure agreement. However, highly relevant results tend to peak when both the structure and operand symbol are heavily weighted (with b_1 and b_2 both close to 1). This observation is expected because a highly ranked result is likely to be a perfect match.

In essence, a structure match without a symbol match is not an effective indicator of relevance for lower-ranked results. This is likely due to the presence of more structural noise in partially relevant results. For example, the query "$E = mc^2$" would not be considered relevant to "$y = ax^2 + bx + c$" unless there is symbol agreement. Such cases are more likely to occur in partially matched candidates, typically ranked lower in the search results.

Formula length penalty and math path weight The formula length penalty in Eq. 3.15 and math path weight in Eq. 3.18 are another set of hyperparameters to be studied. Their impact on effectiveness is plotted separately in Figure 3.10.

Figure 3.10 demonstrates that both hyperparameters exhibit a saturated impact when their values become large. The evaluation results, reported across four metrics, consistently support this observation. The plot associated to formula length penalty (η) looks

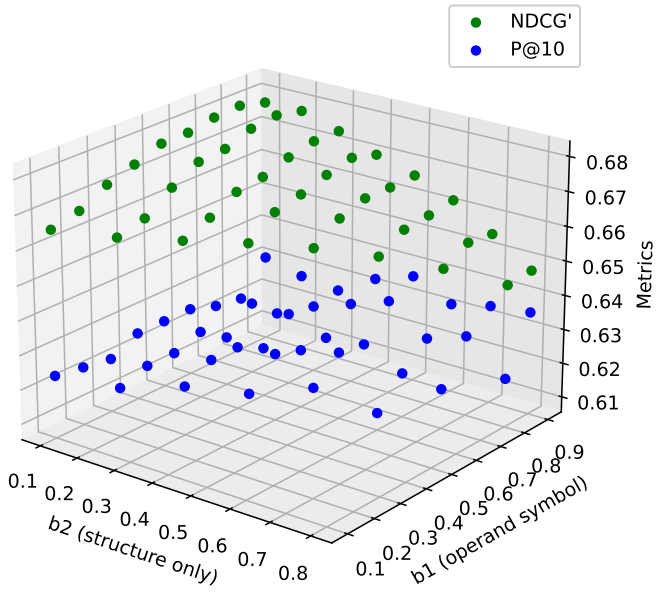


Figure 3.9: The effectiveness (in terms of NDCG' and P@10 scores) using different symbol base scores b_1 and b_2 where $0.1 \leq b_1 \leq b_2 \leq 0.9$.

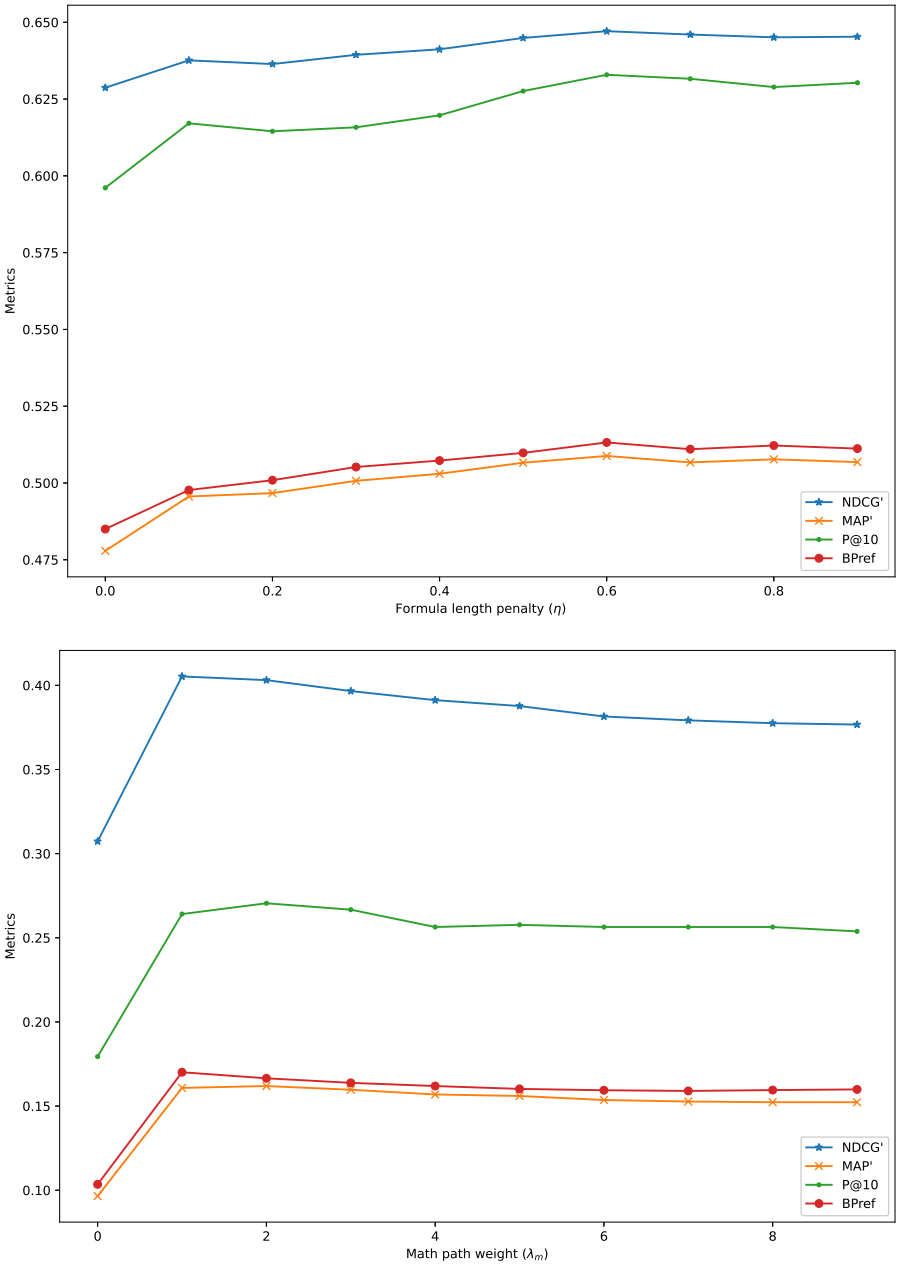


Figure 3.10: The impact on effectiveness using different formula length penalty (η) and math path weight (λ_m) parameters. Results are evaluated on the ARQMath-3 Task 2 and Task 1 datasets, respectively.

more monotonically increasing which means a long formula in a document with a length mismatch to the query should be penalised as much as possible, which creates very precise and perfect matches. But this penalty does not affect further after $\eta = 0.6$.

On the other hand, the effectiveness of formula and text retrieval tops at a small math path weight when $\lambda_m \in [1, 2]$, meaning that the math keywords should be treated equally or a little more importantly than text keywords. Note that when $\lambda_m = 0$, the retrieval deteriorates to text-only retrieval without the ability to identify any similar formulas in a document, therefore, the sharp increase from the effectiveness when $\lambda_m = 0$ to when λ_m reaches its optima in the bottom plot of Figure 3.10 suggests that the math awareness can boost math document retrieval effectiveness to a scale of more than 30% (as the improvement shown in NDCG'). This is intuitive to understand because in many math questions (e.g., questions looking for a proof of a calculus equation), the only informative tokens are math formulas.

Chapter 4

Efficient Structure Search

Traditional information retrieval (IR) systems typically handle millions of documents per thread efficiently. However, running a naive structure search using the methods described in Chapter 3 against such a large number of documents would be time-consuming, taking more than ten seconds per query even with single-tree matching [Zhong et al., 2020]. In this Chapter, I will review our work proposed in Zhong et al. [2020] for accelerating structure search query execution to sub-second latency, which can be integrated with traditional IR models with little overhead.

4.1 Rank-Safe Dynamic Pruning

4.1.1 Intuition and Background

A matching algorithm for multiple common structures can be expensive for the purpose of math similarity search. An intuitive first step to speed up retrieval is to consider only scoring structure similarity for the widest common subtree and counting its operands. As a result, the scoring function can be simplified – the resulting score between query and document OPTs T_q, T_d only depends on the widest common tree. Interestingly, in Section 3.3.2, I have shown that using single-tree matching performs similarly to multi-tree match in terms of effectiveness, with only a 6% drop in bpref score for full relevance when compared to multi-tree matching, which achieves the best unsupervised scores on the NTCIR-12 Wiki Formula Browsing Task.

The simplified scoring function further leads to the application of *dynamic pruning* techniques (see Section 2.1.2) for search space reduction, because obtaining a tight upperbound for the scoring function in Eq. 3.4 is feasible. Obviously, the number of paths in a OPT formula subtree can be conveniently converted to its structure similarity upperbound. Using a tight upperbound, the dynamic pruning algorithm can avoid scoring unnecessary candidates effectively as more candidates are likely to be pruned if their upperbound score is less than the minimal score in the top-k candidates.

Dynamic pruning and index compression power modern search engines, enabling them to achieve remarkable efficiency. Although applying index compression for structure retrieval remains straightforward and very similar to regular inverted list compression, dynamic pruning cannot be directly adapted for structure search. The modeling of structure search differs from the MaxScore dynamic pruning algorithm in that the upperbound scores are also derived from the query tree representation, rather than being naturally assigned to the posting lists individually. Moreover, only a subset of matched paths that are part of the widest common subtrees contribute to the overall score. In contrast, in typical tf-idf scoring, all hit terms contribute to the ranking score.

This chapter proposes a dynamic pruning method for pruning unpromising candidates for a structure query. As structure search typically involves longer queries [Zhong et al., 2023], we adopt a MaxScore-like strategy as suggested by Shan et al. [2012], Antonio Mallia and Suel [2019]. Unlike WAND variants, this strategy avoids the need to sort query terms during merge iterations, which can be costly for long queries. Furthermore, I introduce additional methods to perform unsafe pruning in structure search, boosting structure search efficiency even further.

4.1.2 Definitions

During query processing, a series of query paths will match with paths from a document expression one by one. At each step, a *hit path set*, which includes the matched query paths and the corresponding paths from the document, is examined. We model this path set by a bipartite graph $G(Q, D, E)$ where $Q = \{q : q \in \mathcal{P}(T_q^m), m \in T_q\}$[1] and $D = \{d : d \in \mathcal{P}(T_d^n), n \in T_d\}$ are query and document path sets respectively, and edge sets are ordered pairs $E = \{(q, d) : \text{tokenized}(q) = \text{tokenized}(d), q \in Q, d \in D\}$ representing a potential matching between a query path and a document path.

When paths share the same token sequence, an edge is formed. This allows us to partition the graph into smaller disconnected bipartite graphs denoted as $G_t = G(Q_t, D_t, E_t)$.

[1]Recall that T^m denotes subtree rooted at node m.

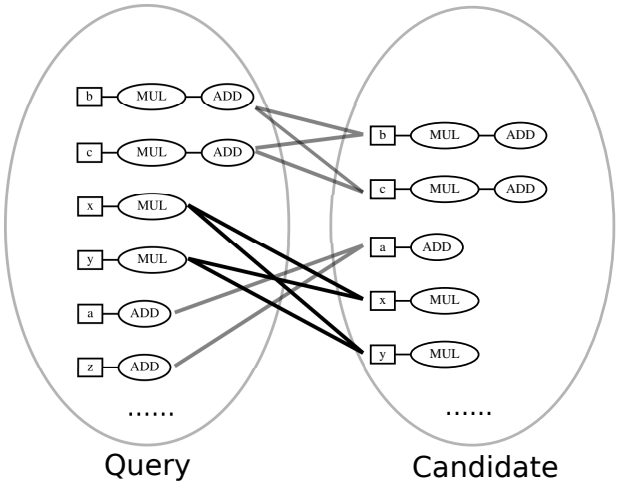


Figure 4.1: Bipartite graph of the (partial) path set for formulas in Figure 3.2 (original leaf symbol is used here to help identify paths). Edges are established if paths from the two sides are the same after tokenization. Edges with shared end points (i.e., same root-end nodes) in original OPTs are highlighted with the same color.

Each graph is fully connected and is identified by a tokenized query path t. Formally, define a sub-bipartite graph that is filtered by path t:

$$Q_t = \{q : q \in Q, \text{tokenized}(q) = t\} \tag{4.1}$$
$$D_t = \{d : d \in D, \text{tokenized}(d) = t\} \tag{4.2}$$
$$E_t = \{(q, d) : (q, d) \in E, \text{tokenized}(q) = \text{tokenized}(d) = t\} \tag{4.3}$$

Figure 4.1 shows an example bipartite graph for a hit path set, this example can be partitioned into disconnected smaller bipartite graphs, associated with tokenized path `VAR/MUL/ADD`, `VAR/MUL` and `VAR/ADD` respectively. Each partition is actually a complete (fully connected) bipartite graph because $\forall e \in (Q_t, D_t)$, we have $e \in E_t$. And for any complete bipartite graph $G(V_1, V_2, E)$, we can easily obtain their *maximum matching* by computing $\min(|V_1|, |V_2|)$.

On the other hand, to calculate the widest formula tree similarity $w^*_{Q,D}$ based on single-tree matching (see Eq. 3.4 or 3.5), we need to find a pair of query and document nodes at which the optima $\hat{T}^*_q, \hat{T}^*_d$ are rooted such that they represent the maximum possible common formula subtree (see Definition 3.1.2). Therefore, we need to also define the matching candidate relations filtered by nodes. Let $G^{(m,n)} = G(Q^{(m)}, D^{(n)}, E^{(m,n)})$ be the

subgraph that only considers matching between query node n and document node m:

$$Q^{(m)} = \{q : q \in Q, \text{root\_end}(q) = m\} \tag{4.4}$$

$$D^{(n)} = \{d : d \in D, \text{root\_end}(d) = n\} \tag{4.5}$$

$$E^{(m,n)} = \{(q,d) : (q,d) \in E, \text{root\_end}(q) = m, \text{root\_end}(d) = n\} \tag{4.6}$$

where the edges in each $E^{(m,n)}$ set correspond to a uniquely colored edge in Figure 4.1.

To find the maximum matched subtree, we can calculate the score of each pair of nodes independently by summing its maximum matching value among partitions, the pair of nodes with the greatest score is essentially our similarity score $w^*_{Q,D}$. Specifically, define *token paths* of T^n as set $\mathfrak{T}(n) = \{t : t = \text{tokenized}(p), p \in \mathcal{P}(T^n)\}$, it follows

$$w^*_{Q,D} = \max_{m \in T_q, n \in T_d} \nu(G^{(m,n)}) \tag{4.7}$$

$$= \max_{m \in T_q, n \in T_d} \sum_{t \in \mathfrak{T}(m)} \nu(G^{(m,n)}_t) \tag{4.8}$$

$$= \max_{m \in T_q, n \in T_d} \sum_{t \in \mathfrak{T}(m)} \min(|Q^{(m)}_t|, |D^{(n)}_t|) \tag{4.9}$$

where $\nu(G)$ is the maximum matching size of bipartite graph G.

Denote $w_{m,t} = |Q^{(m)}_t|$, we call $w_{m,t} \geq \min(|Q^{(m)}_t|, |D^{(n)}_t|)$ as our (pre-computed) partial score upperbound. It is analogous to text search where each posting list has a partial score upperbound, the tf–idf score upperbound is merely their sum. However, in our case, the sum for partial score upperbounds is only for one node or a subtree.

In the following, we propose three strategies to compute $w^*_{Q,D}$ upperbound from partial score upperbounds and to assign posting lists to the non-requirement set.

4.1.3 MaxRef Pruning Strategy

In MaxScore, each posting list can be mapped to a partial score upperbound directly, however, our scoring function implies each posting list can be involved with multiple partial score upperbounds. One simple way to select the non-requirement set in our case is to provide an upperbound score *MaxRef* (for each posting list t) which is the maximum partial score from the query nodes by which this posting list gets "referenced", and if a set of posting lists alone has a sum of MaxRef scores less or equal to θ, they can be safely put into the non-requirement set.

The rank safeness can be justified, since each posting list corresponds to a unique tokenized path t, define $\text{MaxRef}_t = \max_m w_{m,t}$, then $\forall\ m \in T_q, n \in T_d$,

$$\max_{m,n} \sum_{t \in \textbf{Skip}} \min(|Q_t^{(m)}|, |D_t^{(n)}|) \leq \max_m \sum_{t \in \textbf{Skip}} w_{m,t} \leq \sum_{t \in \textbf{Skip}} \text{MaxRef}_t \tag{4.10}$$

which implies the selection of non-requirement set (named **Skip** set here for short) such that $\sum_{t \in \textbf{Skip}} \text{MaxRef}_t \leq \theta$ follows $w^*_{Q,D} \leq \theta$ for all non-requirement set posting lists.

4.1.4 GBP Pruning Strategies

Inequality 4.10 is relaxed twice, so it spurs the motivation to get an tighter upperbound by maximizing posting lists in the non-requirement set so that more posting lists are likely to be skipped. Define partial upperbound matrix $\mathbf{W} = \{w_{i,j}\}_{|T_q| \times |\mathfrak{T}|}$ where $\mathfrak{T} = \{\mathfrak{T}(m), m \in T_q\}$ are all the token paths from query OPT, and a binary variable $\boldsymbol{x}_{|\mathfrak{T}| \times 1}$ indicating which corresponding posting lists are assigned to the non-requirement set. One heuristic objective is to maximize the number of posting lists in the non-requirement set (we call this strategy GBP-NUM where GBP stands for Greedy Binary Programming):

$$\text{maximize} \qquad \mathbf{1}^T \cdot \boldsymbol{x} \tag{4.11}$$

$$s.t. \qquad \mathbf{W}\boldsymbol{x} \leq \theta \tag{4.12}$$

Maximizing the number of posting lists in the non-requirement set does not necessarily result in a significant increase in skipped items, as these posting lists can be very short. Thus, the potential for skipping items, represented by the number of posting list items in the non-requirement set, may not be sufficiently large. Instead, we can maximize the total length of the posting lists in the non-requirement set. In this case, the all-one vector in the objective function 4.11 is replaced with a posting list length vector $\mathbf{L} = \left[L_1, L_2, \ldots L_{|\mathfrak{T}|}\right]^T$ where L_i is the length of posting list i. We call this strategy GBP-LEN.

The two GBP strategies are rank-safe because $\forall\ m \in T_q, n \in T_d$,

$$\sum_t \min(|Q_t^{(m)}|, |D_t^{(n)}|) \leq \sum_t w_{m,t}\ , \tag{4.13}$$

and constraints in inequalities 4.12 demands $\sum_{t \in \textbf{Skip}} w_{m,t} \leq \theta$.

Both binary programming strategies require solving a binary programming problem which is known to be NP-complete thus too heavy to run for long queries. Therefore at implementation we just greedily follow one branch of the binary programming sub-problems to get a greedy solution running in $O(|T_q||\mathfrak{T}|^2)$. See detail steps in Algorithm 3. Note that greedy solutions also meet the constraints in 4.12, thus they are still rank-safe.

Algorithm 3 Greedy solver for GBP strategies

Input: $\mathbf{W}, \mathbf{L}, \theta$
Output: I, p indicating the assignment of posting lists
$I[1:p]$ are posting lists IDs in requirement set;
$I[(p+1):n]$ are posting list IDs in **Skip**.

function SolveRecur($\mathbf{W}, \mathbf{L}, \theta$, I, p)
    n := len($\mathbf{L}$)
    if p $\geq$ n **then**
        return I, p ▷ All in requirement set.
    $\boldsymbol{x}$:= $\{x_i\}_{n\times 1}$ where x[1 : p] are zeros, x[p + 1: n] are ones.
    r := row in $\mathbf{W}\boldsymbol{x}$ that violates the constraints 4.12 the most.
    if no row violates constraints **then**
        return I, p ▷ This is a feasible solution.
    c := a column such that $(\mathbf{L} \cdot \boldsymbol{x}^T)_c$ is minimal and $\mathbf{W}_{r,c} > 0$
    swap column c and p in $\mathbf{W}, \mathbf{L}$ and I ▷ Take out c heuristically.
    p := p + 1
    return SolveRecur($\mathbf{W}, \mathbf{L}, x, \theta$, p)
function Solve($\mathbf{W}, \mathbf{L}, \theta$)
    p := 0 ▷ An index pivot of requirement vs. non-requirement set.
    I := index vector [1 : len($\mathbf{L}$)] ▷ Length of $\mathbf{L}$ is essentially $|\mathfrak{T}|$.
    for each column j in $\mathbf{W}$ **do** ▷ Take out those obviously not in **Skip**.
        m := maximum element of column j in $\mathbf{W}$
        if m $> \theta$ **then**
            swap columns j and p in $\mathbf{W}, \mathbf{L}$ and I
            p := p + 1
    return SolveRecur($\mathbf{W}, \mathbf{L}, \theta$, I, p)

4.2 Implementation

The implementation of the models outlined in Section 4.1 is generally straightforward. However, to ensure efficiency, the index of a math-aware search engine requires special modifications for implementing dynamic pruning in structure search, it also requires a generalized inverted index to handle different modes of query keywords (e.g., text keywords or structured formula keywords). We will see, this setup also facilitates support for boolean specifiers (e.g., AND, OR, NOT) for both types of keywords.

The inverted index allows for the utilization of various optimization techniques such as dynamic pruning, proximity search [Tao and Zhai, 2007], and the straightforward sharding of index and parallelization of query processing using the document-at-a-time (DAAT) paradigm [Schütze et al., 2008]. In this following, I describe in detail how a modified and hierarchical inverted index is implemented to support fast query processing with dynamic pruning and boolean query operator support for querying structure math formulas and text keywords.

4.2.1 Structure Query Processing

Figure 4.2 illustrates structure query processing against the path inverted index for formula search. For each internal node m of the query OPT, I store the number of leaves of m as $w_m = |Q^{(m)}|$, the two numbers are denoted by m/w_m. Each query node refers to one or more tokenized path entries in a dictionary, where each reference is associated with $Q_t^{(m)}$ identified by query m and tokenized path t. For example, in Figure 4.2, node $q1$ from the query has 6 leaves, which is also the upperbound number of path matches for $q1$, i.e, $|Q^{(1)}|$. Since $q1$ consists of 2 tokenized leaf-root paths `VAR/MUL/ADD` and `VAR/ADD`, $q1$ is linked to two posting lists, each associated with a partial upperbound $w_{m,t} = |Q_t^{(m)}|$.

Each posting list corresponds to a token path $t \in \mathfrak{T}$ and has one dynamic reference counter associated, indicating the number of query nodes referring to it (initially $|Q_t|$). As a query node (representing a subexpression in the query math formula) can be pruned when its width is no longer greater than the current dynamic pruning threshold θ, the reference counter may decrease, and the posting list gets removed if its reference counter is less than one. The greatest $w_{m,t}$ value by which a posting list corresponding to path t gets referenced is the MaxRef value.

Each formula is identified by an expression ID, or *ExpID*. For ease of testing termination, I append a very large ExpID *MaxID* at the end, this special ID is greater than any ExpID of real document expression. Each posting list entry identified by an ExpID stores document

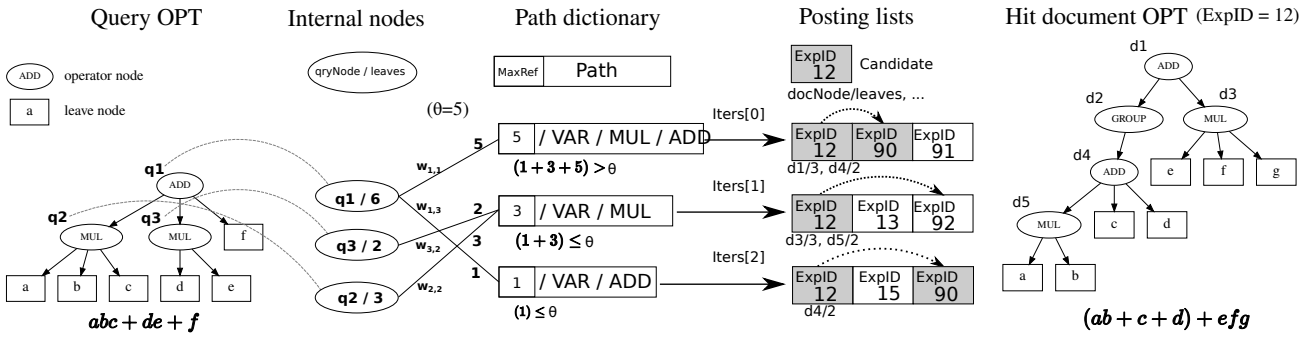


Figure 4.2: Index architecture for formula search with dynamic pruning. The top posting list is the only one in requirement set using strategy MaxRef, and the bottom two are advanced by skipping to candidate ExpID.

root-end node n, the number of paths under it (i.e., $|D_t^{(n)}|$), and other information for similarity scoring. As an example, in Figure 4.2, the hit OPT shown at right (of ExpID 12) has 5 paths tokenized as `VAR/MUL/ADD`, 2 on the left rooted by $d4$ and 3 on the right rooted by $d1$. And the information $(d1/3, d4/2)$ are stored with the corresponding posing list of `VAR/MUL/ADD`. In our implementation, a posting list t is traversed by its iterator $iters[t]$ to allow reentrant reading, and its entry information (denoted as n/w_n) are read by $iters[t].read()$ function from current position of the iterator.

The detailed query processing is described in Algorithm 4. The RequirementSet returns selected iterators of the requirement set (complement of the non-requirement set) depending on the specified pruning strategy. The assignment per strategy is described previously in Section 4.1. For MaxRef strategy in particular, posting lists are sorted in descending order of their MaxRef values, and posting lists are assigned into non-requirement set from lowest MaxRef value.

At merging, a document formula of the current minimal expID among those referenced by the iterators from the requirement set is selected as *candidate*. Iterators in the requirement set are only advanced one by one using the *next()* function, while iterators in the non-requirement set are advancing directly to the ID equal to or greater than the current candidate using the *skipTo()* function. Given Figure 4.2 for example, the posting list corresponding to `VAR/MUL/ADD` is the only posting list in the requirement set (assume it is using the MaxRef strategy). As a result, document formula 13 and 15 will be skipped if the next candidate is 90.

At each iteration, a set of *hitNodes* is inferred from the hit path set, containing query nodes associated with iterators having their current ExpIDs equal to the candidate ID.

Algorithm 4 Formula search using MaxScore pruning.

```
1: function QryNodeMatch(iters, m, candidate, widest, θ)
2:     nodeMatch[ ] := 0⃗; ℓ := |leaves(m)|                ▷ ℓ is the leftover estimated upperbound.
3:     for each m/w_m of tokenized path t rooted at m do
4:         Let i be the iterator index associated with t
5:         if iters[i].docID < candidate then
6:             iters[i].skipTo(candidate)
7:         if iters[i].docID = candidate then
8:             for each n/w_n of t from iters[i].read() do
9:                 nodeMatch[n] := nodeMatch[n] + min(w_m, w_n)
10:        ℓ := ℓ − w_m; estimate := max(nodeMatch) + ℓ                ▷ Update estimation.
11:        if estimate ≤ widest or u(estimate) ≤ θ then
12:            return 0
13:    return max(nodeMatch)
14:
15: function FormulaSearch(iters, strategy, θ_0 = 0)
16:    θ := θ_0; reqs := RequirementSet(θ, strategy)
17:    heap := data structure to hold top-k results
18:    while true do
19:        candidate := minimal ID in current expIDs of reqs
20:        if candidate equals MaxID then                ▷ Search terminated, return results.
21:            return top-k results
22:        Let G(Q, D, E) be the path set bipartite graph.
23:        widest := 0; hitNodes := {root_end(q) : (q, d) ∈ E}
24:        for m in hitNodes do          ▷ Calculate maximum match for each hit query node.
25:            if |leaves(m)| ≤ widest then
26:                continue
27:            maxMatch := QryNodeMatch(iters, m, candidate, widest, θ)
28:            if maxMatch > widest then widest := maxMatch          ▷ Find the widest width.
29:        if widest > 0 then
30:            score := calculate final score (including symbol similarity).      ▷ See Section 3.2.3.
31:            if heap is not full or score > θ then
32:                Push candidate or replace the lowest scored hit in heap.
33:                if heap is full then                ▷ Update current threshold.
34:                    θ := minimal score in current top-k results
35:                    Drop small query nodes and unreferenced iterators.
36:                    reqs := RequirementSet(θ, strategy)          ▷ Update requirement set.
37:        for iters[i] in reqs do                ▷ Advance posting list iterators.
38:            if iters[i].docID = candidate then iters[i].next()
```

QRYNODEMATCH calculates the match of each hit node according to Eq. 3.4 (or Eq. 3.5 if we consider path weights),[2] pruning those nodes whose maximum matching size will be smaller than the matched size of any already calculated node. For example, given query hit node $q1$ in Figure 4.2, QRYNODEMATCH returns

$$\max_{n \in T_d} \nu(G^{(1,n)}) = \max(\min(5,2) + \min(1,2),\ \min(5,3)) = 3. \tag{4.14}$$

Then the algorithm selects the best matched query node, and its matched width w^* (*widest*) is our interested structure similarity as defined in Eq. 4.7.

After obtaining w^*, the overall similarity score (see Section 3.2.3) can be computed. However, due to the additional layer involved in calculating the overall similarity, it becomes necessary to further relax the upperbound. For example, given the overall scoring function specified by Eq. 3.16, the relaxed upperbound function u is

$$u(w^*) = w^* \cdot \max_D\ S_{sym}(Q, D) \cdot P(D; \eta) = w^* \cdot \left[(1-\eta) + \eta \frac{1}{\log(1+w^*)}\right]. \tag{4.15}$$

Whenever threshold θ is updated, the algorithm will examine all the query nodes. If a query node m has an upperbound less or equal to the threshold, then the corresponding subtree of this node is too "small" to be considered as a candidate subexpression, and these nodes are going to be dropped for the remaining query process.[3] As a result, some of the posting lists iterators may also be dropped due to zero reference.

4.2.2 Heterogeneous Query Processing

To handle text keywords simultaneously and leverage the well-established techniques of inverted indexes, I propose a two-level inverted index architecture. The architecture of this hierarchical inverted indexes is illustrated in Figure 4.3.

Keywords are mapped directly to their respective inverted lists, where each item in the list may include the document ID, term frequency, and positional information for each occurrence in the document.

In contrast, when a math query is present in the query, it is mapped to multiple secondary inverted lists. These lists employ specialized structure to support dynamic

[2] Only minor modifications are required to adapt Algorithm 4 to account for path weights.

[3] Note that this pruning method enhancement is another novel addition, providing further improvement to our existing structure dynamic pruning approach.

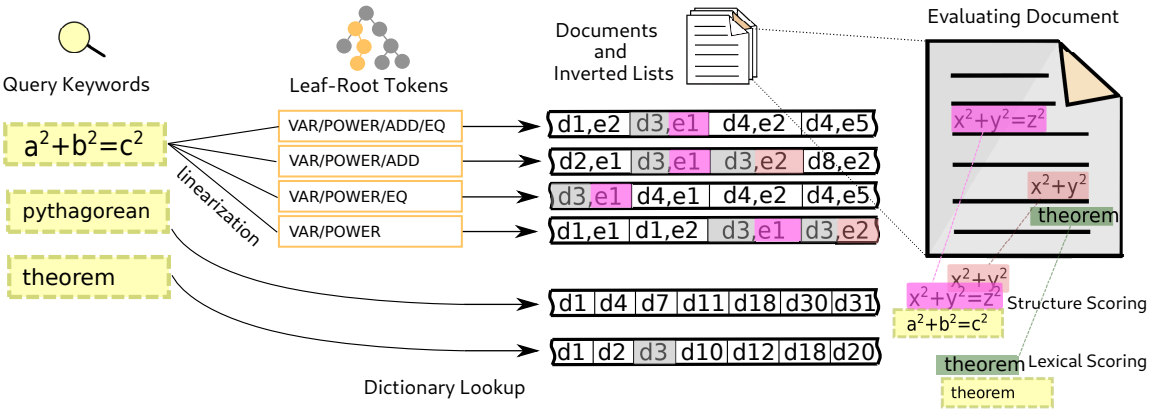


Figure 4.3: The high-level architecture of our hierarchical and heterogeneous indices taking in both math and text keywords. For math query keywords, we employ a linearization process to convert structured formulas into leaf-root paths, and subsequently to second-level inverted lists for formula search. Unlike top-level text inverted lists, the second-level math inverted lists have both document ID (`dX`) and expression ID (`eY`) indexed.

pruning techniques elaborated in Section 4.2.1. For each formula, the leaf-root paths and their prefixes are extracted from the query formula's OPT representation. These paths and prefixes serve as the vocabulary keys and are mapped to the corresponding secondary posting lists, similar to the structure depicted in Figure 4.2. Unlike the text inverted lists, the path inverted list stores additional index information specifically for scoring math formulas. This includes the document ID, formula ID, root-end node IDs of the corresponding path, formula length, frequencies of paths under each subtree, leaf-end symbols (operand symbols), and path *fingerprints* (refer to Section 3.2.1 for details).

When merging the text inverted lists and path inverted lists within the hierarchical indices, the math query processing can be conducted as a subroutine. This abstraction proves advantageous for integrating heterogeneous retrieval models for both math and text. It is worth noting that a document may contain multiple formula matches. Therefore, during query processing, only the maximum score returned by the best-matched formula in a document is returned to the upper-level query processing to facilitate document-level retrieval for formulas. Accordingly, the path indices at the secondary level are abstracted to present themselves solely through an identical text search inverted list interface to the upper-level. In other words, the secondary posting lists for a math keyword behave as if the keyword were a regular text keyword during the upper-level query processing. Specifically, only document-level IDs are returned and merged with the text inverted lists. The formula

IDs of a document are exclusively utilized internally within the sub-level query processing, as detailed in Section 4.2.1.

Algorithm 5 provides a in-detail description of heterogeneous retrieval using MaxScore dynamic pruning as an example. The hierarchical inverted index offers the advantage of being able to apply traditional inverted list optimization techniques with minimal modifications. In comparison to a regular MaxScore algorithm, only Lines 26–30 (highlighted in red) have been introduced to handle math queries. The use of a heterogeneous inverted list allows the formula search algorithm (Algorithm 4) to be executed as a subroutine (Line 28), ensuring consistency in the upper-level implementation with the text retrieval model.

The hierarchical inverted list also enables boolean filtering of queries at the keyword level. A boolean clause can be associated with a keyword, regardless of its type. However, there is a distinction for math queries in terms of boolean filtering: while text keyword boolean specifier as a binary filter, the candidates of a query formula keyword Q are filtered based on a formula partial score threshold. Line 29 illustrates this process. If a formula keyword is specified with a NOT clause, a document formula can be excluded from consideration if its score exceeds a certain threshold $u(Q)$, and

$$u(Q) = \lambda_m \cdot \begin{cases} \lambda_n \cdot l(Q) & \text{for NOT clause} \\ \max_D \ S(Q, D) & \text{otherwise} \end{cases} \tag{4.16}$$

where $l(Q) = \gamma(T_Q) \cdot \min_D \ S_{sym}(Q, D \mid T_Q) \cdot P(D; \eta)$ is the similarity lowerbound – any formula scored lower than this is going to have a partial structure match. Hyperparameter $\lambda_n \in [0, 1]$ controls the threshold to ignore any formula scored above it if the formula is specified with a NOT clause, and λ_m is the math path weight (see Section 3.2.3). Lastly, $S(Q, D)$ is the similarity score for query and document formula Q and D (see Eq. 3.16).

4.3 More Efficient Dynamic Pruning

4.3.1 Initial Threshold

In Algorithm 4, an *initial threshold* θ_0 is defined as a parameter. The initial threshold provides an effective method to trade off ranking safety for efficiency. By setting a non-zero value for θ_0, a minimum score is defined for any hit to qualify as a candidate, irrespective of whether it exceeds the eventual minimum top-k score. Consequently, the guarantee of ranking safeness cannot be ensured in this scenario. However, this small modification

Algorithm 5 Heterogeneous search using MaxScore pruning.

Input iters: inverted list iterators; u: corresponding upperbounds.

```
1: function HETEROGENEOUSMAXSCORE(iters, u)
2:     heap := data structure to hold top-k results
3:     iters := iters sorted by u in descending order
4:     θ, N := 0, number of iterators in iters
5:     u[i], op[i] := the associated upperbound and boolean clause for iters[i], i = 1, 2, ..., N
6:     a[i] := Σ_{j=i}^{N} u[i] for i = 1, 2, ..., N or ∞ for i > N
7:     pivot := argmax_i a[i] > θ
8:     while true do
9:         candidate := minimal docID in current reqs
10:        if candidate equals MaxID then                    ▷ Search terminated, return results.
11:            return top-k results
12:        score := 0                                        ▷ Candidate document score.
13:        for i := 1, 2, ..., N do
14:            if score + a[i] < θ then      ▷ Updated score estimation cannot make into top-k.
15:                break
16:            else if i > pivot then                        ▷ Skip items in non-requirement set.
17:                iters[i].skipTo(candidate)
18:            if iters[i].docID ≠ candidate then
19:                if op[i] is AND then
20:                    break                                 ▷ Handle AND clause.
21:                else
22:                    continue
23:            if iters[i] is associated to a text keyword then
24:                if op[i] is NOT then break                ▷ Handle text NOT clause.
25:                score := score + tf–idf_i
26:            else if iters[i] is associated to a math query keyword then
27:                θ_i := (θ− score −(a[i] − u[i]))/λ_m
28:                m := highest score in formula(s) of candidate   ▷ Applying Algorithm 4 w/ θ_i.
29:                if op[i] is NOT and λ_m · m > u[i] then break   ▷ Handle math NOT clause.
30:                score := score + λ_m · m
31:        if i = N and (heap is not full or score > θ) then
32:            Push candidate or replace the lowest scored hit in heap.
33:            if heap is full then
34:                θ := minimal score in current top-k results     ▷ Update current threshold.
35:                pivot := argmax_i a[i] > θ                      ▷ Update the pivot.
36:        for i ∈ 1, 2, ..., pivot do                   ▷ Advance iterators in requirement set.
37:            if iters[i].docID = candidate then iters[i].next()
```

has significant practical implications: In real-world scenarios, query formulas often contain tens if not hundreds of leaf-root paths. Even with a small non-zero initial threshold, a large number of document formulas can be pruned effectively.

While the inclusion of a non-zero θ_0 may compromise the ranking safeness of the algorithm, it is important to note that math formula search inherently looks for candidates with substantial overlapping common subexpression(s). Otherwise, if a candidate formula exhibits a substantial difference in OPT size compared to the query formula, it indicates a low likelihood of preference for that particular candidate. Therefore, this extra pruning does not affect candidates that exhibit closer structure similarity to the query, while boosting the efficiency further.

As a result, employing a small θ_0 proportionally to the size of the query formula is generally safe. To achieve this, we introduce a hyperparameter known as the *initial threshold factor* $\lambda_0 \geq 0$. This factor allows us to automatically adjust θ_0 to be equal to the lowerbound score of small subset of query operands:

$$\theta_0 = l(\lambda_0 \cdot \text{leaves}(Q)) \tag{4.17}$$

where the lowerbound function l is also used in Eq. 4.16.

4.3.2 Combining with Dynamic Thresholds

Each time the threshold is updated in the heterogeneous search, the threshold for each math formula used in the formula search subroutine needs to be re-calculated. This is done in two cases during query processing: (1) If precise scores for some query keywords have already been obtained, the dynamic threshold is computed based on those scores, as described in Line 27 of Algorithm 5. (2) Otherwise, independent of the document, the dynamic threshold is estimated by assuming that other query keywords q_j contribute their upper-bound scores:

$$\theta_i = \frac{1}{\lambda_m}\left[\theta - \sum_{j \neq i} u(q_j)\right] \tag{4.18}$$

where $u(q_j)$ is the upperbound score for a query keyword q_j.

In both scenarios, the final effective threshold can be determined by combining the dynamic threshold with the initial threshold. This is achieved by taking the maximum value between θ_0 (the static initial threshold) and θ_i (the dynamic threshold computed for each math formula keyword). In other words, the final effective threshold takes into

account both the initial threshold and the dynamically estimated threshold based on the specific query and available information. The threshold in case (1) provides a more accurate estimate and a higher value for pruning more candidates, making it consistently utilized. The threshold in case (2) remains relatively static and serves as the basis for determining the pruning strategies described in Section 4.1.

4.4 Evaluation

4.4.1 Experimental Setup

Two formula-only datasets are considered: the NTCIR-12 WMB benchmark and our self-hosted MSE corpus. The NTCIR-12 WMB benchmark is chosen for evaluating efficiency due to its diversified queries, which cover math formulas with varying levels of complexity. On average, these math queries are more complex compared to ARQMath topics [Mansouri et al., 2021a]. There are 20 queries containing wildcards in this task, we add support for the wildcards by simply treating internal nodes (representing a rooted subexpression) of formulas as additional "leaves" (by ignoring their descendants), and the wildcard specifiers in a query are treated as normal leaves to match those indexed wildcard paths. The MSE corpus is created with a sufficient number of document formulas to ensure noticeable efficiency differences when compared to the baselines. The evaluation of query efficiency against this dataset is conducted using exactly the same NTCIR-12 WMB topics.

In the implementation, all relevant inverted lists are compressed if not specified otherwise. The inverted list is implemented by a two-level skip list data structure running on disk, and list items are integer values compressed using Patched Frame-of-reference encoding, or PFOR [Zukowski et al., 2006] (the document IDs are further delta compressed, i.e., using delta encoding [Schütze et al., 2008]).

In the evaluation, queries are executed on the same hardware environment. Depending on the experiment, two workstations are used: one with Intel Core at 3.6 GHz per core and SSD drive, another is a personal workstation with Intel Core i5-8600K CPU and Toshiba HDWD110 hard drive.

Queries for the same experiment are executed in the same environment using single threads without index sharding. During evaluation, the run times solely focus on evaluating the merging of inverted lists. This means that the search server is initialized only once for the entire evaluation process. In addition, the first run is excluded because we find that

the first "warm-up" run has a high variance – after the first run, the index is implicitly cached into memory (by the OS or filesystem).[4]

4.4.2 Main Results

Table 4.1: Query run times (in milliseconds) for different pruning strategies compared to exhaustive search baseline. k is the number of search results we keep dynamically.

	Runs		Non-wildcards					Wildcards				
	k	**Strategy**	μ	σ	**median**	**min**	**max**	μ	σ	**median**	**min**	**max**
NTCIR-12 WMB	100	Baseline	540.12	569.44	360.50	7.00	2238.00	426.73	383.47	225.50	8.00	1338.00
	100	MaxRef	90.29	74.14	79.00	3.00	312.00	145.50	121.19	136.00	**7.00**	573.00
		GBP-NUM	84.90	80.44	52.50	3.00	321.00	138.82	102.55	135.00	9.00	428.00
		GBP-LEN	**67.49**	**61.40**	**45.00**	**2.00**	**218.00**	**125.27**	**97.28**	**103.50**	9.00	**404.00**
	200	MaxRef	107.71	**82.64**	102.00	**5.00**	**322.00**	160.10	121.40	149.00	9.00	583.00
		GBP-NUM	105.34	99.51	71.50	**5.00**	357.00	155.52	110.61	153.00	**8.00**	479.00
		GBP-LEN	**89.63**	83.20	**62.00**	**5.00**	330.00	**142.78**	**103.11**	**143.50**	9.00	**446.00**
	1000	MaxRef	154.51	**93.75**	157.50	**6.00**	**361.00**	211.86	140.01	186.00	10.00	662.00
		GBP-NUM	159.80	143.70	120.50	**6.00**	626.00	208.91	136.42	178.50	10.00	591.00
		GBP-LEN	**144.25**	126.95	**105.00**	**6.00**	622.00	**195.70**	**122.25**	**176.00**	**9.00**	**536.00**
MSE Corpus	100	Baseline	15134.10	15186.78	11161.00	157.00	55499.00	13450.57	12554.19	7075.50	304.00	47513.00
	100	MaxRef	1083.23	1274.23	745.50	28.00	5922.00	3188.66	2458.91	2925.00	85.00	10412.00
		GBP-NUM	1202.24	1240.21	815.00	37.00	4987.00	2943.79	2025.96	2987.00	**84.00**	8775.00
		GBP-LEN	**562.83**	**635.26**	**382.50**	**24.00**	**2313.00**	**2257.95**	**1491.59**	**2346.50**	86.00	**4494.00**
	200	MaxRef	1261.21	1368.93	1012.50	30.00	6439.00	3416.77	2753.09	3032.50	160.00	12412.00
		GBP-NUM	1378.19	1398.08	998.50	39.00	5863.00	3174.93	2283.05	3125.00	**159.00**	10099.00
		GBP-LEN	**697.32**	**739.11**	**478.00**	**27.00**	**2925.00**	**2504.90**	**1683.16**	**2382.50**	**159.00**	**6049.00**
	1000	MaxRef	2030.05	1746.17	1796.50	53.00	7816.00	4123.26	3510.01	3473.00	287.00	16981.00
		GBP-NUM	1952.52	1746.05	1530.50	60.00	7197.00	3786.89	2744.99	3493.50	**281.00**	11323.00
		GBP-LEN	**1217.16**	**1083.53**	**764.50**	**47.00**	**3756.00**	**3304.69**	**2403.09**	**2812.00**	285.00	**9895.00**

Considering different dynamic pruning strategies (Section 4.1), Table 4.1 reports the run time statistics. Non-pruning (exhaustive search) baselines with k = 100 are also compared.

Almost consistently, GBP-LEN achieves the best efficiency with smaller variance. This is expected since GBP-LEN models the skipping possibility better than GBP-NUM. Although GBP-NUM yields a tighter upperbound than MaxRef, it is optimized using a greedy method and only maximizes the number of posting lists in the non-requirement set; the latter may lead to bad performance when these posting lists are short. There are a few times the best minimal run times are from other strategies, for those with meaningful

[4]To justify this, we use the Linux strace -cw command to track the wall clock times for the query processing, first-time run spends 62.10% of the time on disk IOs, and after the first run, only 16.46% of the time is spent on the read system calls.

Table 4.2: Post-hoc bpref scores for the dynamic pruning strategy GBP-LEN compared against the baselines of other most-effective systems and our own non-pruning methods. All scores are evaluated on the NTCIR-12 dataset.

System	Non-Wildcard		Wildcard		All queries	
	Full	Partial	Full	Partial	Full	Partial
MCAT	.5678	.5698	**.4725**	.5015	.5202	.5356
Tangent-S	.6361	.5872	.4699	**.5368**	**.5530**	**.5620**
base-best	**.6726**	**.5950**	-	-	-	-
base-opd-only	.6586	.5153	-	-	-	-
Ours (pruning)	.6586	.5173	.3678	.3973	.5132	.4573
Ours (exhaustive)	.6586	.5173	.3678	.3973	.5132	.4573

gaps, i.e., in NTCIR-12 WMB non-wildcard queries when $k = 1000$, MaxRef outperforms in standard deviation and maximum run time by a notable margin; however, it is likely due to a small threshold and a large k, so that the efficiency on this smaller dataset is less affected by pruning (small θ means less pruning potential) compared to the time complexity added from assigning to the requirement set. The latter is more dominant in GBP runs. In wildcard queries, however, many expressions can match the query thus the threshold value is expected to be larger than that in the non-wildcard case.

In additional to theoretical justifications in Section 4.1, I empirically show the rank-safeness of the proposed dynamic pruning strategies by comparing the effectiveness of pruning and exhaustive versions for top-1000 results shown in Table 4.2. The post-hoc bpref scores for the dynamic pruning methods are compared against the baselines of other effective systems, including our own non-pruning versions. Among the systems evaluated on this dataset, our previous version, specifically the `base-best` run, along with MCAT and Tangent-S, are selected because they are most effective non-supervised retrievers.

Our pruning strategies are rank-safe (pruning and exhaustive version shows exactly the same bpref scores) but there is a minor bpref difference between the newer versions and our previous baseline (`base-opd-only`, evaluating only single-tree matched operands) due to parser changes we have applied to support wildcards (e.g., to handle single left brace array as seen in a wildcard query) and they happen to slightly improve accuracy in partially relevant cases.

Compared to other systems, our pruning methods achieve the second best fully-relevant results while being able to execute queries with a maximum of half-second run time on

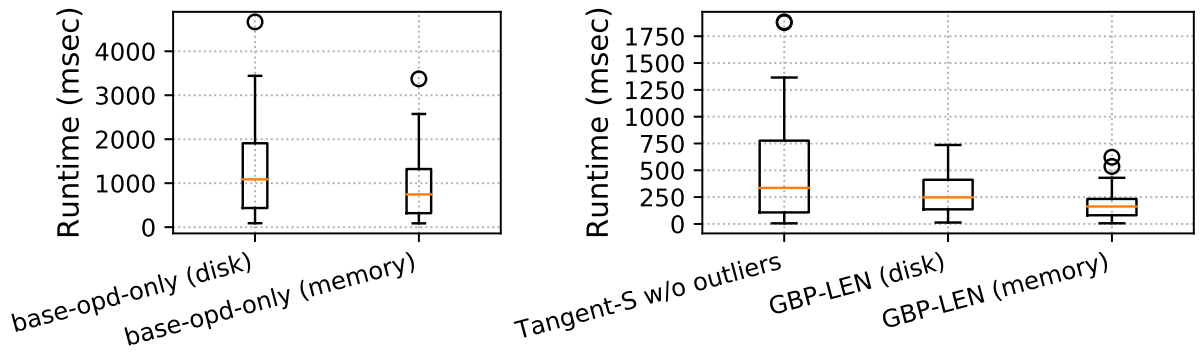


Figure 4.4: Our efficiency results evaluated on the NTCIR-12 dataset. Our non-pruning versions and the Tangent-S system are compared, with Tangent-S outlier queries excluded. At the time when the experiment is conducted, the Tangent-S system stands as the most efficient structure search approach evaluated on this dataset.

NTCIR 12 (see Table 4.1). It is worth noting that our efficiency surpasses that of the `base-opd-only` system, even though the latter only considers less complex non-wildcard queries. In contrast, MCAT reportedly has a median query execution time around 25 seconds, using a server machine and multi-threading [Kristianto et al., 2016]. On the other hand, Tangent-S has a few significant outliers (with run times > 50 seconds) as a result of its costly alignment algorithm to rerank structure and find the Maximum Subtree Similarity [Davila and Zanibbi, 2017]; its non-linear complexity makes it expensive for some long queries (especially in wildcard case). GBP-LEN outperforms Tangent-S in efficiency even if we exclude their outlier queries (see Figure 4.4), with a higher bpref in non-wildcard fully relevant results. However, our overall effectiveness is skewed by bad performance of wildcard queries because a much more expensive phase is introduced to boost accuracy by other systems to handle inherently difficult structrual wildcards.

4.4.3 Analysis

To comprehensively assess the effects of different parameters in the proposed structure dynamic pruning, I employ the optimal GBP-LEN strategy using single-tree matching.

Impacts of k As dynamic pruning efficiency is largely affected by the threshold value, we vary parameter k to get query execution performance under different threshold settings. As depicted in Figure 4.5, the dynamic-pruning structure search method demonstrates

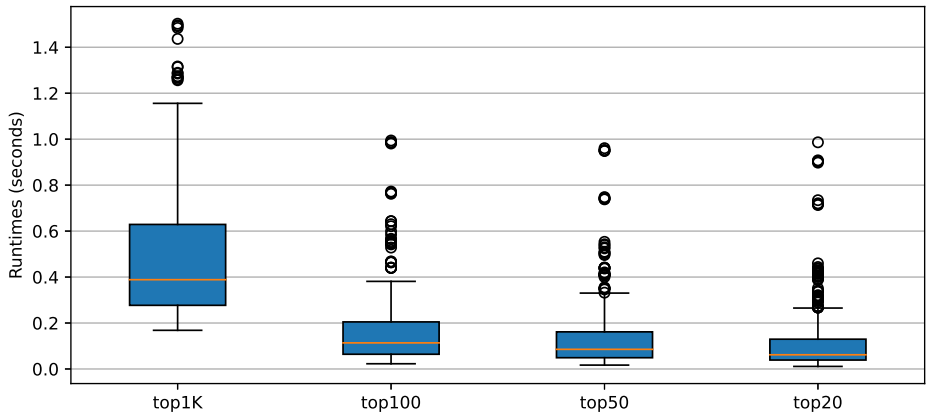


Figure 4.5: Query run times with dynamic pruning for different k values, evaluated on the ARQMath-3 dataset using Task 1 topics.

practical efficiency by completing a query in under a second on average, even when configured to retrieve 1000 results from the million-scale ARQMath corpus. Furthermore, the method's performance can be further enhanced, achieving an median query time of under 200 milliseconds for smaller values of k.

To the best of our knowledge, this is the first instance where a structure search retrieval model, utilizing a traditional inverted index, has accomplished sub-second query run times on a real-world scale of data. In typical online scenarios, it is common to have multiple search nodes operating in parallel, each node typically returns only 100 results with 10 results per page. Consequently, the reported run times for $k = 100$ can be considered as practical run times for real-world scenarios.

Impacts of threshold (θ) to pruning Compared to traditional dynamic pruning, an additional number of factors are involved to cause pruning for structure search. This includes deleted query nodes and the amount of removed (zero-referenced) inverted list iterators as a result of deleted query nodes. Importantly, the reduction of the number of inverted lists and of the size of the requirement set are directly responsible for query speedup. We plot these important pruning factors at different time steps in Figure 4.6.

As illustrated in Figure 4.6, the thresholds exhibit a consistent monotonic increase across various topics. Consequently, the number of query OPT nodes typically experiences a significant decrease, particularly because multiple nodes within a single tree possess an equal number of children. As a result, it leads to a notable reduction in the number of query

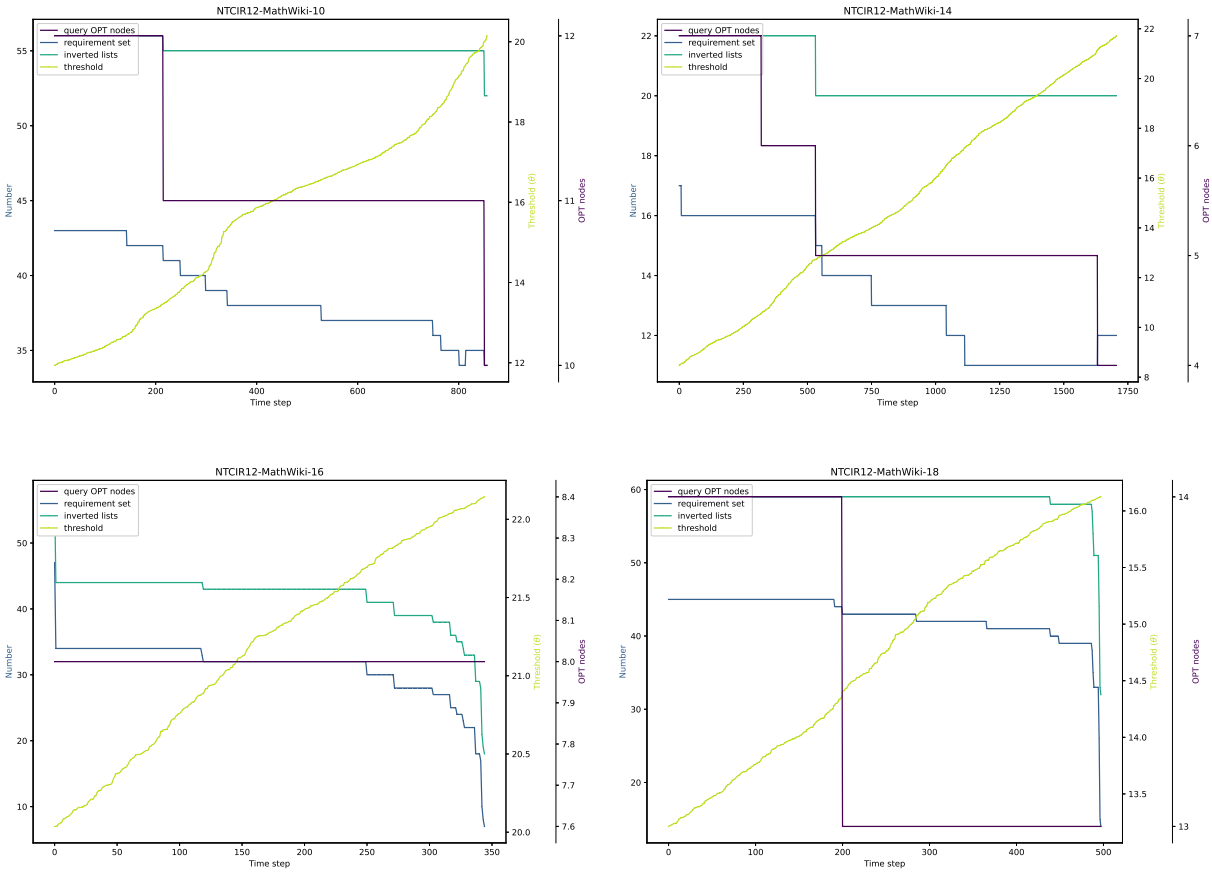


Figure 4.6: Single formula query pruning logs for different topics sampled from and evaluated on the NTCIR-12 WFB dataset. Threshold value (θ), the number of query OPT nodes, the size of requirement set, and the number of inverted list iterators are plotted at different time steps, showcasing the evolving nature of the thresholds and their corresponding changes across various topics.

OPT nodes. In contrast, the size of the requirement set and the total number of inverted lists decrease gradually until a sudden drop at the end of the query processing. This phenomenon is most likely caused by the iterators reaching the end of the inverted lists. Note that the size of the requirement set is always less than the total number of inverted lists, but different queries demonstrate different ratios during the retrieval process.

In summary, the pruning characteristics are influenced by the specific query being executed, and employing various pruning factors collectively leads to improved efficiency.

Initial thresholds As shown in Figure 4.7, additional acceleration can be obtained by increasing the initial threshold. Across the two formula search datasets and different evaluation metrics, we observe consistent patterns regarding the trade-off between effectiveness and efficiency with respect to this parameter. Specifically, an initial threshold of $\theta = 1.0$ has minimal impact on effectiveness, followed by a rapid decline in effectiveness after $\theta = 1.5$. This confirms our hypothesis in Section 4.3 that employing a small threshold is relatively safe. On the other hand, there is a slight recovery around $\theta = 2.5$ when the remaining candidates exhibit high similarity to the query structures. In this case, a slight increase in the threshold may help maintain a high precision. However, as the threshold further increases and begins to submerge the entire query structure, the effectiveness deteriorates.

These patterns underscore the delicate balance between effectiveness and efficiency in retrieval performance. To achieve a safer trade-off with minimal efficiency drops while still maintaining a reasonable level of effectiveness, it is suggested to select an initial threshold lower than 1.0.

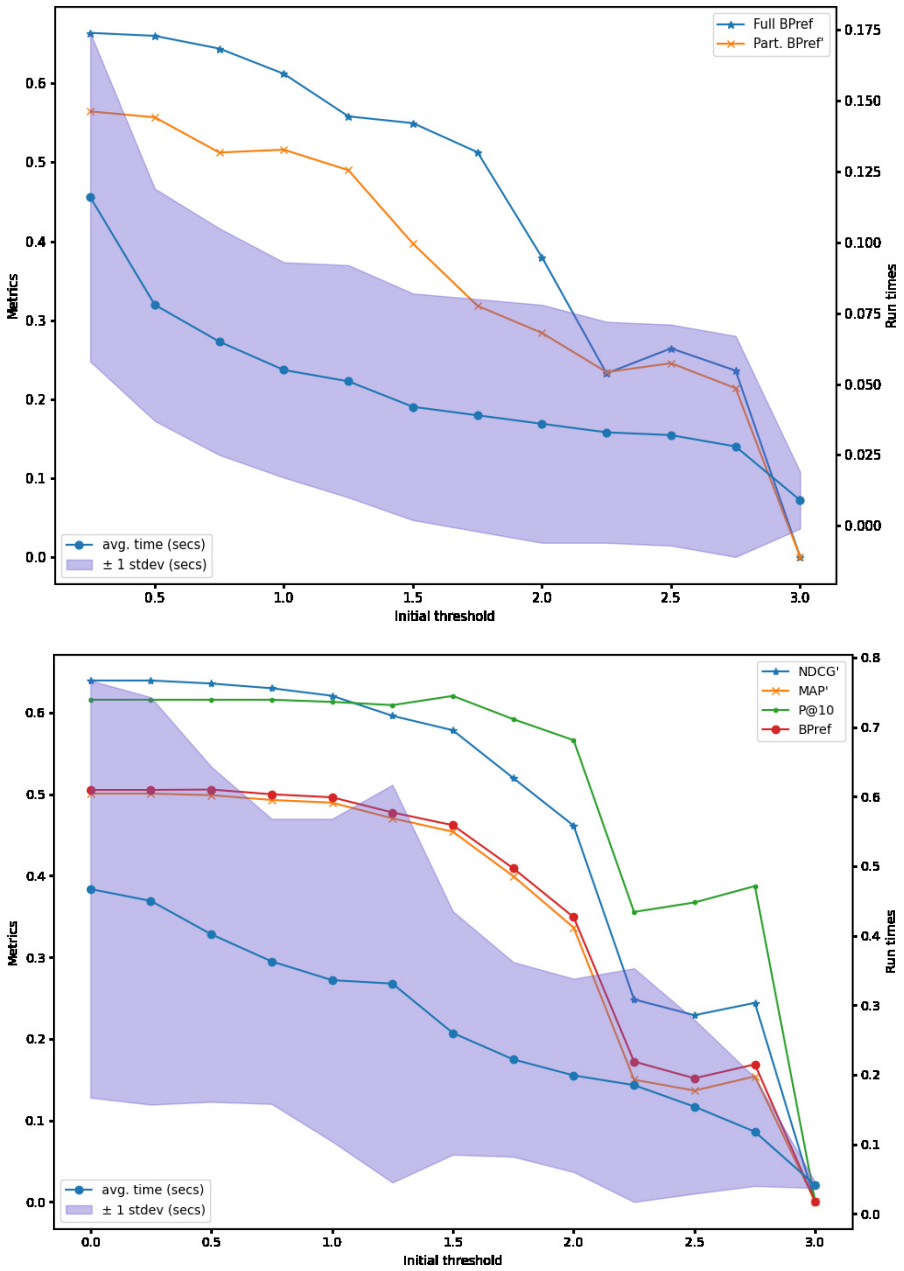


Figure 4.7: The query run times within one standard deviation using dynamic pruning for different k values, evaluated on the NTCIR-12 WFB (top) and the ARQMath-3 Task 1 (bottom). The corresponding effectiveness metrics at each initial threshold sample point are also depicted.

Chapter 5

Supervised and Hybrid Search

Supervised retrieval aims to capture nuanced and diverse aspects of semantic similarities. In recent developments [Izacard et al., 2021, Ram et al., 2021], leveraging powerful architectures like Transformers and abundant large-scale data containing relevance signals, supervised retrieval has surpassed highly effective unsupervised methods such as BM25 in zero-shot settings.

Supervised methods are particularly valuable in conjunction with structure search when scoring math documents. This is because math documents, having formulas surrounded by highly contextualized natural language, require learning from data to capture similarity nuances and contextual connections between a formula and surrounding texts. Additionally, unlike structure search, manually defining context similarity in an unsupervised manner is challenging. Lastly, recent work has successfully achieved good effectiveness using *hybrid search*, i.e., combining supervised retrieval with unsupervised retrieval in both general information retrieval [Lin, 2022, Ram et al., 2021, Shen et al., 2022] and math information retrieval [Zhong et al., 2022b,a, Kane et al., 2022].

In this Chapter, I will describe a novel training scheme for Transformer-based retrieval models that improves the supervised representation for highly contextualized math documents. Furthermore, I propose to advance math-aware retrieval using hybrid search with different effective learned representations tailored for complementary purposes. This method has achieved the state-of-the-art effectiveness on the most recent full-text MIR benchmarks, while employing efficient retrieval components that allow for sub-second query execution times [Zhong et al., 2023].

5.1 Contextualized Pretraining

Formulas in math documents are highly contextual, often their semantics are defined by surrounding words. A context-enhanced pretraining method is described to improve the downstream math-aware retrieval effectiveness.

5.1.1 Preliminaries

A Transformer encoder $\text{Enc}^{(l)}$ at layer $l = 1, 2, ...L$ encodes a sequence of hidden states produced from the lower layer $l-1$ and outputs hidden states $h_i^{(l)} \in \mathbb{R}^d$ for each token i of this layer l:

$$h_0^{(l)}, h_1^{(l)}, ..., h_n^{(l)} = \text{Enc}^{(l)} \left(h_0^{(l-1)}, h_1^{(l-1)}, ..., h_n^{(l-1)} \right) \tag{5.1}$$

$$h_i^{(0)} = \text{PosEmb} \left(t_i \cdot E^T,\ i \right) \ , \quad i = 0, 1, ..., n. \tag{5.2}$$

where each initial-layer hidden state $h_i^{(0)}$ at position i is the positional embedding (PosEmb) of an input token embedding and the corresponding position i. The input token embedding of $t_i \in \mathbb{R}^V$ is mapped by the learnable embedding matrix $E \in \mathbb{R}^{d \times V}$ where V is the vocabulary size and d is the dimension of hidden state. Each hidden state from the outputs of a Transformer encoder is *contextual* since it is dependent on all the hidden states from the previous layer.

BERT pretraining

The pretraining of the BERT model requires stacking two heads on top of a Transformer encoder, one for the MLM task and the other for the NSP task [Devlin et al., 2019]:

$$w_i = H_{\text{MLM}}(h_i^{(L)}) \cdot E + b \ , \quad i = 0, 1, ..., n; \tag{5.3}$$

$$s = H_{\text{NSP}}(h_0^{(L)}) \tag{5.4}$$

where H_{MLM} is a linear projection followed by a GELU activation layer [Hendrycks and Gimpel, 2016] and layer normalization. Whereas H_{NSP} is a projection to a binary-classification distribution $s \in \mathbb{R}^2$.

In BERT, sentence pairs with the delimiter token `[SEP]` are used as inputs for pretraining: For the MLM task, some of the original tokens $\bar{t}_i$ from random indices but excluding

indices of special tokens, i.e., $i \in I = I_{\text{random}} \setminus I_{\text{special}}$, are either replaced by a `[MASK]` token or altered to another random token in the vocabulary before feeding to the Transformer; for the NSP task, a one-hot label $\bar{s}$ indicating whether a pair of sentences are continuous is used for calculating the NSP loss:

$$\mathcal{L}_{\text{MLM}} = \frac{1}{|I|} \sum_{i \in I} \text{CrossEntropy}(w_i, \bar{t}_i) \tag{5.5}$$

$$\mathcal{L}_{\text{NSP}} = \text{CrossEntropy}(s, \bar{s}). \tag{5.6}$$

In actual inputs, the first token $\bar{t}_0$ is set to `[CLS]` so that its corresponding contextualized embedding $h_0^{(L)}$ is free to be used for representing passage-level embedding.

MAE pretraining

Compared to the BERT pretraining, the Masked Auto-Encoding (MAE) pretraining for retrievers [Gao and Callan, 2021b,a, Lu et al., 2021, Wu et al., 2022] further adds a decoder Dec on top of the Transformer encoder. It relies on the hidden state $h_0^{(L)}$ as the information bottleneck to *auto encode* relevant context signals from predicting an auxiliary MLM task in the decoder. The decoder usually uses the same Transformer architecture, where the number of layers M is generally smaller than L. Additionally, the decoder uses hidden states from a lower layer k of the encoder as its inputs:

$$\widetilde{h}_0^{(l)}, \widetilde{h}_1^{(l)}, ..., \widetilde{h}_n^{(l)} = \text{Dec}^{(l)} \left(\widetilde{h}_0^{(l-1)}, \widetilde{h}_1^{(l-1)}, ..., \widetilde{h}_n^{(l-1)} \right) \tag{5.7}$$

$$\widetilde{h}^{(0)} = \left[h_0^{(L)}, h_1^{(k)}, h_2^{(k)}, ..., h_n^{(k)} \right]^T \tag{5.8}$$

where $\widetilde{h}_i^{(l)}$ is the decoder hidden state at position i in decoder layer $l = 1, 2, ..., M$.

The aforementioned setting of using a smaller decoder and skipping from a lower layer is to limit the power of the decoder so that the learned code at $h_0^{(L)}$ is encouraged to be more informative to assist the decoding. Similarly, the decoder MLM loss is defined as

$$\widetilde{w}_i = \widetilde{H}_{\text{MLM}}(\widetilde{h}_i^{(M)}) \cdot E + \widetilde{b}\,, \quad i = 0, 1, ..., n. \tag{5.9}$$

$$\mathcal{L}_{\text{MLM}}^{\text{dec}} = \frac{1}{|I|} \sum_{i \in I} \text{CrossEntropy}(\widetilde{w}_i, \bar{t}_i) \tag{5.10}$$

where $\widetilde{H}_{\text{MLM}}$ is the decoder MLM head, which can share the same parameters with the encoder, i.e., H_{MLM}.

Instead of using sentence pairs, MAE-based pretraining generally simulates the downstream retrieval tasks and encode a complete passage or span in a query-agnostic fashion. Specifically, the *Inverse Cloze Task* (ICT) contrastive loss [Lee et al., 2019] applied on in-context document span pairs s_i, s_i^* in a batch B is often used to replace NSP:

$$\mathcal{L}_{\text{ICT}}(s_1, s_1^*, ..., s_{|B|}, s_{|B|}^*) = -\frac{1}{|B|}\sum_{i \in B} \log \frac{\exp\left(S(s_i, s_i^*)\right)}{\sum_{j \in B} \exp\left(S(s_j, s_j^*)\right)} \tag{5.11}$$

where function S computes the dot-product similarity between the two spans using their passage-level embeddings. This *document-as-query* self-supervised training for retrieval purposes has different variations [Chang et al., 2020, Gao and Callan, 2021a, Ram et al., 2021, Ma et al., 2022a, Wu et al., 2022]. Although we specifically model formulas and their surrounding text by creating contrastive span pairs here, we still use the name ICT to refer to these pretraining tasks because they all share the same spirit.

5.1.2 Coco-MAE Pretraining

In our pretraining data, math documents are split into spans of random lengths consisting of heterogeneous tokens. Given a span s, we pair it with an in-document excerpt s^* that does not intersect s.

A more challenging task is proposed to inject more information to the contexualized representation of math documents. We use the MAE pretraining architecture, but the encoded $h_0^{(L)}$ codes are swapped before feeding them to the decoder so that the decoder has to rely on the in-context code for the MLM task. Presumably, this will encourage the model to better encode higher-level context. More specifically,

$$\widetilde{h}^{(0)} = h_0^{(L)*}, h_1^{(k)}, h_2^{(k)}, ..., h_n^{(k)} \tag{5.12}$$

$$\widetilde{h}^{(0)*} = h_0^{(L)}, h_1^{(k)*}, h_2^{(k)*}, ..., h_n^{(k)*} \tag{5.13}$$

where the star notation on the superscripts is used to denote which spans the hidden states are taking input from, given a span s and its in-context pair s^*. Note that this setting is similar to the Cot-MAE pretraining in general text retrieval [Wu et al., 2022]. However, we use the encoder layer at $k > 0$ to feed the decoder while Cot-MAE starts the skip links from $k = 0$, where embeddings are non-contextual. Skipping from a higher layer prevents the learned code in MAE from encoding too much lower-level syntactic information from math tokens. Also different from Cot-MAE, we further include the ICT loss for the learned code to better capture highly contextual math semantics during the pretraining stage.

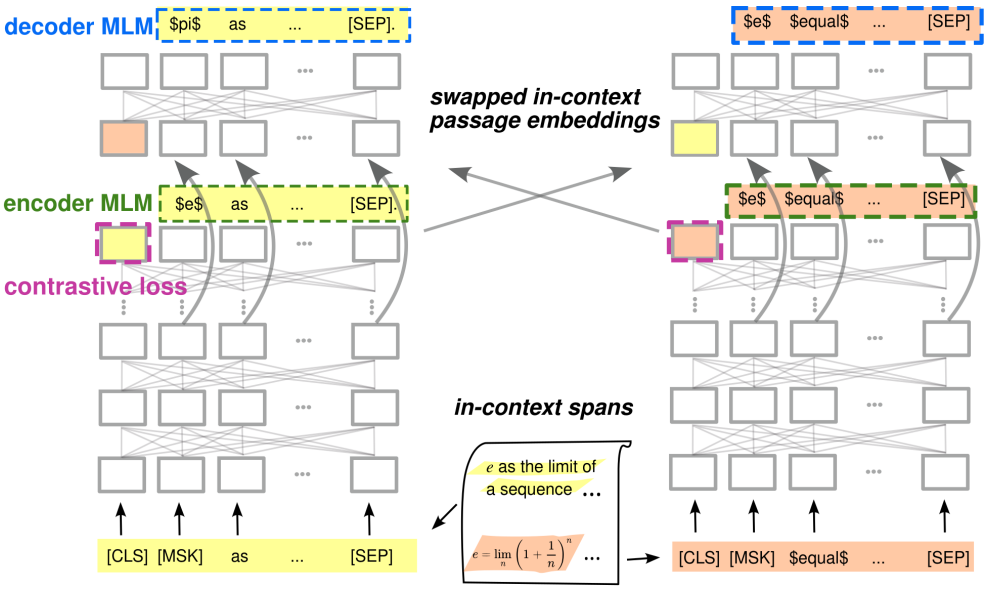


Figure 5.1: Illustration of our contrastive pretraining. A decoder is placed on top of a retriever backbone to train an auxiliary MLM objective at the same time including the contrastive loss to create a good balance of passage- and token-level information in the learned retrieval representation.

Our pretraining loss is a summation of encoder MLM, decoder MLM, and contrastive ICT in a batch $\{s_i, s_i^*\}_i^B$:

$$\mathcal{L} = \mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{MLM}}^{\text{dec}} + \mathcal{L}_{\text{ICT}}(s_1, s_1^*, ..., s_{|B|}, s_{|B|}^*) \tag{5.14}$$

We name this pretraining method Coco-MAE due to its similarity to both CoCondenser [Gao and Callan, 2021a] and Cot-MAE [Wu et al., 2022]. Coco-MAE differs from CoCondenser in that the in-context MAE code is swapped before feeding to the MLM decoder, leading to a more challenging decoder task. For heterogeneous math topics, we will show in Section 5.3.3 that this change is important for learning efficiency. Our pretraining scheme is illustrated in Figure 5.1.

5.2 Domain-optimized Hybrid Search

In Zhong et al. [2023], we also complement math structure search with efficient components and hybrid representations of different strengths (see Section 5.2.3). We suggest a set

of complementary search components (under the name MABOWDOR) for effective and efficient MIR.

5.2.1 Hybrid Components

DPR In the Dense Passage Retriever (DPR) architecture [Karpukhin et al., 2020], a Transformer encoder $E(\cdot)$ is applied to the query or passage, and the output embedding corresponding to the `[CLS]` token, i.e., $h_0^{(L)}$, is used directly to calculate similarity scores. Since the $h_0^{(L)}$ dense vector is trained to represent passage-level embedding even during the pretraining stage, the DPR model can capture higher level semantics and bridge the vocabulary gap where passages do not share relevant keywords.

To facilitate retrieval efficiency, the similarity is modeled using the dot product between a query q and a passage p:

$$S(q,p) = E(q)^T \cdot E(p). \tag{5.15}$$

SPLADE For sparse representations, the SPLADE models [Formal et al., 2021b,a] serve as effective representatives. These models utilize the sparse output w_i from the pretrained MLM head to represent the (contextually) relevant words in the vocabulary for an input token t_i at position i. Subsequently, they fine-tune this representation for similarity comparison. Similarly, SPLADE models the similarity of two inputs p and q by computing the dot products from their sparse representations:

$$S(q,d) = \eta_q(q)^T \cdot \eta_d(d) \tag{5.16}$$

where $\eta_q(\cdot)$ and $\eta_d(\cdot)$ refer to query and document encoders, respectively.

In particular, the SPLADE-max model extracts similarity features by aggregating from max non-negative weights of relevant words among all input tokens:

$$\eta(t) = \left\{ \max_{i=0,1,\ldots,n} \log(1 + \text{ReLU}(w_{i,j})) \right\}_j^V, \tag{5.17}$$

and the loss in SPLADE models also includes a regularizer, weighted by hyperparameters λ_q and λ_d, which encourages sparse representation by penalizing the query and document encoder FLOPs (floating-point operations).

Query: [CLS] Find all functions $f : \mathbb{R} \to \mathbb{R}$ such that $f(f(x)^2+f(y)) = xf(x) + y$ for all $x, y \in \mathbb{R}$. ...

Relevant doc#1929693 returned by our hybrid search:

[CLS] The function defined by $f(x) = 0$ for all $x \in \mathbb{R}$ is a solution to the functional equation. Suppose that f is some other solution. Then there is a real number ... gives us that $f(x^2+xf(y)) = xf(x+y) = xf(c) = \frac{r}{f(c)} \cdot f(c) = r$ ...

Figure 5.2: Illustration of the hybrid search using an example of Topic A.355 from the ARQMath-3 collection. Different highlighting colors denote different types of semantic matches (orange for passage-level semantics, light red for sparse lexical matches over a threshold of importance, and blue for math formula structure matches where an example of symbol substitution is shown in cyan).

5.2.2 Fine-Tuning

During neural retriever finetuning, a pretrained model (i.e., a *backbone*) is used as the initial encoder state, the retriever is optimized through contrastive triplets, each having a query q and a pair of positive and negative samples, p^+ and p^-.

For all retrievers created in this chapter, a similar practice as Eq. 5.11 is used to fine-tune the model via a contrastive loss $\mathcal{L}_{\text{tune}}$ using a batch of triplet inputs $\{q_i, p_i^+, p_i^-\}_i^{|B|}$, and other instances from the batch are utilized as additional in-batch negatives:

$$\mathcal{L}_{\text{tune}} = -\frac{1}{|B|}\sum_{i=1}^{|B|} \log \frac{\exp\left(S(q_i, p_i^+)\right)}{\sum_{j=1}^{|B|} \exp\left(S(q_i, p_j^+)\right) + \sum_{j=1}^{|B|} \exp\left(S(q_i, p_j^-)\right)}. \tag{5.18}$$

5.2.3 Combining Complementary Models

As illustrated in Figure 5.2, the intuition of hybrid search is to combine different schemes of representations which can be complementary. In particular, I have proposed employing a hybrid search to better capture passage-level, lexical-level, and formula structure semantics,

acknowledging that not all methods are suitable for all modes of data in heterogeneous math documents. This hybrid search is dubbed MABOWDOR – Math-Aware Best-of-Worlds Domain Optimized Retriever.

The MABOWDOR hybrid search includes a structure search to handle math tokens, a single-vector dense retriever to capture passage-level semantics, and a sparse retriever to improve non-math token retrieval:

- For structure search, it aims to capture math formula structure similarity and symbol substitutions efficiently. To make full-text search possible, structure search is complemented with a text scoring scheme, such as BM25+ [Lv and Zhai, 2011], to handle non-math tokens. Both methods are unsupervised.
- For context semantics, a DPR model is fine-tuned on the Coco-MAE backbone and is used for capturing passage-level semantics, reducing the dependence on lexical matches.
- To enhance text retrieval, a text-only SPLADE-max model is optionally added (only in MABOWDOR-full). As the name suggests, we optimize the sparse model in the MIR domain by selectively using dimensions associated to non-math tokens in the representations.

Let Γ be the set of indices that associates with non-math tokens, our text-only SPLADE is trained with the text-only encoder

$$\eta(t) = \left\{ \mathbf{1}_\Gamma(j) \cdot \max_{i=0,1,\ldots,n} \log(1 + \mathrm{ReLU}(w_{i,j})) \right\}_j^V \tag{5.19}$$

where $\mathbf{1}_\Gamma(j)$ is the indicator function that maps j to one if $j \in \Gamma$, and to zero otherwise. Note that we do not mask out math tokens from the input t directly; instead, the masking on the final representation allows the "text-only" SPLADE encoder to peek at math context and use only relevant text words for retrieval.

5.2.4 Fusing Different Relevance Signals

Recent ARQMath tasks have demonstrated that no single-modal search method, whether based on classic retrieval methods, approximate tree matching approaches, or even deep

models with cross encoders, is capable of surpassing others in all effectiveness metrics [Mansouri et al., 2021c]. Therefore, it is an important question to learn how to combine different signals for the best effectiveness.

In the work presented by Peng et al. [2021], a structure-aware Transformer model is trained to rerank results obtained from clustering structure embeddings. However, a supervised retriever is presumably most useful to enhance fuzziness and recall in math retrieval without the constraint of requiring a structure match in candidates.

Indeed, the fusion between a learned score that does not explicitly model structure similarity, and a specialized structure search system designed to particularly uncover structural similarities, has more complementary strengths and achieves higher effectiveness compared to reranking methods [Zhong et al., 2022a,b]. Consequently, our hybrid approach in Zhong et al. [2023] trains supervised retrieval models end-to-end without explicitly filtering structure mismatches.

One way to combine scores from different systems is through a linear interpolation:

$$S_f = \sum_i w_i \cdot S_i \tag{5.20}$$

where i corresponds to the i-th system, and S_i is the document score generated by that system and $w_i \in [0, 1]$ are the parameter weights with the constraint that they are summed to one. Empirical evidence has demonstrated that this fusion scheme is more robust than other alternatives for information retrieval purposes [Bruch et al., 2022, Zhong et al., 2022b].

5.3 Evaluation

5.3.1 Experimental Setup

Benchmarks Our experiments are conducted on ARQMath-2 (2021) [Mansouri et al., 2021c] and ARQMath-3 (2022) [Mansouri et al., 2022]. In the experiments, only the main task (Task 1) topics of the ARQMath datasets are evaluated. This is because the supervised retrievers proposed in this chapter are specifically designed to handle both text and math, focusing on math-aware retrieval.

Evaluation metrics The official evaluation protocols in ARQMath are used (see Section 2.4). In addition to the official effectiveness metrics NDCG', MAP', and P'@10, our

Table 5.1: Different backbones explored in our experiments. All backbones except vanilla BERT are further pretrained to adapt to math domains.

Backbone	**Further pretrained**	**MLM decoder**	**Contrast. loss**	**Swap Psg. Emb.**
BERT (vanilla) [Devlin et al., 2019]	No	No	Yes (NSP)	-
BERT^m [Zhong et al., 2022a]	Yes	No	Yes (NSP)	-
Math ALBERT [Reusch et al., 2022]	Yes	No	Yes (SOP)	-
ORQA^m [Lee et al., 2019]	Yes	No	Yes (ICT)	-
CoCondenser^m [Gao and Callan, 2021a]	Yes	Yes	Yes (ICT)	No
Cot-MAE^m [Wu et al., 2022]	Yes	Yes	No	Yes
Coco-MAE (this chapter)	Yes	Yes	Yes (ICT)	Yes

($*^m$): We use the same pretraining scheme; however, backbone weights are different from their original checkpoints once domain adapted.

experiments also include bpref [Buckley and Voorhees, 2004] in our measurements as it naturally excludes unjudged documents. Compared to NDCG′, we also find MAP′ and bpref are less affected by lower-ranked hits.

Preprocessing Math formulas in LaTeX markup are pre-tokenized into symbols using the PyA0 toolkit [Zhong and Lin, 2021]. The PyA0 lexer reduces input math tokens to a smaller set where semantically identical math tokens (e.g., `\emptyset`, `\varnothing`, or `\empty`) are unified. After pre-tokenization, math tokens are treated the same as text tokens, i.e., each as a single word even for syntactic LaTeX tokens and number digits. Despite being quite simple, this tokenization scheme has been shown to be effective for both math retrieval [Zhong et al., 2022a] and arithmetic calculation [Nogueira et al., 2021].

The input undergoes further tokenization using a word-piece tokenizer before being fed into the models. However, math tokens are not converted to lower case as they likely contain crucial math semantics. For example, an upper-case letter often denotes a matrix. The resulting added math vocabulary size for the tokenizer is about 1,000.

Inputs are handled differently in the structure search and BM25 passes, specifically, math tokens are parsed sequentially into OPT representations where syntactic literals are dropped for structure search, and text words are preprocessed with Porter stemmer.

Further-pretraining As shown in Table 5.1, we consider a list of pretraining schemes for retrievers in our experiments. Except for Math ALBERT [Reusch et al., 2022], they

are all initialized with the same vanilla BERT checkpoint (`bert-base-uncased` from HuggingFace [Wolf et al., 2020]) and most of them are further pretrained for the math domain with added math vocabularies (except vanilla BERT, which is set up to examine whether further pretraining is helpful [Zhu et al., 2021]).

Our data for further pretraining are crawled from MSE and AoPS websites (the training pairs are contained in one passage, no ground-truth pair will be used for supervising model training). For MAE models, we follow Gao and Callan [2021a] and use a 2-layer decoder ($M = 2$) and 6 early layers (fixed $k = 6$). Models further pretrained by us (excluding Math ALBERT) use a batch size of 96 and are trained for a total of 6 epochs. We uniformly use the AdamW [Loshchilov and Hutter, 2017] optimizer with one warm-up epoch from zero to 10^{-4} learning rate linearly.

Fine-tuning We fine-tune retrievers using the ARQMath training data, consisting of Q&A posts prior to the year 2018. Following Zhong et al. [2022b], the training triplets are made of a question, a positive answer sampled from an accepted answer, duplicate questions, or any answer posts to the query receiving more than 7 upvotes. A random answer passage is sampled as a hard negative from a question with a shared tag. A total of 594,000 triplets were sampled for fine-tuning. For comparison, we fix a unique backbone for each type of pretraining architecture and use the same data for each model's fine-tuning process. We additionally trained a ColBERT model [Khattab and Zaharia, 2020] for comparison purposes; it is trained in half-precision to avoid excessive index storage. We used the vanilla ColBERT model due to its simplicity and straightforward integration into our code framework for the purpose of fair comparison.

All dense retrievers are trained using a batch size of 54 using a learning rate 2×10^{-5} with one epoch linear warmup, and all sparse retrievers are trained with a uniform regularizer weight $\lambda_q = \lambda_d = 10^{-4}$ using a batch size of 24 under the same learning rate schedule. The DPR and SPLADE models are fine-tuned with 2.5 epochs by default. However, the ColBERT model is trained for 7 epochs, which helps us to estimate the best neural retrieval effectiveness potential we can obtain from using a multi-vector dense retriever.

Inference For dense retrieval, we evaluated both the exact search (Flat) and HNSW indexes [Malkov and Yashunin, 2018] from the Faiss [Johnson et al., 2019] implementation (M=256, `efConstruction`=16, and `efSearch`=64). The candidate filter stage in the multi-vector representation case (i.e., ColBERT v1 [Khattab and Zaharia, 2020]) is quantized and clustered for efficient approximate nearest-neighbors (ANN) search. For sparse retrieval, we convert the lexical weights to frequencies by scaling and rounding them up, and then index on top of Anserini [Yang et al., 2017]. For unsupervised retrieval, inference is done with the same manually extracted keywords used in Zhong et al. [2022a].

Baseline systems We select representative and effective runs reported on the ARQMath-2 and -3 datasets [Mansouri et al., 2021c, 2022] for comparisons.

For traditional retrieval systems, we include a community-driven oracle baseline extracted from MSE linked posts [Mansouri et al., 2022], a classic tf–idf model from Terrier using Robertson's tf and the standard Sparck Jones' idf [Terrier contributors, 2014], and the most effective BM25+ run generated by the MSM team submitted at ARQMath-3.

Second, we include the most effective structure search submissions to the ARQMath-3 lab: (1) Approach0 [Zhong et al., 2022a] and (2) MathDowsers [Kane et al., 2022] which extracts bags of features from SLT and handles both math and text using BM25.

Third, we make a fair comparison among DPR, SPLADE, and ColBERT retrievers we have built (in the same code framework) on top of our further pretrained Coco-MAE backbone. We also compare a DPR model fine-tuned by our setup but on the TU_DBS Math ALBERT backbone [Reusch et al., 2022].[1] The most effective neural retrievers submitted to ARQMath are included, e.g., the TU_DBS *math_10* run [Reusch et al., 2022] generated by reranking candidates on the entire set of answers using a fine-tuned cross encoder based on ALBERT [Lan et al., 2019]; additionally, the Mini-LM and RoBERTa two-stage retrieval system by the MIRMU team [Geletka et al., 2022], where the Mini-LM is commonly used in some effective cross-domain neural retrievers [Wang et al., 2021b, Thakur et al., 2022].

Lastly, top effective systems using hybrid search are also compared, i.e., (1) the MSM ensemble system using reciprocal rank fusion (RRF) to combine classic lexical retrievers with a domain-adapted Sentence Transformer [Mansouri et al., 2022, Novotnỳ et al., 2021]; (2) the MSM + MIRMU ensemble run integrated with the two-stage MIRMU run in a post-hoc experiment [Geletka et al., 2022]; (3) the Approach0 fusion run from combining Approach0 and a fine-tuned ColBERT model [Zhong et al., 2022a]. And finally, (4) a concatenated run from the best runs of the MSM team and the Approach0 team, which achieves the best-published scores on the ARQMath-3 dataset to date. Note that the ARQMath-2 scores in this concatenated run are unknown because it was created by a third team in post-hoc experiments [Kane et al., 2022].

5.3.2 Main Results

Tables 5.2 and 5.3 summarizes our main results. We include a column that indicates whether a corresponding system can practically generate the run using only a single CPU

[1] We are aware of other existing MIR backbones [Geletka et al., 2022, Peng et al., 2021]; however, either they do not pretrain a passage-level embedding or the checkpoints are not available for our evaluation.

Table 5.2: Systems compared on the ARQMath-2 dataset using the official metrics. Whether or not a run can be practically retrieved using a single CPU is indicated in the last column. Hybrid search scores are reported using the linear interpolation parameters tuned on 5-fold cross validations.

Row	System	ARQMath-2			1-CPU
		NDCG'	MAP'	P'@10	
	Baseline / Traditional IR				
1	Community Linked	0.203	0.120	0.282	-
2	tf–idf	0.185	0.046	0.063	✓
3	BM25+	0.285	0.082	0.116	✓
	Structure search				
4	Approach0 (tree matching)	0.383	0.185	0.241	✓
5	MathDowsers (bag of struct. feats.)	0.510	0.223	0.265	✓
	Neural retrievers				
6	Our DPR (Math ALBERT, Flat index)	0.315	0.118	0.189	✓
7	TU_DBS math_10 cross encoder	0.454	0.228	**0.321**	✗(‡)
8	MIRMU Mini-LM and RoBERTa	0.487	0.233	<u>0.316</u>	✗(‡)
9	Our DPR (Coco-MAE, HNSW index)	0.403	0.165	0.238	✓
10	Our DPR (Coco-MAE, Flat index)	0.416	0.173	0.247	✓
11	Our SPLADE-max (Coco-MAE, text-only)	0.408	0.171	0.209	✓
12	Our ColBERT (Coco-MAE)	0.441	0.198	0.251	✗(†)
	Hybrid systems				
13	MSM ensemble	0.381	0.119	0.152	✓
14	MSM + MIRMU ensemble (*)	0.467	0.186	0.261	✗(‡)
15	Approach0_fusion	0.460	0.226	0.296	✗(†)
16	MSM ensemble + Approach0_fusion (*)	-	-	-	✗(†)
	Our hybrid search				
17	MABOWDOR (base, HNSW index)	<u>0.515</u>	0.250	0.306	✓
18	MABOWDOR (base, Flat index)	0.514	<u>0.254</u>	0.307	✓
19	MABOWDOR (full, Flat index)	**0.527**	**0.265**	<u>0.316</u>	✓

(∗): Systems directly tuned on the datasets in a post-hoc experiment.
(‡): Not practical for single-CPU search due to using a cross-encoder reranker.
(†): Depending on an expensive multi-vector dense retriever, i.e., ColBERT (v1).

Table 5.3: Systems compared on the ARQMath-3 dataset using the official metrics. Whether or not a run can be practically retrieved using a single CPU is indicated in the last column. Hybrid search scores are reported using the linear interpolation parameters tuned on 5-fold cross validations.

Row	System	ARQMath-3			1-CPU
		NDCG'	MAP'	P'@10	
	Baseline / Traditional IR				
1	Community Linked	0.112	0.054	0.177	-
2	tf–idf	0.272	0.064	0.124	✓
3	BM25+	0.396	0.122	0.194	✓
	Structure search				
4	Approach0 (tree matching)	0.397	0.159	0.271	✓
5	MathDowsers (bag of struct. feats.)	0.474	0.164	0.247	✓
	Neural retrievers				
6	Our DPR (Math ALBERT, Flat index)	0.394	0.147	0.260	✓
7	TU_DBS math_10 cross encoder	0.436	0.158	0.263	✗(‡)
8	MIRMU Mini-LM and RoBERTa	0.498	0.184	0.267	✗(‡)
9	Our DPR (Coco-MAE, HNSW index)	0.460	0.194	0.319	✓
10	Our DPR (Coco-MAE, Flat index)	0.464	0.192	0.324	✓
11	Our SPLADE-max (Coco-MAE, text-only)	0.472	0.179	0.287	✓
12	Our ColBERT (Coco-MAE)	0.490	0.202	0.310	✗(†)
	Hybrid systems				
13	MSM ensemble	0.504	0.157	0.241	✓
14	MSM + MIRMU ensemble (*)	<u>0.576</u>	0.214	0.309	✗(‡)
15	Approach0_fusion	0.508	0.216	0.345	✗(†)
16	MSM ensemble + Approach0_fusion (*)	**0.594**	0.234	0.345	✗(†)
	Our hybrid search				
17	MABOWDOR (base, HNSW index)	0.536	<u>0.241</u>	<u>0.381</u>	✓
18	MABOWDOR (base, Flat index)	0.534	0.239	<u>0.381</u>	✓
19	MABOWDOR (full, Flat index)	0.553	**0.246**	**0.386**	✓

(∗): Systems directly tuned on the datasets in a post-hoc experiment.
(‡): Not practical for single-CPU search due to using a cross-encoder reranker.
(†): Depending on an expensive multi-vector dense retriever, i.e., ColBERT (v1).

– we use this resource requirement to estimate system expense in efficiency because many system runs do not have query latency reported in Mansouri et al. [2022].

We have named two optimal configurations in our hybrid search, i.e., MABOWDOR-base and MABOWDOR-full. The MABOWDOR-base setting uses structure search for math, BM25+ for text, and dense representations produced by DPR for passage-level retrieval; the MABOWDOR-full configuration additionally includes a sparse SPLADE-max model, which encodes only text tokens (see Eq. 5.19). More specifically, MABOWDOR-base linearly combines systems at row 4 and 9 or row 4 and row 10,[2] depending on the index scheme; and MABOWDOR-full further combines the SPLADE-max run at row 11.

As shown in Tables 5.2 and 5.3, traditional retrieval falls short in the MIR domain – around 3 times less effective than our hybrid search in the worst case (comparing rows 3 and 19). This has to do with the inability of traditional bag-of-words models to capture sequential, structural semantics and symbol substitutions. Neural retrievers (rows 6 to 12), in general, achieve a higher level of effectiveness similar to that of structure search methods (rows 4, 5), but many are impractical for running in a single-CPU inference environment. Hybrid systems (rows 13 to 19), on the other hand, show better overall effectiveness. Our MABOWDOR hybrid models achieve better effectiveness. In the best case, MABOWDOR-full outperforms the previous best submission in ARQMath-3 (i.e., Approach0_fusion) by 9% in NDCG′, 14% in MAP′, and 12% in P′@10.

From the efficiency and model complexity perspective, the MSM ensemble runs at row 13 or 16 in Tables 5.2 and 5.3 are contributed by at least 4 systems, and the MSM + MIRMU run at row 14 consists of a total of 9 different systems [Geletka et al., 2022]. Although Approach0 fusion at row 15 has only 3 subsystems, its practical adoption is limited by its dependency on multi-vector dense representations, which result in either an overly large index (costing ∼80 GiB for indexing the ARQMath corpus in our case) or creating a complex scoring pipeline [Santhanam et al., 2021, 2022a]. Systems using a cross encoder or depending on an expensive multi-vector dense retriever are considered to have an efficiency bottleneck here because a query can take minutes to run on a single CPU. In contrast, our MABOWDOR-base hybrid search only requires a single learned component (i.e., DPR) while achieving much faster run times when using an HNSW index, i.e., (avg, max, med, std) = (0.40, 1.03, 0.36, 0.22) seconds; the run times of the structure search are (0.52, 1.68, 0.42, 0.31) seconds. All run times are reported using in-memory and single-threaded retrieval on an Intel mid-range i5-8600K CPU with DDR4 memory.

[2]We do not use MathDowsers for structure search because its updated version takes more than 10 seconds on average per query [Kane et al., 2022].

Table 5.4: Ablations on MABDOWDOR components and the standalone sparse retriever, evaluated on ARQMath-3. Linear interpolation is tuned using cross-validation. Underlined ($p < 0.01$) and italic ($p < 0.05$) are considered significantly different from the non-ablated case using the two-tailed pairwise t-test.

#	Ablations	NDCG'	MAP'	P'@10
1	**MABDOWDOR (full)**	**0.553**	**0.246**	**0.386**
2	minus sparse vector (text only)	0.534	0.239	0.381
3	using text + math sparse vector	0.542	0.240	0.376
4	minus dense vector	0.534	0.224	0.358
5	minus structure search	0.501	0.209	0.322
6	**Sparse vect. (Coco-MAE, text-only)**	**0.472**	**0.179**	**0.287**
7	using text + math sparse vector	0.404	0.148	*0.256*

5.3.3 Analysis

Is each component of hybrid search necessary? What are their best scope and combination?

As shown in Table 5.4, we break down our hybrid search to study its source of effectiveness. Rows 1, 2, 4, and 5 indicate that the structure search and dense representation are more effective components than when each is combined with sparse retrieval. Each component of the hybrid search is necessary since removing any component will lead to a significant drop in both NDCG′ and MAP′. In particular, the structure search is crucial to be incorporated for good top precision – otherwise, the P′@10 will be significantly lower in row 5. These results justify our hybrid search components and indicate that the structure search and dense representation used in our proposed MABOWDOR-base hybrid search are complementary at the same time using a minimal number of components.

Interestingly, in both hybrid search and standalone scenarios – according to rows 1, 3, 6, and 7 in Table 5.4 – the sparse representation is more effective when it is only used to represent text tokens instead of being used to weigh math tokens. To understand why the full-scope sparse vector is sub-optimal, we have probed some sample search results and found that math tokens tend to be commonly shared by many contents from different topics, and including math token weights in lexical retrieval causes many false positives if the model cannot correctly recognize the topic of a post (an example is shown in Figure 5.3).

To further discover the best scope of each component, we disassemble the topics into two important dimensions in MIR, i.e., the formula search topics and the heterogeneous

Q: What is the Fourier transform of function $f(x) = \frac{1}{|x|}$?

SPLADE (text + math), doc#1125280 (irrelevant):

... $\ln|x|$ is defined for all $x \neq 0$. It derivative $\frac{1}{x}$ for all $x \neq 0$, so $\int \frac{1}{x}\,dx = \ln|x| + C$.

Top scored tokens: <vert>, <frac>, <x>, absolute, <ln>.

SPLADE (text only), doc#1085035 (relevant):

In general, $1/f$ may not have a Fourier transform, even in the sense of tempered distributions. ... And similarly for $1/f(t)$ where we use the partial fraction expansion of $1/f$.

Top scored tokens: fourier, f, reciprocal, transform, fraction.

Figure 5.3: Example question from the ARQMath-3 dataset (Topic A.360) and the retrieved passages by SPLADE models using all and text-only representations, respectively. Tokens wrapped in angle brackets are math tokens.

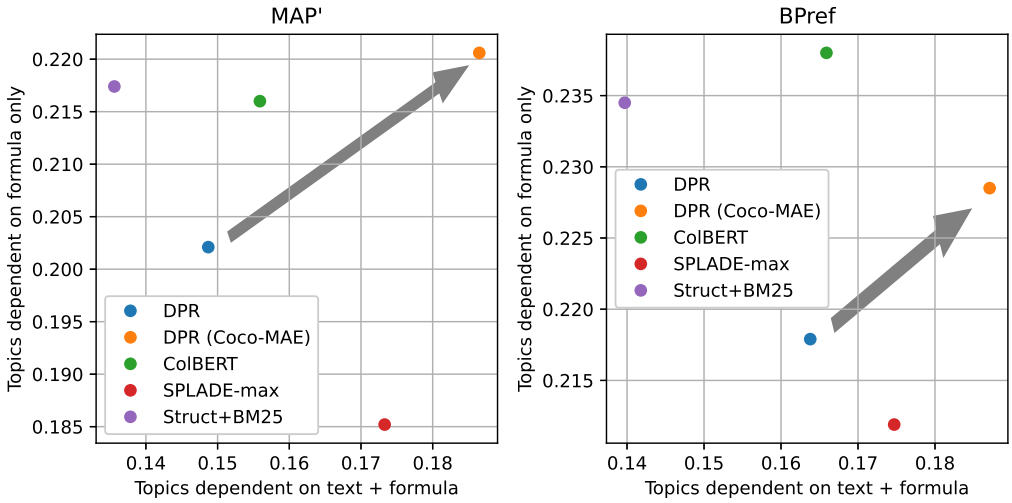


Figure 5.4: MAP' and bpref score divided into two different topic dependency dimensions. The upper points are better at formula search topics while the points on the right are better at heterogeneous topics that depend on both text and formula(s).

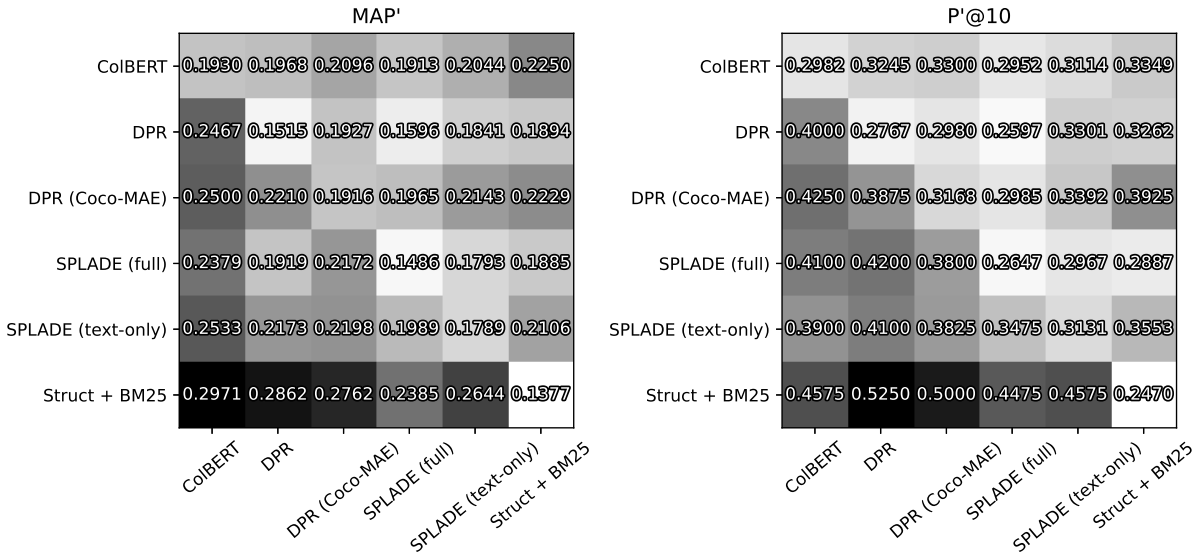


Figure 5.5: The pair-wise fusion test score matrices from 5-fold cross-validation in MAP' and P'@10. The lower triangular and diagonal elements are scores evaluated for topics that depend on formula and text, and the upper triangular elements are evaluated for formula search topics. Score are shown in grids where their background grey scale is min-max normalized in each measurement.

topics (the former are formula-centered, and the latter further require capturing contextual semantics around formulas); both categories are provided in the official ARQMath datasets. In Figure 5.4, we show a scatter plot to illustrate the different strengths of various methods. We can see that structure search excels at formula search topics, dense representations (either multi-vector or single-vector) can be good at both dimensions, and the lexical sparse model is better at handling heterogeneous topics with more non-math tokens. This also helps to attribute the success of combining structure search as the Coco-MAE backbone boosts DPR retriever effectiveness in both dimensions, especially for heterogeneous topics where structure search is performing poorly. Therefore, our dense representation complements structure search effectively.

Lastly, to confirm which two components demonstrate the best hybrid effectiveness, we conduct a pair-wise fusion among different methods and consider the two kinds of topics individually (see Figure 5.5). The results echo our findings that the structure search and dense retrievers are the most complementary combinations, especially for heterogeneous topics. This suggests that passage-level representations are greatly beneficial to discover semantic connections in text-formula contexts, where unsupervised lexical matching and structure search are inadequate. In addition, data-driven methods in general can complement unsupervised retrieval by bridging vocabulary mismatch; this is useful in math-aware retrieval if math expressions have different structures or symbols.

Interestingly, DPR based on pretrained vanilla BERT is on par with its counterpart based on Coco-MAE when combined with structure retrieval at heterogeneous topics (at the bottom rows in Figure 5.5) while it performs not as well at formula search topics (at the right-most columns in Figure 5.5). This suggests that the injected token-level information through the MLM decoder task of the Coco-MAE pretraining is most helpful in formula search when complementing structure search.

Results in Figure 5.5 show that dense representation improves effectiveness when combined with structure search more than when it is combined with learned sparse representation (i.e., from the SPLADE model). As a result, the MABOWDOR-base model is regarded as a performant and cost-effective MIR model although an additional sparse model could add more effectiveness.

How do dense retrieval and structure search mutually benefit each other?

To explain the significant effectiveness boosts obtained from combining structure and dense retrieval, it has been found that the improvement primarily stems from reduced false positive rates, as a result of the mutual compatibility between these two methods.

To support this conclusion, I initially select the top-10 topics that achieve the highest MAP' boost in the fusion run, compared to either the structure search or the dense retrieval

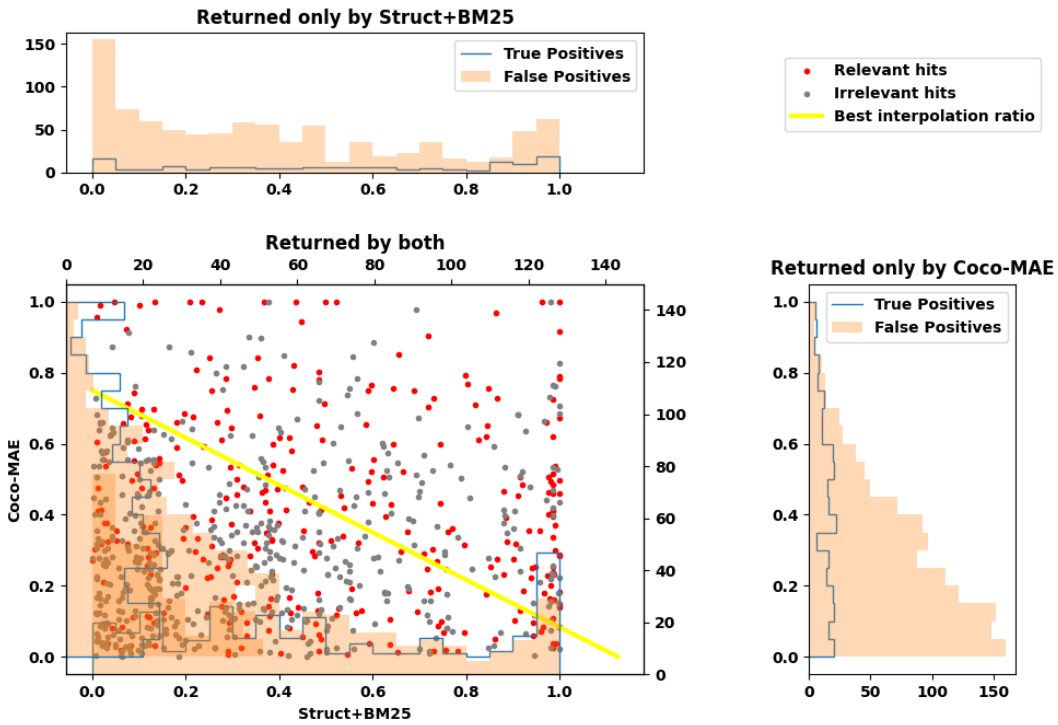


Figure 5.6: Relevant and non-relevant hits returned by complementary systems. Intersected hits with their ranking scores and histograms are in the scatters plot, and hits returned by only one system is illustrated in a side histogram in the corresponding axis.

Table 5.5: Ablations on the MABDOWDOR-(base), sparse, multi- and single-dense representations using different backbones. All hybrid scores are cross-validated results. Underlined ($p < 0.01$) and italic ($p < 0.05$) are considered significantly different from the non-ablated case using the two-tailed pairwise t-test.

#	Ablations	NDCG'	MAP'	P'@10
1	**MABDOWDOR (base)**	**0.534**	**0.239**	**0.381**
2	replace backbone w/ ORQA^m	0.526	0.228	*0.345*
3	replace backbone w/ Cot-MAE ^{m}	**0.534**	0.236	0.369
4	replace backbone w/ CoCondenser^m	0.526	0.230	0.367
5	replace backbone w/ BERT ^{m}	0.507	*0.219*	0.367
6	replace backbone w/ BERT (vanilla)	0.493	0.202	*0.340*
7	**Sparse vect. (Coco-MAE, text-only)**	**0.472**	**0.179**	**0.287**
8	replace backbone w/ BERT ^{m}	0.464	0.176	**0.287**
9	**Multiple dense vect. (Coco-MAE)**	**0.490**	**0.202**	**0.310**
10	replace backbone w/ BERT ^{m}	0.416	0.168	**0.310**
11	**Single dense vect. (Coco-MAE)**	**0.464**	**0.192**	**0.324**
12	replace backbone w/ BERT ^{m}	0.408	0.164	0.297

alone. Subsequently, I create a scatter graph that depicts the fusion hits and no-intersection hits. As shown in Figure 5.6, the results returned by structure search alone has a higher false positive rate compared to dense retrieval. However, when a structure hit is also found by dense retriever, the false positive rate (especially at the top results) greatly reduces. A similar pattern also happens in the dense retrieval pass except the no-intersection hits generated by the dense retriever alone already has a good precision among the top results. This is presumably attributed to the good context-level semantic matching ability from dense retrieval (compared to focusing on only structure features); by combining it with structure search, the relevant hits get further boosted to a higher ratio in the top results.

Is the pretraining scheme we propose beneficial? In what way does it benefit the most in our hybrid search?

Another ablation result is shown in Table 5.5, to study the benefits of using different pretraining methods. By comparing rows 1 – 6, we see that the Coco-MAE backbone is the most effective one among all the considered backbones. Moreover, in all types of representations, our proposed pretraining scheme boosts the downstream MIR tasks in NDCG′ and MAP′ scores. When compared to retrievers on top of a domain-adapted

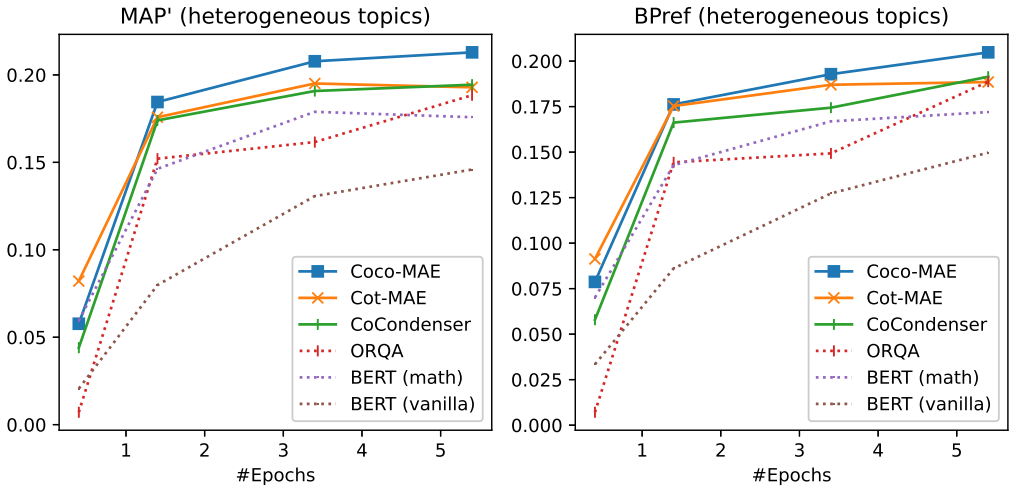


Figure 5.7: Change of MAP' and bpref scores for the DPR model fine-tuning on different backbones and at different training stage on the ARQMath-2 dataset. Scores are measured by topics that depend on both text and formula (i.e., heterogeneous topics). The dashed lines denote models without a decoder.

BERT backbone (row 1, and rows 5 – 12), Coco-MAE improves NDCG′ and MAP′ scores significantly ($p < 0.05$ at least) except for the sparse representation case.

Apart from being sensitive to input scope as we have observed in Table 5.4, the sparse representation is affected less by the replacement of the backbone (shown by row 7 and 8 in Table 5.5), which is expected because the sparse representation almost never depends on the passage-level code. From another perspective, the multi-vector representations from ColBERT at rows 9 and 10 suffer the largest degradation by switching the backbone from Coco-MAE to BERT^m, even if both are further pretrained for the math domain. The single dense representation also suffers more than the sparse representation does, by comparing rows 7, 8 and 11, 12. These imply that our enhancement to the bottleneck code during Coco-MAE pretraining boosts dense representations in general and improves retrievers relying on passage-level representations, whereas the sparse retriever effectiveness is instead mostly dependent on the encoder MLM objective and thus is less affected because both Coco-MAE and BERT^m also have this MLM task during pretraining.

Admittedly, the effectiveness gains for the Coco-MAE backbone compared to a few competitive alternatives in rows 3 and 4 are not significant. However, we find that the Coco-MAE backbone has been shown consistently better effectiveness in heterogeneous

topics during different stages of training (see Figure 5.7) – notably better than other backbones without an MLM decoder. This indicates our pretraining method achieves better training efficiency for heterogeneous topics that require a contextualized understanding of both math and text content, which creates merits to be used for retrieval in the MIR domain in particular. As shown in Figure 5.7, the Coco-MAE, Cot-MAE^m, and CoCondenser^m backbones all offer distinctly overall better effectiveness during fine-tuning, highlighting an advantage to having an MLM decoder architecture compared to the ORQA^m and BERT backbones. On the other hand, the ORQA^m backbone can nevertheless achieve a comparable effectiveness level towards the end of the 6th epoch, which indicates that a pretraining objective that better mimics the retrieval task (i.e., ICT) is also critical.

In general, the consistent training efficiency seen in Figure 5.7 validates that the swap of in-context passage embeddings during pretraining has further boosted the quality of encoding formula-text semantics. This benefits the heterogeneous data retrieval presumably because we have created a more challenging pretraining task that predicts context words from relevant formulas and vice versa.

What are the relative importance of the two unsupervised methods?

The unsupervised retrieval described so far contains two methods, i.e., structure search for math formulas and a bag-of-words model for retrieving regular text words using BM25+ (see Section 3.2.3). It is also interesting to understand the individual importance of each unimodal method and how different ratios of these individual methods affect the complementary effectiveness when combined with dense retrieval.

As a result, the unsupervised retrieval is further broken down into pure structure search targeting only math formulas and text search using only the text keywords. Table 5.6 summarizes the effectiveness result for each unimodal component as well as when they are combined with a dense retriever (D). In the case of math-only search (M), a query will be skipped if it contains only text keywords, similarly, a query will be skipped if it contains only math formulas in text-only search (T).

As shown in Table 5.6, row 3 and 4 represent the most effective cases when we solely consider unsupervised search by adjusting the interpolation weights between math-only and text-only search, with zero contribution from the dense retriever (row 1–5). Interestingly, math weight and text weight are even when they achieve the highest precision score. This finding highlights the equal importance of both formula and text matching in our unsupervised retrieval approach. It suggests that considering both mathematical formulas and textual content is crucial for achieving effective retrieval results.

However, when the data-driven retriever is introduced (i.e., 6 – 15 rows associated to non-zero dense retriever weights), the unsupervised text retrieval becomes the least

Table 5.6: The effectiveness evaluated on the ARQMath-3 dataset by performing a grid search for linear interpolation weights. The interpolation weights associated with the scores from the dense retriever with a Coco-MAE backbone, the structure search using math keywords only, and the text-only retrieval using BM25+ are represented by **D**, **M**, and **T** respectively. Bold and underlined are the first and second highest scores for each metric in each block. Results are generated by merging top-1000 results individually.

Row	D	M	T	NDCG'	MAP'	P@10	bpref
1	0.00	0.00	1.00	0.304	0.095	0.174	0.104
2	0.00	0.25	0.75	0.384	0.134	0.246	0.145
3	0.00	0.50	0.50	**0.396**	<u>0.149</u>	**0.269**	**0.162**
4	0.00	0.75	0.25	<u>0.395</u>	**0.152**	<u>0.258</u>	<u>0.161</u>
5	0.00	1.00	0.00	0.328	0.121	0.212	0.137
6	0.25	0.00	0.75	0.454	0.168	0.255	0.160
7	0.25	0.25	0.50	0.505	0.204	0.315	0.194
8	0.25	0.50	0.25	<u>0.517</u>	<u>0.219</u>	<u>0.355</u>	<u>0.210</u>
9	0.25	0.75	0.00	0.475	0.194	0.300	0.187
10	0.50	0.00	0.50	0.488	0.197	0.342	0.190
11	0.50	0.25	0.25	**0.531**	**0.229**	**0.368**	**0.223**
12	0.50	0.50	0.00	0.494	0.210	0.349	0.206
13	0.75	0.00	0.25	0.497	0.205	0.340	0.200
14	0.75	0.25	0.00	0.499	0.211	0.340	0.206
15	1.00	0.00	0.00	0.464	0.192	0.324	0.192

important because the best scores are now achieved by rows at 8 and 11, where text search has the smallest weight in either row. Presumably, when dense retrieval is introduced, it is able to replace some unsupervised text-based retrieval capabilities. Consistent with our expectations, the overall good effectiveness observed from rows 10-15, where the math-only retrieval has a substantial weight in the interpolation (at least 0.5), further emphasizes the importance of structure search and its continued vital role in obtaining effective supervised math-aware retrieval results.

Chapter 6

Conclusion and Future Work

Unlike traditional information retrieval, math information retrieval (MIR) poses unique challenges due to its special characteristics. In this thesis, I have made significant advancements in MIR by proposing and combining two efficient and complementary methods: structure search and contextual supervised retrieval based on Transformer models.

For previous work in MIR (reviewed in Section 2.3), the structure search has often been costly and involved extensive feature engineering. However, the reviewed structure search in this thesis is not only efficient but also having simple structure features. It utilizes leaf-root paths to uncover meaningful and well-defined structure matches (see Section 3.1) which can be understood and visualized in search results. Furthermore, the single-tree matching structure search objective can be accelerated using specialized inverted index and dynamic pruning optimizations. These optimizations require minimal changes to be compatible with the traditional IR architecture. Notably, the proposed structure search approach achieves top-effective and real-time math formula retrieval [Zhong et al., 2020].

In addition, a hybrid search method using both structure search and supervised retrieval is proposed with sub-second run times in each component and has achieved the state-of-the-art effectiveness on ARQMath datasets [Zhong et al., 2023]. Combining structure search with a supervised retriever in an end-to-end pipeline proves highly effective for math answer retrieval. This is because the Transformer model uncovers contextual connections and overcomes the issue of imposing too many lexical or structural constraints over search candidates. On the other hand, structure similarity in math formulas has a strong relevance signal that leads to the state-of-the-art unsupervised MIR search. By combining these methods, different semantic aspects in math documents can be captured, offering a cost-effective solution for addressing MIR challenges.

Looking to the future, continuous advancements in math retrieval will be crucial: First, MIR also requires image or multi-modal retrieval to handle topics like geometry and topology. Second, a query formula structure needs to be transformed to recall other mathematically equivalent formulas but in different structures. Lastly, it is an ongoing endeavor to continuously improve the modeling of math language and its representation. This is because math language comprises intricate reasoning and abstract concepts that arguably represent a pinnacle of intellectual sophistication.

Previously, these challenges were difficult to address. However, with the recent rapid advancements in large language models (LLMs), I believe we can potentially tackle these challenges in a progressively unified manner. Particularly, LLMs have the potential to serve as a "central processing unit" capable of handling multi-modality representations [Li et al., 2023, Driess et al., 2023], they can also function as retrieval-augmented generative models, extracting more precise information from the results obtained by first-stage retrievers [Guu et al., 2020, Izacard and Grave, 2020, Borgeaud et al., 2022, Shi et al., 2023]. Recently, multi-hop query refinement has been developed to further break down an input question, to ask follow-up questions, and to gather existing retrieval results [Yao et al., 2022, Khattab et al., 2022]. In the case of MIR, this signifies the great potential to transform an unfamiliar form of a math formula into a more commonly recognized identity that is more suitable for use as a structure search query. Furthermore, we will have the ability to automatically select best formula(s) for structure search to enhance the effectiveness of math-aware retrieval. Lastly, thanks to the significant advancements in the mathematical abilities of LLMs [Rae et al., 2021, Lewkowycz et al., 2022, Bubeck et al., 2023], we are witnessing a promising pathway to directly address math problems or subproblems with the assistance of these models.

Enlightened by these advancements, I have done a *preliminary* experiment to evaluate the impact of these technologies on math-aware retrieval. For this experiment, I utilized the most advanced, yet proprietary, OpenAI APIs available at the time of writing. For the baseline systems, I select the most high-performing dense retriever and hybrid search system described in this thesis, namely the DPR (Dense Passage Retrieval) based on the Coco-MAE backbone, and the MABOWDOR-base model, which achieves the optimal effectiveness while maintaining a balanced level of efficiency. I rerank the full top-1000 results using the cheaper but up-to-date embedding API from OpenAI[1] ($0.0001 / 1K tokens), and for the evaluation of GPT-4 model[2] ($0.06 / 1K output tokens), I limit the reranking to top-10 results generated from baseline systems.

[1] Ada-002 model, details are available at: https://platform.openai.com/docs/models/embeddings

[2] See https://platform.openai.com/docs/models/gpt-4.

To generate reranked order using the GPT-4 generative model, I employ the following prompt template:

```
Sort the list PASSAGES by how good each text answers the QUESTION (in
descending order of relevancy).

QUESTION = "...", PASSAGE1 = "...", PASSAGE2 = "...", ..., PASSAGE10
= "...".

PASSAGES = [PASSAGE1, PASSAGE2, PASSAGE3, PASSAGE4, PASSAGE5, PASSAGE6,
PASSAGE7, PASSAGE8, PASSAGE9]

SORTED_PASSAGES = [
```

As shown by Table 6.1 and 6.2, the best dense retriever proposed in this thesis is slightly more effective than the Ada-002 embedding. Admittedly, the proposed DPR model is especially fine-tuned on the math domain, and the Ada-002 embedding can be used for general text retrieval; however, both the pretraining and fine-tuning of the Coco-MAE and DPR retriever consume minimal data and budget (using the 110M-parameter `bert-base` model and fine-tuned only against the Math StackExchange website with half-million sample pairs). On the other hand, the hybrid search combining dense retrieval and structure search is shown to be highly effective, outperforming Ada-002 and on par with GPT-4 in top-10 precisions. However, GPT-4, despite being a generalist model [Bubeck et al., 2023], can boost the effectiveness of math retrieval in other cases. [3]

We are currently witnessing an exciting era where language models like GPT-4 continue to astound us with their capabilities. The potential of a generative approach to directly or even fully handle queries in the field of Math Information Retrieval looks more promising than ever. Nevertheless, I believe that the cost-effective retrieval methods introduced in this thesis, based on structure similarity and hybrid search, will play a vital role in serving as a first-stage efficient retriever or as a complementary tool for LLMs [Schick et al., 2023, Shen et al., 2023, Patil et al., 2023, Hao et al., 2023, Lu et al., 2023]. Moreover, in order to address the remaining challenges in this domain, it is reasonable to expect the emergence of more automatic and closed-loop learning approaches. I believe these approaches will effectively and interactively integrate math retrieval models, leading to further enhanced math answering performance.

[3] However, we cannot rule out the possibility that these proprietary LLMs are trained on the MSE dataset that we are using for testing.

Table 6.1: Evaluation of the OpenAI reranked results on the ARQMath-3 (Task 1) using DPR (Coco-MAE, HNSM indexed) as base run. The up-to-date best available embedding model Ada-002 and generative model GPT-4 are compared.

Row	Run	NDCG'	MAP'	P'@10	bpref
	Top-1000 reranking				
(1)	DPR (Coco-MAE)	**0.460**	**0.194**	**0.319**	0.195
(2)	Ada-002	0.453	0.190	0.309	**0.202**
	Top-10 reranking				
(3)	DPR (Coco-MAE)	0.108	0.048	**0.236**	0.062
(4)	Ada-002	0.112	0.044	0.235	0.063
(5)	GPT-4	**0.121**	**0.059**	0.229	**0.068**

Table 6.2: Evaluation of the OpenAI reranked results on the ARQMath-3 (Task 1) dataset using MABOWDOR-base (HNSM indexed) as base run. The up-to-date best available embedding model Ada-002 and generative model GPT-4 are compared.

Row	Run	NDCG'	MAP'	P'@10	bpref
	Top-1000 reranking				
(1)	MABOWDOR-base	**0.536**	**0.241**	**0.381**	**0.232**
(2)	Ada-002	0.497	0.205	0.318	0.208
	Top-10 reranking				
(3)	MABOWDOR-base	0.132	0.063	**0.330**	0.088
(4)	Ada-002	0.121	0.049	0.248	0.069
(5)	GPT-4	**0.153**	**0.079**	0.321	**0.092**

References

Muhammad Adeel, Hui Siu Cheung, and Sikandar Hayat Khiyal. MATH GO! prototype of a content based mathematical formula search engine. *Journal of Theoretical & Applied Information Technology*, 2008. URL http://www.jatit.org/volumes/research-papers/Vol4No10/15Vol4No10.pdf.

Qingyao Ai, Liu Yang, Jiafeng Guo, and W. Bruce Croft. Analysis of the paragraph vector model for information retrieval. In *ICTIR*, 2016. URL https://dl.acm.org/doi/pdf/10.1145/2970398.2970409.

Akiko Aizawa. An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, 2003. URL https://www.sciencedirect.com/science/article/pii/S0306457302000213.

Akiko Aizawa, Michael Kohlhase, and Iadh Ounis. NTCIR-10 Math Pilot Task Overview. In *NTCIR*, 2013. URL http://ntcir-math.nii.ac.jp/wp-content/blogs.dir/23/files/2013/10/01-NTCIR10-OV-MATH-AizawaA.pdf.

Akiko Aizawa, Michael Kohlhase, Iadh Ounis, and Moritz Schubotz. NTCIR-11 Math-2 Task Overview. In *NTCIR*, 2014. URL https://research.nii.ac.jp/ntcir/workshop/OnlineProceedings11/pdf/NTCIR/OVERVIEW/01-NTCIR11-OV-MATH-AizawaA.pdf.

Miltiadis Allamanis, Pankajan Chanthirasegaran, Pushmeet Kohli, and Charles Sutton. Learning continuous semantic representations of symbolic expressions. In *ICML*, 2017. URL http://proceedings.mlr.press/v70/allamanis17a/allamanis17a.pdf.

Vo Ngoc Anh and Alistair Moffat. Pruned query evaluation using pre-computed impacts. In *SIGIR*, 2006. URL https://dl.acm.org/doi/10.1145/1148170.1148235.

Michal Siedlaczek Antonio Mallia and Torsten Suel. An experimental study of index compression and DAAT query processing methods. In *ECIR*, 2019. URL https://link.springer.com/chapter/10.1007/978-3-030-15712-8_23.

Andrea Asperti, Ferruccio Guidi, Claudio Sacerdoti Coen, Enrico Tassi, and Stefano Zacchiroli. A content based mathematical search engine: Whelp. In *TYPES*. Springer, 2006. URL https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=1e0f2b0a1625150fba154f2983af06108e82c37e.

Avinash Atreya and Charles Elkan. Latent semantic indexing (LSI) fails for TREC collections. *SIGKDD*, 2011. URL https://dl.acm.org/doi/pdf/10.1145/1964897.1964900.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv:1607.06450*, 2016. URL https://arxiv.org/abs/1607.06450.

Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. *arXiv:1809.10853*, 2018. URL https://arxiv.org/abs/1809.10853.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*, 2014. URL https://arxiv.org/abs/1409.0473.

Yang Bai, Xiaoguang Li, Gang Wang, Chaoliang Zhang, Lifeng Shang, Jun Xu, Zhaowei Wang, Fangshan Wang, and Qun Liu. SparTerm: Learning term-based sparse representation for fast text retrieval, 2020. URL https://arxiv.org/abs/2010.00768.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. MS MARCO: A human generated machine reading comprehension dataset. *arXiv:1611.09268*, 2016. URL https://arxiv.org/abs/1611.09268.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm. Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *ICML*. PMLR, 2022. URL https://proceedings.mlr.press/v162/borgeaud22a/borgeaud22a.pdf.

Andrei Z Broder, David Carmel, Michael Herscovici, Aya Soffer, and Jason Zien. Efficient query evaluation using a two-level retrieval process. In *CIKM*, 2003. URL https://dl.acm.org/doi/pdf/10.1145/956863.956944.

Sebastian Bruch, Siyu Gai, and Amir Ingber. An analysis of fusion functions for hybrid retrieval. *arXiv:2210.11934*, 2022. URL https://arxiv.org/abs/2210.11934.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of Artificial General Intelligence: Early experiments with GPT-4. 2023. URL https://www.microsoft.com/en-us/research/publication/sparks-of-artificial-general-intelligence-early-experiments-with-gpt-4.

Chris Buckley and Ellen M. Voorhees. Retrieval evaluation with incomplete information. In *SIGIR*, 2004. URL https://dl.acm.org/doi/10.1145/1008992.1009000.

Fredrik Carlsson, Amaru Cuba Gyllensten, Evangelia Gogoulou, Erik Ylipää Hellqvist, and Magnus Sahlgren. Semantic re-tuning with contrastive tension. In *ICLR*, 2020. URL https://www.diva-portal.org/smash/get/diva2:1684806/FULLTEXT01.pdf.

Wei-Cheng Chang, Felix X Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. Pre-training tasks for embedding-based large-scale retrieval. *arXiv:2002.03932*, 2020. URL https://arxiv.org/abs/2002.03932.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv:1604.06174*, 2016. URL https://arxiv.org/abs/1604.06174.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *ICML*. PMLR, 2020. URL http://proceedings.mlr.press/v119/chen20j/chen20j.pdf.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv:1409.1259*, 2014. URL https://arxiv.org/abs/1409.1259.

W. Bruce Croft and David J. Harper. Using probabilistic models of document retrieval without relevance information. *Journal of documentation*, 1979. URL https://www.emerald.com/insight/content/doi/10.1108/eb026683/full/html.

W. Bruce Croft, Donald Metzler, and Trevor Strohman. *Search engines: Information retrieval in practice*. Addison-Wesley Reading, 2010. URL https://www.academia.edu/download/30740463/z2009_2465.pdf.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained Transformers. *arXiv:2104.08696*, 2021. URL https://arxiv.org/abs/arXiv:2104.08696.

Yifan Dai, Liangyu Chen, and Zihan Zhang. An N-ary tree-based model for similarity evaluation on mathematical formulae. In *International Conference on SMC*. IEEE, 2020. URL https://ieeexplore.ieee.org/iel7/9282733/9282811/09283495.pdf.

Zhuyun Dai and Jamie Callan. Context-aware term weighting for first stage passage retrieval. In *SIGIR*, 2020. URL https://dl.acm.org/doi/pdf/10.1145/3397271.3401204.

Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *WSDM*, 2018. URL https://dl.acm.org/doi/pdf/10.1145/3159652.3159659.

Fraser Dallas. Math information retrieval using a text search engine. Master's thesis, University of Waterloo, 2018. URL https://uwspace.uwaterloo.ca/handle/10012/13329.

Kenny Davila and Richard Zanibbi. Layout and semantics: Combining representations for mathematical formula search. In *SIGIR*, 2017. URL https://dl.acm.org/doi/abs/10.1145/3077136.3080748.

Kenny Davila, Richard Zanibbi, Andrew Kane, and Frank Wm. Tompa. Tangent-3 at the NTCIR-12 MathIR Task. In *NTCIR*, 2016. URL https://www.cs.rit.edu/~rlaz/files/ntcir2016_tangent.pdf.

Kenny Davila, Ritvik Joshi, Srirangaraj Setlur, Venu Govindaraju, and Richard Zanibbi. Tangent-V: Math formula image search using line-of-sight graphs. In *ECIR*, 2019. URL https://par.nsf.gov/servlets/purl/10124341.

Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. Neural ranking models with weak supervision. In *SIGIR*, 2017. URL https://dl.acm.org/doi/pdf/10.1145/3077136.3080832.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*, 2019. URL https://arxiv.org/abs/1810.04805.

Shuai Ding and Torsten Suel. Faster top-k document retrieval using block-max indexes. In *SIGIR*, 2011. URL https://dl.acm.org/doi/10.1145/2009916.2010048.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-E: An embodied multimodal language model. *arXiv:2303.03378*, 2023. URL https://arxiv.org/abs/2303.03378.

Susan T. Dumais. Latent semantic analysis. *Annual Review of Information Science and Technology (ARIST)*, 2004. URL https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/aris.1440380105.

Hui Fang and Chengxiang Zhai. An exploration of axiomatic approaches to information retrieval. In *SIGIR*, 2005. URL https://dl.acm.org/doi/10.1145/1076034.1076116.

Deborah Ferreira and André Freitas. STAR: Cross-modal statement representation for selecting relevant mathematical premises. In *EACL*, 2021. URL https://aclanthology.org/2021.eacl-main.282.pdf.

Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. SPLADE v2: Sparse lexical and expansion model for information retrieval. *arXiv:2109.10086*, 2021a. URL https://arxiv.org/abs/2109.10086.

Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. SPLADE: Sparse lexical and expansion model for first stage ranking. *arXiv:2107.05720*, 2021b. URL https://arxiv.org/abs/2107.05720.

Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. From distillation to hard negative sampling: Making sparse neural ir models more effective. *arXiv:2205.04733*, 2022. URL https://arxiv.org/abs/2205.04733.

Dallas Fraser, Andrew Kane, and Frank Wm. Tompa. Choosing math features for BM25 ranking with Tangent-L. In *DocEng*, 2018. URL https://dl.acm.org/doi/abs/10.1145/3209280.3209527.

George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 1987. URL https://dl.acm.org/doi/pdf/10.1145/32206.32212.

Neeraj Gangwar and Nickvash Kani. Semantic representations of mathematical expressions in a continuous vector space. *arXiv:2211.08142*, 2022. URL https://arxiv.org/abs/2211.08142.

Liangcai Gao, Ke Yuan, Yuehan Wang, Zhuoren Jiang, and Zhi Tang. The math retrieval system of ICST for NTCIR-12 MathIR Task. In *NTCIR*, 2016. URL https://research.nii.ac.jp/ntcir/workshop/OnlineProceedings12/pdf/ntcir/MathIR/03-NTCIR12-MathIR-GaoL.pdf.

Liangcai Gao, Zhuoren Jiang, Yue Yin, Ke Yuan, Zuoyu Yan, and Zhi Tang. Preliminary exploration of formula embedding for mathematical information retrieval: Can mathematical formulae be embedded like a natural language? *arXiv:1707.05154*, 2017. URL https://arxiv.org/abs/1707.05154.

Luyu Gao and Jamie Callan. Unsupervised corpus aware language model pre-training for dense passage retrieval. *arXiv:2108.05540*, 2021a. URL https://arxiv.org/abs/2108.05540.

Luyu Gao and Jamie Callan. Condenser: A pre-training architecture for dense retrieval. *arXiv:2104.08253*, 2021b. URL https://arxiv.org/abs/2104.08253.

Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. Complement lexical retrieval model with semantic residual embeddings. In *ECIR*, 2021a. URL https://arxiv.org/pdf/2004.13969.pdf.

Luyu Gao, Yunyi Zhang, Jiawei Han, and Jamie Callan. Scaling deep contrastive learning batch size under memory limited setup. *arXiv:2101.06983*, 2021b. URL https://arxiv.org/abs/2101.06983.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple contrastive learning of sentence embeddings. *arXiv:2104.08821*, 2021c. URL https://arxiv.org/abs/2104.08821.

Jonas Geiping and Tom Goldstein. Cramming: Training a language model on a single gpu in one day. *arXiv:2212.14034*, 2022. URL https://arxiv.org/abs/2212.14034.

Martin Geletka, Vojtěch Kalivoda, Michal Štefánik, Marek Toma, and Petr Sojka. Diverse semantics representation is king: MIRMU and MSM at ARQMath 2022. In *CLEF*, 2022. URL http://www.dei.unipd.it/~ferro/CLEF-WN-Drafts/CLEF2022/paper-02.pdf.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *arXiv:2012.14913*, 2020. URL https://arxiv.org/abs/2012.14913.

André Greiner-Petter, Abdou Youssef, Terry Ruas, Bruce R. Miller, Moritz Schubotz, Akiko Aizawa, and Bela Gipp. Math-word embedding in math search and semantic extraction. *Scientometrics*, 2020. URL https://link.springer.com/article/10.1007/s11192-020-03502-9.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *CIKM*, 2016. URL https://dl.acm.org/doi/pdf/10.1145/2983323.2983769.

Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W. Bruce Croft, and Xueqi Cheng. A deep look into neural ranking models for information retrieval. *Information Processing & Management*, 2020. URL https://www.sciencedirect.com/science/article/pii/S0306457319302390.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv:2004.10964*, 2020. URL https://arxiv.org/abs/2004.10964.

K Guu, K Lee, Z Tung, P Pasupat, and MW Chang. REALM: Retrieval-augmented language model pre-training. *arXiv:2002.08909*, 2020. URL https://arxiv.org/abs/2002.08909.

Hiroya Hagino and Hiroaki Saito. Partial-match retrieval with structure-reflected indices at the NTCIR-10 math task. In *NTCIR*, 2013. URL https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=2e0c5ec56bf204c0302dc81a0ab3c2d44c76e373.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. ToolkenGPT: Augmenting frozen language models with massive tools via tool embeddings. *arXiv:2305.11554*, 2023. URL https://arxiv.org/abs/2305.11554.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. URL https://openaccess.thecvf.com/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. DeBERTa: Decoding-enhanced bert with disentangled attention. *arXiv:2006.03654*, 2020. URL https://arxiv.org/abs/2006.03654.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv:1606.08415*, 2016. URL https://arxiv.org/abs/1606.08415.

Yoshinori Hijikata, Hideki Hashimoto, and Shogo Nishida. An investigation of index formats for the search of MathML objects. In *International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops*, 2007. URL https://ieeexplore.ieee.org/document/4427581.

Yoshinori Hijikata, Hideki Hashimoto, and Shogo Nishida. Search mathematical formulas by mathematical formulas. In *SHI (Symposium on Human Interface)*, 2009. URL https://link.springer.com/content/pdf/10.1007/978-3-642-02556-3_46.pdf.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012. URL https://arxiv.org/abs/1207.0580.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997. URL https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf.

Thomas Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, 1999. URL https://dl.acm.org/doi/pdf/10.1145/312624.312649.

Xuan Hu, Liangcai Gao, Xiaoyan Lin, Zhi Tang, Xiaofan Lin, and Josef B. Baker. WikiMirs: A mathematical information retrieval system for wikipedia. In *JCDL*, 2013. URL https://dl.acm.org/doi/abs/10.1145/2467696.2467699.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*, 2013. URL ihttps://dl.acm.org/doi/pdf/10.1145/2505515.2505665.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. *arXiv:1905.01969*, 2019. URL https://arxiv.org/abs/1905.01969.

Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. *arXiv:2007.01282*, 2020. URL https://arxiv.org/pdf/2007.01282.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Towards unsupervised dense information retrieval with contrastive learning. *arXiv:2112.09118*, 2021. URL https://arxiv.org/abs/2112.09118.

Shuiwang Ji, Yaochen Xie, and Hongyang Gao. A mathematical view of attention models in deep learning. *Texas A&M University (Lecture Notes)*, 2019. URL http://people.tamu.edu/~sji/classes/attn.pdf.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 2022. URL https://dl.acm.org/doi/pdf/10.1145/3571730.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. *arXiv:1909.10351*, 2019. URL https://arxiv.org/abs/1909.10351.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 2019. URL https://ieeexplore.ieee.org/document/8733051.

Simon Jonassen and Svein Erik Bratsberg. Efficient compressed inverted index skipping for disjunctive text-queries. In *ECIR*, 2011. URL https://link.springer.com/chapter/10.1007/978-3-642-20161-5_53.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972. URL https://www.emerald.com/insight/content/doi/10.1108/eb026526.

Shahab Kamali and Frank Wm. Tompa. Improving mathematics retrieval. *TDML*, 2009. URL https://dml.cz/bitstream/handle/10338.dmlcz/702556/DML_002-2009-1_6.pdf.

Shahab Kamali and Frank Wm. Tompa. A new mathematics retrieval system. In *ICKIM*, 2010. URL https://dl.acm.org/doi/pdf/10.1145/1871437.1871635.

Shahab Kamali and Frank Wm. Tompa. Structural similarity search for mathematics retrieval. In *MKM*. Springer, 2013. URL https://link.springer.com/chapter/10.1007/978-3-642-39320-4_16.

Chris Kamphuis, Arjen P. de Vries, Leonid Boytsov, and Jimmy Lin. Which BM25 do you mean? A large-scale reproducibility study of scoring variants. In *ECIR*, 2020. URL https://link.springer.com/chapter/10.1007/978-3-030-45442-5_4.

Andrew Kane, Yin Ki Ng, and Frank Wm. Tompa. Dowsing for answers to math questions. doing better with less. *CLEF*, 2022.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv:2001.08361*, 2020. URL https://arxiv.org/pdf/2001.08361.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-Tau Yih. Dense passage retrieval for open-domain question answering. *arXiv:2004.04906*, 2020. URL https://arxiv.org/abs/2004.04906.

Omar Khattab and Matei Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *SIGIR*, 2020. URL https://dl.acm.org/doi/abs/10.1145/3397271.3401075.

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. Demonstrate-Search-Predict: Composing retrieval and language models for knowledge-intensive NLP. *arXiv:2212.14024*, 2022. URL https://arxiv.org/abs/2212.14024.

Shinil Kim, Seon Yang, and Youngjoong Ko. Mathematical equation retrieval using plain words as a query. In *CIKM*, 2012. URL https://dl.acm.org/doi/pdf/10.1145/2396761.2398653.

Michael Kohlhase and Ioan Sucan. A search engine for mathematical formulae. In Jacques Calmet, Tetsuo Ida, and Dongming Wang, editors, *Artificial Intelligence and Symbolic Computation*, 2006. URL https://link.springer.com/chapter/10.1007/11856290_21.

Michael Kohlhase, Bogdan A Matican, and Corneliu-Claudiu Prodescu. MathWebSearch 0.5: Scaling an open formula search engine. In *CICM*, 2012. URL https://link.springer.com/chapter/10.1007/978-3-642-31374-5_23.

Giovanni Kristianto, Florence Ho, Goran Topic, and Akiko Aizawa. The MCAT math retrieval system for NTCIR-11 math track. In *NTCIR*, 2014. URL https://research.nii.ac.jp/ntcir/workshop/OnlineProceedings11/pdf/NTCIR/Math-2/06-NTCIR11-MATH-KristiantoGY.pdf.

Giovanni Yoko Kristianto, Goran Topic, and Akiko Aizawa. MCAT math retrieval system for NTCIR-12 MathIR task. In *NTCIR*, 2016. URL http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings12/pdf/ntcir/MathIR/04-NTCIR12-MathIR-KristiantoGY.pdf.

Kriste Krstovski and David M. Blei. Equation embeddings. *arXiv:1803.09123*, 2018. URL https://arxiv.org/abs/1803.09123.

George Labahn, Edward Lank, Mirette Marzouk, Andrea Bunt, Scott MacLean, and David Tausky. MathBrush: A case study for pen-based interactive mathematics. In *Proceedings*

of the Fifth Eurographics Conference on Sketch-Based Interfaces and Modeling, 2008. URL https://cs.uwaterloo.ca/~msmarzou/sbim2008.pdf.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. *arXiv:1909.11942*, 2019. URL https://arxiv.org/abs/1909.11942.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. *arXiv:1906.00300*, 2019. URL https://arxiv.org/abs/1906.00300.

Lillian Lee. IDF revisited: A simple new derivation within the robertson-spärck jones probabilistic model. In *SIGIR*, 2007. URL https://dl.acm.org/doi/pdf/10.1145/1277741.1277891.

Whay C. Lee and Edward A. Fox. Experimental comparison of schemes for interpreting boolean queries. 1988. URL https://eprints.cs.vt.edu/archive/00000112/01/TR-88-27.pdf.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *arXiv:2206.14858*, 2022. URL https://arxiv.org/abs/2206.14858.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *NeurIPS*, 2018. URL https://proceedings.neurips.cc/paper/2018/file/a41b3bb3e6b050b6c9067c67f663b915-Paper.pdf.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv:2301.12597*, 2023. URL https://arxiv.org/abs/rXiv:2301.12597.

Jimmy Lin. A proposed conceptual framework for a representational approach to information retrieval, 2022. URL https://dl.acm.org/doi/pdf/10.1145/3527546.3527552.

Jimmy Lin and Xueguang Ma. A few brief notes on deepimpact, COIL, and a conceptual framework for information retrieval techniques. *arXiv:2106.14807*, 2021. URL https://arxiv.org/abs/2106.14807.

Xiaoyan Lin, Liangcai Gao, Xuan Hu, Zhi Tang, Yingnan Xiao, and Xiaozhong Liu. A mathematics retrieval system for formulae in layout presentations. In *SIGIR*, 2014. URL https://dl.acm.org/doi/pdf/10.1145/2600428.2609611.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv:1703.03130*, 2017. URL https://arxiv.org/abs/1703.03130.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized bert pretraining approach. *arXiv:1907.11692*, 2019. URL https://arxiv.org/abs/1907.11692.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv:1711.05101*, 2017. URL https://arxiv.org/abs/1711.05101.

Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. A survey of deep learning for mathematical reasoning. *arXiv:2212.10535*, 2022. URL https://arxiv.org/pdf/2212.10535.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. Chameleon: Plug-and-play compositional reasoning with large language models. *arXiv:2304.09842*, 2023. URL https://arxiv.org/abs/2304.09842.

Shuqi Lu, Di He, Chenyan Xiong, Guolin Ke, Waleed Malik, Zhicheng Dou, Paul Bennett, Tieyan Liu, and Arnold Overwijk. Less is more: Pre-train a strong text encoder for dense retrieval using a weak decoder. *arXiv:2102.09206*, 2021. URL https://arxiv.org/abs/2102.09206.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. Sparse, dense, and attentional representations for text retrieval. *TACL*, 2021. URL https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00369/100684.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *arXiv:1508.04025*, 2015. URL https://arxiv.org/abs/1508.04025.

Yuanhua Lv and Chengxiang Zhai. Lower-bounding term frequency normalization. In *CIKM*, 2011. URL https://dl.acm.org/doi/abs/10.1145/2063576.2063584.

Kai Ma, Siu Cheung Hui, and Kuiyu Chang. Feature extraction and clustering-based retrieval for mathematical formulas. In *International Conference on Software Engineering and Data Mining*, 2010. URL https://ieeexplore.ieee.org/iel5/5510904/5542824/05542894.pdf.

Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Xiang Ji, and Xueqi Cheng. PROP: Pre-training with representative words prediction for ad-hoc retrieval. In *WSDM*, 2021a. URL https://dl.acm.org/doi/pdf/10.1145/3437963.3441777.

Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Yingyan Li, and Xueqi Cheng. B-PROP: Bootstrapped pre-training with representative words prediction for ad-hoc retrieval. In *SIGIR*, 2021b. URL https://dl.acm.org/doi/pdf/10.1145/3404835.3462869.

Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, and Xueqi Cheng. Pre-train a discriminative text encoder for dense retrieval via contrastive span prediction. *arXiv:2204.10641*, 2022a. URL https://arxiv.org/abs/2204.10641.

Yi Ma, Doris Tsao, and Heung-Yeung Shum. On the principles of parsimony and self-consistency for the emergence of intelligence. *arXiv:2207.04630*, 2022b. URL https://arxiv.org/abs/2207.04630.

Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. CEDR: Contextualized embeddings for document ranking. In *SIGIR*, 2019. URL https://dl.acm.org/doi/pdf/10.1145/3331184.3331317.

Joel Mackenzie, Andrew Trotman, and Jimmy Lin. Wacky weights in learned sparse representations and the revenge of score-at-a-time query evaluation. *arXiv:2110.11540*, 2021. URL https://arxiv.org/abs/2110.11540.

Yu A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 2018. URL https://ieeexplore.ieee.org/iel7/34/4359286/08594636.pdf.

Antonio Mallia, Michał Siedlaczek, and Torsten Suel. An experimental study of index compression and DAAT query processing methods. In *ECIR*. Springer, 2019. URL https://par.nsf.gov/servlets/purl/10171629.

Antonio Mallia, Omar Khattab, Torsten Suel, and Nicola Tonellotto. Learning passage impacts for inverted indexes. In *SIGIR*, 2021. URL https://dl.acm.org/doi/pdf/10.1145/3404835.3463030.

Antonio Mallia, Joel Mackenzie, Torsten Suel, and Nicola Tonellotto. Faster learned sparse retrieval with guided traversal. *arXiv:2204.11314*, 2022. URL https://arxiv.org/abs/2204.11314.

Behrooz Mansouri, Shaurya Rohatgi, Douglas W. Oard, Jian Wu, C Lee Giles, and Richard Zanibbi. Tangent-CFT: An embedding model for mathematical formulas. In *SIGIR*, 2019a. URL https://dl.acm.org/doi/abs/10.1145/3341981.3344235.

Behrooz Mansouri, Richard Zanibbi, and Douglas W. Oard. Characterizing searches for mathematical concepts. In *JCDL*. IEEE, 2019b. URL https://ieeexplore.ieee.org/iel7/8776994/8791105/08791164.pdf.

Behrooz Mansouri, Anurag Agarwal, Douglas W. Oard, and Richard Zanibbi. Finding old answers to new math questions: The ARQMath Lab at CLEF 2020. In *Advances in Information Retrieval*, 2020a. URL https://ceur-ws.org/Vol-2696/paper_271.pdf.

Behrooz Mansouri, Douglas W. Oard, and Richard Zanibbi. DPRL systems in the CLEF 2020 ARQMath lab. In *CLEF*, 2020b. URL https://par.nsf.gov/servlets/purl/10198749.

Behrooz Mansouri, Douglas W. Oard, Anurag Agarwal, and Richard Zanibbi. Effects of context, complexity, and clustering on evaluation for math formula retrieval. *arXiv:2111.10504*, 2021a. URL https://arxiv.org/abs/2111.10504.

Behrooz Mansouri, Douglas W. Oard, and Richard Zanibbi. DPRL systems in the CLEF 2021 ARQMath Lab: Sentence-BERT for answer retrieval, learning-to-rank for formula retrieval. In *CLEF*, 2021b. URL http://ceur-ws.org/Vol-2936/paper-04.pdf.

Behrooz Mansouri, Richard Zanibbi, Douglas W. Oard, and Anurag Agarwal. Overview of ARQMath-2 (2021): Second CLEF lab on Answer Retrieval for Questions on Math (Working Notes Version). In *CLEF*, 2021c. URL http://ceur-ws.org/Vol-2936/paper-01.pdf.

Behrooz Mansouri, Vít Novotný, Anurag Agarwal, Douglas W. Oard, and Richard Zanibbi. Overview of ARQMath-3 (2022): Third CLEF lab on Answer Retrieval for Questions on Math (Working Notes Version). In *CLEF*, 2022. URL https://ceur-ws.org/Vol-3180/paper-01.pdf.

Jordan Meadows and Andre Freitas. A survey in mathematical language processing. *arXiv:2205.15231*, 2022. URL https://arxiv.org/abs/2205.15231.

Erica Melis, Giorgi Goguadze, Martin Homik, Paul Libbrecht, Carsten Ullrich, and Stefan Winterstein. Semantic-aware components and services of ActiveMath. *British Journal of Educational Technology*, 2006. URL

https://bera-journals.onlinelibrary.wiley.com/doi/pdfdirect/10.1111/j.1467-8535.2006.00613.x.

Bruce R. Miller and Abdou Youssef. Technical aspects of the digital library of mathematical functions. In *AMAI*, 2003. URL https://link.springer.com/article/10.1023/A:1022967814992.

Bruce R. Miller and Abdou Youssef. Augmenting presentation mathml for search. *Lecture Notes in Computer Science*, 2008. URL https://www2.seas.gwu.edu/~ayoussef/papers/Augmenting%20Presentation%20MathML%20for%20Search.pdf.

Robert Miner and Rajesh Munavalli. An approach to mathematical search through query formulation and data normalization. In *MKM*. Springer, 2007. URL https://link.springer.com/chapter/10.1007/978-3-540-73086-6_27.

Jozef Mišutka and Leo Galamboš. Extending full text search engine for mathematical content. *Towards Digital Mathematics Library*, 2008. URL https://dml.cz/bitstream/handle/10338.dmlcz/702546/DML_001-2008-1_7.pdf.

Rajesh Munavalli and Robert Miner. Mathfind: A math-aware search engine. In *SIGIR*, 2006. URL https://dl.acm.org/doi/pdf/10.1145/1148170.1148348.

Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. Improving document ranking with dual word embeddings. In *WWW*, 2016. URL https://dl.acm.org/doi/pdf/10.1145/2872518.2889361.

Yin Ki Ng. Dowsing for math answers: exploring MathCQA with a math-aware search engine. Master's thesis, University of Waterloo, 2021. URL https://uwspace.uwaterloo.ca/handle/10012/17696.

Yin Ki Ng, Dallas Fraser, Besat Kassaie, and Frank Wm. Tompa. Dowsing for answers to math questions: Ongoing viability of traditional MathIR. In *CLEF*, 2021. URL http://ceur-ws.org/Vol-2936/paper-05.pdf.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A human generated machine reading comprehension dataset. In *CoCoNIPs*, 2016. URL http://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y Zhao, Yi Luan, Keith B Hall, Ming-Wei Chang, et al. Large dual encoders are generalizable retrievers. *arXiv:2112.07899*, 2021. URL https://arxiv.org/abs/2112.07899.

Gavin Nishizawa, Jennifer Liu, Yancarlos Diaz, Abishai Dmello, Wei Zhong, and Richard Zanibbi. MathSeer: A math-aware search interface with intuitive formula editing, reuse, and lookup. In *ECIR*, 2020. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7148076.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. *arXiv:1901.04085*, 2019. URL https://arxiv.org/abs/1901.04085.

Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. From doc2query to docTTTTTquery. 2019a. URL https://cs.uwaterloo.ca/~jimmylin/publications/Nogueira_Lin_2019_docTTTTTquery-v2.pdf.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with BERT. *arXiv:1910.14424*, 2019b. URL https://arxiv.org/abs/1910.14424.

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. Document expansion by query prediction. *arXiv:1904.08375*, 2019c. URL https://arxiv.org/abs/1904.08375.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. Investigating the limitations of Transformers with simple arithmetic tasks. *arXiv:2102.13019*, 2021. URL https://arxiv.org/abs/2102.13019.

Landon Curt Noll. FNV Hash. 2023. URL http://www.isthe.com/chongo/tech/comp/fnv/index.html. [Online; accessed 2023-06-16].

Immanuel Normann and Michael Kohlhase. Extended formula normalization for ε-retrieval and sharing of mathematical knowledge. In *MKM*. Springer, 2007. URL https://link.springer.com/chapter/10.1007/978-3-540-73086-6_28.

Vít Novotnỳ and Michal Štefánik. Combining sparse and dense information retrieval. *CLEF*, 2022. URL http://ceur-ws.org/Vol-3180/paper-06.pdf.

Vít Novotnỳ, Petr Sojka, Michal Stefánik, and Dávid Lupták. Three is better than one: Ensembling math information retrieval systems. In *CLEF*, 2020. URL http://ceur-ws.org/Vol-2696/paper_235.pdf.

Vít Novotnỳ, Michal Štefánik, Dávid Lupták, Martin Geletka, Petr Zelina, and Petr Sojka. Ensembling ten math information retrieval systems. In *CLEF*, 2021. URL http://ceur-ws.org/Vol-2936/paper-06.pdf.

Vít Novotnỳ. *Interpretable Representations for Fast and Accurate Retrieval of Mathematical Information*. PhD thesis, Dissertation, Masaryk University, 2021. URL https://is.muni.cz/th/o4thd/Revidovana_verze_po_obhajobe_disertace.pdf.

Shunsuke Ohashi, Giovanni Yoko Kristianto, Akiko Aizawa, et al. Efficient algorithm for math formula semantic search. *IEICE Transactions*, 2016. URL https://www.jstage.jst.go.jp/article/transinf/E99.D/4/E99.D_2015DAP0023/_pdf.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive APIs. *arXiv:2305.15334*, 2023. URL https://arxiv.org/abs/2305.15334.

Nidhin Pattaniyil and Richard Zanibbi. Combining TF–IDF text retrieval with an inverted index over symbol pairs in math expressions: The tangent math search engine at ntcir 2014. In *NTCIR*, 2014. URL https://research.nii.ac.jp/ntcir/workshop/OnlineProceedings11/pdf/NTCIR/Math-2/08-NTCIR11-MATH-PattaniyilN.pdf.

Shuai Peng, Ke Yuan, Liangcai Gao, and Zhi Tang. MathBERT: A pre-trained model for mathematical formula understanding. *arXiv:2105.00377*, 2021. URL https://arxiv.org/abs/2105.00377.

Lukas Pfahler and Katharina Morik. Self-supervised pretraining of graph neural network for the retrieval of related mathematical expressions in scientific articles. *arXiv:2209.00446*, 2022. URL https://arxiv.org/abs/2209.00446.

Xin Qian and Diego Klabjan. The impact of the mini-batch size on the variance of gradients in stochastic gradient descent. *arXiv:2004.13146*, 2020. URL https://arxiv.org/abs/2004.13146.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. *arXiv:2010.08191*, 2020. URL https://arxiv.org/abs/2010.08191.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training Gopher. *arXiv:2112.11446*, 2021. URL https://arxiv.org/abs/2112.11446.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text Transformer. *JMLR*, 2020. URL https://www.jmlr.org/papers/volume21/20-074/20-074.pdf.

Ori Ram, Gal Shachaf, Omer Levy, Jonathan Berant, and Amir Globerson. Learning to retrieve passages without supervision. *arXiv:2112.07708*, 2021. URL https://arxiv.org/abs/2112.07708.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. *arXiv:1908.10084*, 2022. URL https://arxiv.org/abs/1908.10084.

Anja Reusch, Maik Thiele, and Wolfgang Lehner. An ALBERT-based similarity measure for mathematical answer retrieval. In *SIGIR*, 2021a. URL https://dl.acm.org/doi/abs/10.1145/3404835.3463023.

Anja Reusch, Maik Thiele, and Wolfgang Lehner. TU_DBS in the ARQMath lab 2021, CLEF. In *CLEF*, 2021b. URL http://ceur-ws.org/Vol-2936/paper-07.pdf.

Anja Reusch, Maik Thiele, and Wolfgang Lehner. Transformer-encoder and decoder models for questions on math. *CLEF*, 2022. URL http://ceur-ws.org/Vol-3180/paper-07.pdf.

Stephen E. Robertson. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation*, 2004. URL https://www.emerald.com/insight/content/doi/10.1108/00220410410560582/full/html.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline M. Hancock-Beaulieu, Mike Gatford, et al. Okapi at TREC-3. *TREC*, 1995. URL https://www.microsoft.com/en-us/research/publication/okapi-at-trec-3.

Shaurya Rohatgi, Jian Wu, and C Lee Giles. PSU at CLEF-2020 ARQMath Track: Unsupervised re-ranking using pretraining. In *CLEF*, 2020. URL http://ceur-ws.org/Vol-2696/paper_121.pdf.

Shaurya Rohatgi, Jian Wu, and C Lee Giles. Ranked list fusion and re-ranking with pre-trained transformers for arqmath lab. 2021. URL http://ceur-ws.org/Vol-2936/paper-08.pdf.

Tetsuya Sakai and Noriko Kando. On information retrieval metrics designed for evaluation with incomplete relevance assessments. *Information Retrieval*, 2008. URL https://link.springer.com/article/10.1007/s10791-008-9059-7.

Sidath Harshanath Samarasinghe and Siu Cheung Hui. Mathematical document retrieval for problem solving. In *International Conference on Computer Engineering and Technology*. IEEE, 2009. URL https://ieeexplore.ieee.org/iel5/4769406/4769407/04769534.pdf.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. *arXiv:2112.01488*, 2021. URL https://arxiv.org/abs/2112.01488.

Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. PLAID: An efficient engine for late interaction retrieval. *arXiv:2205.09707*, 2022a. URL https://arxiv.org/abs/2205.09707.

Keshav Santhanam, Jon Saad-Falcon, Martin Franz, Omar Khattab, Avirup Sil, Radu Florian, Md Arafat Sultan, Salim Roukos, Matei Zaharia, and Christopher Potts. Moving beyond downstream task accuracy for information retrieval benchmarking. *arXiv:2212.01340*, 2022b. URL https://arxiv.org/abs/2212.01340.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv:2302.04761*, 2023. URL https://arxiv.org/pdf/2302.04761.

Till Hendrik Schulz, Tamás Horváth, Pascal Welke, and Stefan Wrobel. A generalized Weisfeiler-Lehman graph kernel. *Machine Learning*, 2022. URL https://link.springer.com/article/10.1007/s10994-022-06131-w.

Hinrich Schütze, Christopher D. Manning, and Prabhakar Raghavan. *Introduction to information retrieval*. Cambridge University Press Cambridge, 2008. URL https://nlp.stanford.edu/IR-book/information-retrieval-book.html.

Ron Shamir and Dekel Tsur. Faster subtree isomorphism. *Journal of Algorithms*, 1999. URL https://www.sciencedirect.com/science/article/pii/S0196677499910441.

Dongdong Shan, Shuai Ding, Jing He, Hongfei Yan, and Xiaoming Li. Optimized top-k processing with global page scores on block-max indexes. In *WSDM*, 2012. URL https://dl.acm.org/doi/10.1145/2124295.2124346.

Mohammed Shatnawi and Abdou Youssef. Equivalence detection using parse-tree normalization for math search. In *ICDIM*. IEEE, 2007. URL https://ieeexplore.ieee.org/iel5/4444189/4444274/04444297.pdf.

Tao Shen, Xiubo Geng, Chongyang Tao, Can Xu, Kai Zhang, and Daxin Jiang. Unifier: A unified retriever for large-scale retrieval. *arXiv:2205.11194*, 2022. URL https://arxiv.org/abs/2205.11194.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving AI tasks with ChatGPT and its friends in HuggingFace. *arXiv:2303.17580*, 2023. URL https://arxiv.org/abs/2303.17580.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. Replug: Retrieval-augmented black-box language models. *arXiv:2301.12652*, 2023. URL https://arxiv.org/abs/2301.12652.

Petr Sojka and Martin Líška. The art of mathematics retrieval. In *DocEng*, 2011a. URL https://dl.acm.org/doi/10.1145/2034691.2034703.

Petr Sojka and Martin Líška. Indexing and searching mathematics in digital libraries: architecture, design and scalability issues. In *MKM*. Springer, 2011b. URL https://is.muni.cz/th/udtlg/teze_Archive.pdf.

Yujin Song and Xiaoyu Chen. Searching for mathematical formulas based on graph representation learning. In *CICM*, 2021. URL https://link.springer.com/chapter/10.1007/978-3-030-81097-9_11.

David Stalnaker. Math expression retrieval using symbol pairs in layout trees. 2013. URL https://www.cs.rit.edu/~dprl/files/StalnakerMScThesisAug2013.pdf.

David Stalnaker and Richard Zanibbi. Math expression retrieval using an inverted index over symbol pairs. In *Document recognition and retrieval*, 2015. URL https://www.cs.rit.edu/~rlaz/files/drr-stalnaker2015-revised.pdf.

Trevor Strohman, Howard Turtle, and W. Bruce Croft. Optimization strategies for complex queries. In *SIGIR*, 2005. URL https://dl.acm.org/doi/10.1145/1076034.1076074.

Tao Tao and Chengxiang Zhai. An exploration of proximity measures in information retrieval. In *SIGIR*, 2007. URL https://dl.acm.org/doi/10.1145/1277741.1277794.

Terrier contributors. Terrier documenetation, 2014. URL http://terrier.org/docs/v4.0/javadoc/org/terrier/matching/models/TF_IDF.html. [Online; accessed 2022-12-17].

Nandan Thakur, Nils Reimers, and Jimmy Lin. Domain adaptation for memory-efficient dense retrieval. *arXiv:2205.11498*, 2022. URL https://arxiv.org/abs/2205.11498.

Nicola Tonellotto, Craig Macdonald, and Iadh Ounis. Efficient and effective retrieval using selective pruning. In *WSDM*, 2013. URL https://dl.acm.org/doi/pdf/10.1145/2433396.2433407.

Goran Topić, Giovanni Yoko Kristianto, Minh-Quoc Nghiem, and Akiko Aizawa. The MCAT math retrieval system for NTCIR-10 math track. In *NTCIR*, 2013. URL http://ntcir-math.nii.ac.jp/wp-content/blogs.dir/23/files/2013/10/05-NTCIR10-MATH-TopicG.pdf.

Andrew Trotman and Matt Crane. Micro-and macro-optimizations of SAAT search. *Software: Practice and Experience*, 2019. URL https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.2683.

Howard Turtle and James Flood. Query evaluation: strategies and optimizations. *Information Processing & Management*, 1995. URL https://www.sciencedirect.com/science/article/pii/030645739500020H.

Gabriel Valiente. *Algorithms on trees and graphs*. Springer, 2002. URL https://link.springer.com/book/10.1007/978-3-030-81885-2.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Ellen M. Voorhees. The evolution of cranfield. *Information Retrieval Evaluation in a Changing World: Lessons Learned from 20 Years of CLEF*, 2019. URL https://link.springer.com/chapter/10.1007/978-3-030-22948-1_2.

Kexin Wang, Nils Reimers, and Iryna Gurevych. TSDAE: Using Transformer-based sequential denoising auto-encoder for unsupervised sentence embedding learning. *arXiv:2104.06979*, 2021a. URL https://arxiv.org/abs/2104.06979.

Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. GPL: Generative pseudo labeling for unsupervised domain adaptation of dense retrieval. *arXiv:2112.07577*, 2021b. URL https://arxiv.org/abs/2112.07577.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. SimLM: Pre-training with representation bottleneck for dense passage retrieval. *arXiv:2207.02578*, 2022. URL https://arxiv.org/abs/2207.02578.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. URL http://openaccess.thecvf.com/content_cvpr_2018/papers/Wang_Non-Local_Neural_Networks_CVPR_2018_paper.pdf.

Yuehan Wang, Liangcai Gao, Simeng Wang, Zhi Tang, Xiaozhong Liu, and Ke Yuan. WikiMirs 3.0: A hybrid MIR system based on the context, structure and importance of formulae in a document. In *JCDL*, 2015. URL https://dl.acm.org/doi/pdf/10.1145/2756406.2756918.

Zichao Wang, Andrew S Lan, and Richard G. Baraniuk. Mathematical formula representation via tree embeddings. In *iTextbooks AIED*, 2021c. URL http://ceur-ws.org/Vol-2895/paper02.pdf.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *EMNLP*, 2020. URL https://aclanthology.org/2020.emnlp-demos.6.

Xing Wu, Guangyuan Ma, Meng Lin, Zijia Lin, Zhongyuan Wang, and Songlin Hu. Contextual mask auto-encoder for dense passage retrieval. *arXiv:2208.07670*, 2022. URL https://arxiv.org/abs/2208.07670.

Shitao Xiao and Zheng Liu. RetroMAE v2: Duplex masked auto-encoder for pre-training retrieval-oriented language models. *arXiv:2211.08769*, 2022. URL https://arxiv.org/abs/2211.08769.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. End-to-end neural ad-hoc ranking with kernel pooling. In *SIGIR*, 2017. URL https://dl.acm.org/doi/pdf/10.1145/3077136.3080809.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv:2007.00808*, 2020a. URL https://arxiv.org/abs/2007.00808.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the Transformer architecture. In *International Conference on Machine Learning*. PMLR, 2020b. URL http://proceedings.mlr.press/v119/xiong20b/xiong20b.pdf.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv:1810.00826*, 2018. URL https://arxiv.org/abs/1810.00826.

Peilin Yang, Hui Fang, and Jimmy Lin. Anserini: Enabling the use of lucene for information retrieval research. In *SIGIR*, 2017. URL https://dl.acm.org/doi/pdf/10.1145/3077136.3080721.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv:2210.03629*, 2022. URL https://arxiv.org/abs/2210.03629.

Michihiro Yasunaga and John D Lafferty. TopicEQ: A joint topic and mathematical equation model for scientific texts. In *AAAI*, 2019. URL https://ojs.aaai.org/index.php/AAAI/article/download/4728/4606.

Keisuke Yokoi and Akiko Aizawa. An approach to similarity search for mathematical expressions using MathML. In *DML (Digital Mathematics Library)*, 2009. URL https://dml.cz/handle/10338.dmlcz/702557.

Abdou Youssef. Search of mathematical contents: Issues and methods. In *IASSE*, 2005. URL https://www2.seas.gwu.edu/~ayoussef/papers/MathSearch-IASSE05.pdf.

Abdou Youssef. Methods of relevance ranking and hit-content generation in math search. In *MKM*. Springer, 2007. URL https://link.springer.com/chapter/10.1007/978-3-540-73086-6_31.

Yaodong Yu, Sam Buchanan, Druv Pai, Tianzhe Chu, Ziyang Wu, Shengbang Tong, Benjamin D Haeffele, and Yi Ma. White-box transformers via sparse rate reduction. *arXiv:2306.01129*, 2023. URL https://arxiv.org/abs/2306.01129.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J Reddi, and Sanjiv Kumar. Are Transformers universal approximators of sequence-to-sequence functions? *arXiv:1912.10077*, 2019. URL https://arxiv.org/abs/1912.10077.

Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik Learned-Miller, and Jaap Kamps. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *CIKM*, 2018. URL https://dl.acm.org/doi/pdf/10.1145/3269206.3271800.

Richard Zanibbi and Dorothea Blostein. Recognition and retrieval of mathematical expressions. In *IJDAR*, 2012. URL https://link.springer.com/article/10.1007/s10032-011-0174-4.

Richard Zanibbi and Li Yu. Math Spotting: Retrieving math in technical documents using handwritten query images. In *ICDAR*, 2011. URL https://ieeexplore.ieee.org/iel5/6065245/6065247/06065351.pdf.

Richard Zanibbi, Kenny Davila, Andrew Kane, and Frank Wm. Tompa. The Tangent search engine: Improved similarity metrics and scalability for math formula search. *arXiv:1507.06235*, 2015. URL https://arxiv.org/abs/1507.06235.

Richard Zanibbi, Akiko Aizawa, Michael Kohlhase, Iadh Ounis, Goran Topic, and Kenny Davila. NTCIR-12 MathIR task overview. In *NTCIR*, 2016a. URL https://research.nii.ac.jp/ntcir/workshop/OnlineProceedings12/pdf/ntcir/OVERVIEW/01-NTCIR12-OV-MathIR-ZanibbiR.pdf.

Richard Zanibbi, Kenny Davila, Andrew Kane, and Frank Wm. Tompa. Multi-stage math formula search: Using appearance-based similarity metrics at scale. In *SIGIR*, 2016b. URL https://dl.acm.org/doi/abs/10.1145/2911451.2911512.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. Learning to retrieve: How to train a dense retrieval model effectively and efficiently. *arXiv:2010.10469*, 2020a. URL https://arxiv.org/abs/2010.10469.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. RepBERT: Contextualized text embeddings for first-stage retrieval. *arXiv:2006.15498*, 2020b. URL https://arxiv.org/abs/2006.15498.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. Optimizing dense retrieval model training with hard negatives. In *SIGIR*, 2021. URL https://dl.acm.org/doi/pdf/10.1145/3404835.3462880.

Kaizhong Zhang and Tao Jiang. Some MAX SNP-hard results concerning unordered labeled trees. *Information Processing Letters*, 1994. URL https://www.sciencedirect.com/science/article/abs/pii/0020019094900620.

Mengxue Zhang, Zichao Wang, Richard Baraniuk, and Andrew Lan. Math operation embeddings for open-ended solution analysis and feedback. *arXiv:2104.12047*, 2021. URL https://arxiv.org/abs/2104.12047.

Jin Zhao, Min-Yen Kan, and Yin Leng Theng. Math information retrieval: user requirements and prototype implementation. In *JCDL*, 2008. URL https://dl.acm.org/doi/pdf/10.1145/1378889.1378921.

Tiancheng Zhao, Xiaopeng Lu, and Kyusong Lee. SPARTA: Efficient open-domain question answering via sparse Transformer matching retrieval. *arXiv:2009.13013*, 2020. URL https://arxiv.org/abs/2009.13013.

Wei Zhong. A novel similarity-search method for mathematical content in latex markup and its implementation. Master's thesis, University of Delaware, 2015. URL https://udspace.udel.edu/handle/19716/17656.

Wei Zhong and Hui Fang. OPMES: A similarity search engine for mathematical content. In *ECIR*, 2016. URL https://link.springer.com/chapter/10.1007/978-3-319-30671-1_79.

Wei Zhong and Jimmy Lin. PyA0: A Python toolkit for accessible math-aware search. In *SIGIR*, 2021. URL https://dl.acm.org/doi/abs/10.1145/3404835.3462794.

Wei Zhong and Richard Zanibbi. Structural similarity search for formulas using leaf-root paths in operator subtrees. In *ECIR*, 2019. URL https://par.nsf.gov/servlets/purl/10124342.

Wei Zhong, Shaurya Rohatgi, Jian Wu, Lee Giles, and Richard Zanibbi. Accelerating substructure similarity search for formula retrieval. In *ECIR*, 2020. URL https://link.springer.com/chapter/10.1007/978-3-030-45439-5_47.

Wei Zhong, Xinyu Zhang, Ji Xin, Jimmy Lin, and Richard Zanibbi. Approach Zero and Anserini at the CLEF-2021 ARQMath track: Applying substructure search and BM25 on operator tree path tokens. In *CLEF*, 2021. URL http://ceur-ws.org/Vol-2936/paper-09.pdf.

Wei Zhong, Yuqing Xie, and Jimmy Lin. Applying structural and dense semantic matching for the ARQMath Lab 2022, CLEF. *CLEF 2022*, 2022a. URL http://ceur-ws.org/Vol-3180/paper-09.pdf.

Wei Zhong, Jheng-Hong Yang, and Jimmy Lin. Evaluating token-level and passage-level dense retrieval models for math information retrieval, 2022b. URL https://arxiv.org/abs/2203.11163.

Wei Zhong, Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. One blade for one purpose: Advancing math information retrieval using hybrid search. In *SIGIR*, 2023.

Qi Zhu, Yuxian Gu, Lingxiao Luo, Bing Li, Cheng Li, Wei Peng, Minlie Huang, and Xiaoyan Zhu. When does further pre-training MLM help? An empirical study on

task-oriented dialog pre-training. In *EMNLP Insights Workshop*, 2021. URL https://aclanthology.org/2021.insights-1.9.

Shengyao Zhuang and Guido Zuccon. TILDE: Term independent likelihood model for passage re-ranking. In *SIGIR*, 2021a. URL https://dl.acm.org/doi/pdf/10.1145/3404835.3462922.

Shengyao Zhuang and Guido Zuccon. Fast passage re-ranking with contextualized exact term matching and efficient passage expansion. *arXiv:2108.08513*, 2021b. URL https://arxiv.org/abs/arXiv:2108.08513.

Justin Zobel and Alistair Moffat. Exploring the similarity space. In *SIGIR*, 1998. URL https://dl.acm.org/doi/pdf/10.1145/281250.281256.

Marcin Zukowski, Sandor Heman, Niels Nes, and Peter Boncz. Super-scalar RAM-CPU cache compression. In *ICDE*. IEEE, 2006. URL https://ieeexplore.ieee.org/iel5/10757/33902/01617427.pdf.

Glossary

first-stage retrieval An often more efficient stage that searches against the whole index to filter candidates used for potentially a more expensive later stage(s). 2

math-aware Being aware of the heterogeneous data often seen in a math document that contains both math language (i.e., formulas) and their context language. 3

MIR Math information retrieval (Math IR). See Section 2.3 4

OPT Operator Tree. The OPT represents a math formula by identifying operators and operands in the expression, and constructing a tree recursively where each internal node is the operator of its children operands. 32

structure search The retrieval of formulas based on their structure similarities. 22

structure similarity In our context, the similarity measured by largest common substructure(s) between formula OPT representations. 35