Barcelona School of Economics

Barcelona School of Economics

Master's Degree in Data Science:
Data Science Methodology Program

Master's Thesis

Fine-Tuning an embedding model to improve retrieval of mathematical documents

Authors:
Pol Garcia
Natalia Lavrova
Alex Malo

Supervisor:
Antonio Lozano

June 4, 2025

Abstract

Just as humans provide better answers when they have access to clear and relevant information, Retrieval-Augmented Generation systems rely on high-quality retrieval to generate accurate responses.

In this paper we apply Fine-Tuning to modernBERT, combined with other techniques such as Domain-Adaptive Pretraining to enhance the retrieval component of a RAG system, specifically within the domain of mathematical texts written in LaTeX.

We use mathematical papers published in January, February, and March 2025, scraped from arXiv, and extract statements such as Theorems, Lemmas and Definitions. The Fine-Tuning dataset consists on pairs of questions and a mathematical statement that could be used to answer the question, and we use a large language model (LLaMA 3.2) to synthetically generate questions. This setup enables the model to learn meaningful sentence embeddings tailored to the mathematical domain, ultimately improving the retrieval component of the RAG pipeline. We also experiment with the use of DAPT — to familiarize the model with mathematical specialized structure and terminology — and a Sentence Embedding optimized version of modernBERT.

We find that retrieval performance improves by up to **600%** across key retrieval metrics — including Recall, Precision, Accuracy, nDCG, and MRR. In a corpus of nearly 90,000 mathematical statements, the system boosts the probability of retrieving a relevant paper in the **top-5 results from under 25% to over 95%**, demonstrating near-deployment-level performance for domain-specific question answering.

Keywords: ModernBERT, RAG, DAPT, Mathematical Papers, Fine-Tuning, Embedding Model.

Contents

1 Introduction **4**

2 Literature Review **6**

3 Data and Preprocessing **9**
(a) arXiv Data 9
(b) Preprocessing 9
(c) Query Creation 10
(d) Final dataset structure 13

4 Methodology **15**
(a) Model architecture 15
(i) Tokenization 17
(ii) Embedding 19
(iii) Transformer encoder: self-attention layers 20
(iv) Pooling layer: sentence-level representation 22
(v) Sentence embedding construction 23
(b) DAPT 24
(c) Fine-tuning objective: Multiple Negatives Ranking Loss 26
(i) Mathematical Formulation 27
(d) Evaluation Metrics 28

5 Results and Discussion **31**
(a) ModernBERT base 31
(b) ModernBERT with DAPT 33
(c) Nomic AI ModernBERT 34
(d) Discussion 35

6 Conclusions and Recommendations **39**
(a) Conclusions 39
(b) Further Study 39

7 Appendix **43**

Symbols and Abbreviations

- **Statements** - In this paper, mathematical statement refer to the following structures: `theorem`, `lemma`, `definition`, `proposition` and `corollary`, that are extracted from the mathematical papers.
- **LLM** – Large Language Model: A transformer-based model trained on large-scale corpora to generate and understand natural language text.
- **FT** - Fine Tuning: process of taking a pre-trained model and further training it on a smaller, specialized dataset to make it better for a specific task.
- **RAG** – Retrieval-Augmented Generation: A hybrid architecture that combines information retrieval with a generative language model to improve response relevance.
- **DAPT** – Domain-Adaptive Pre-Training: A technique where a pre-trained language model is further trained on domain-specific unlabeled data to improve performance in that domain.
- **MRR** – Mean Reciprocal Rank: An evaluation metric that captures how early the first relevant result appears in a list of retrieved items.
- **nDCG@k** – Normalized Discounted Cumulative Gain at rank k: A ranking metric that considers the graded relevance of k items and their positions in the retrieved list.
- **Precision@k** – Precision at rank k: The proportion of relevant items in the top k retrieved results.
- **Recall@k** – Recall at rank k: The proportion of all relevant items that are retrieved in the top k.
- **Embedding Model** – A model that maps textual inputs to dense vector representations, typically in a high-dimensional semantic space.
- `modernBERT` – The pre-trained embedding model used as the baseline in this paper. It is adapted and evaluated for mathematical document retrieval.

1 Introduction

Nowadays, chatbots based on Large Language Models (LLMs)- such as ChatGPT- have become common and effective tools for automatically generating text. These models are capable of producing coherent and relevant responses across a wide range of contexts, making them valuable for applications ranging from customer support to content generation.

However, despite their usefulness, LLMs are not infallible and, at times, may generate incorrect or imprecise answers when dealing with specific or specialized information. Such errors can be especially problematic in domains like academic research, where the accuracy and reliability of information are crucial.

To address this, Retrieval-Augmented Generation (RAG) systems enhance LLM responses by integrating relevant information retrieved from external knowledge sources. In such systems, the quality of retrieval plays a central role in the effectiveness of the generated response. This retrieval process typically relies on embedding models, which map documents and queries into a shared vector space where their semantic similarity defines relevance of documents given a query. In this approach, the retrieval process is one of the most critical stages, as the quality of the retrieved information directly influences the quality of the generated responses.

In domain-specific scenarios like mathematical research, general-purpose embeddings often fall short. They struggle to capture the nuanced semantics of mathematical language, which includes not only dense technical jargon but also complex symbolic structures expressed through LaTeX. This concern leads to poor semantic alignment between queries and relevant documents, sabotaging the very objective of RAG.

The core problem addressed in this project is the study of adapting general embedding models for mathematical document retrieval. Our goal is to develop a system that can be practically useful in real-world scenarios. For instance, when a user asks a question about a specific mathematical theorem, the system should be able to retrieve the paper where the theorem originates. Similarly, a researcher exploring a new idea could input their concept or theorem and quickly discover related existing work, significantly reducing the time spent on manual literature review.

We do not aim to surpass state-of-the-art benchmarks in general-purpose retrieval, as our focus is domain-specific: mathematical documents written in LaTeX format. To address the domain's structural peculiarities, we apply preprocessing steps to remove non-informative LaTeX elements. Without this preprocessing, the retrieval performance would be significantly worse due to noise and irrelevant formatting commands.

The main objective of this project is to **Evaluate and quantify the impact of domain-specific fine-tuning on retrieval performance**. Given the growing need to limit computational costs and reduce CO_2 emissions, adapting a general-purpose embedding model—rather than training a new one from scratch—offers a practical and sustainable solution. This approach aligns with current industry demands for efficient, resource-conscious AI development.

To further study the retrieval performance variation that fine tuning yields, we introduce the use of Domain-Adaptive Pre-Training (DAPT). This technique consists on further training our baseline model- `modernBERT` - with large-scale mathematical corpora, to force the model to develop awareness of mathematical terminology, syntax, and structure.

To evaluate the effects of fine tuning, we compare the retrieval performance of a baseline model against three other models.

- ModernBERT fine-tuned directly on mathematical retrieval tasks,
- ModernBERT trained with DAPT on mathematical corpora, followed by fine-tuning,
- `nomic-ai/ModernBERT`, a model that has already been fine-tuned and is capable of producing sentence embeddings.

The corpus used in the project is composed of mathematical statements (including Theorems, Lemmas, Definitions and Corollaries) extracted from arXiv papers published between January and March 2025. Despite having access to large-scale datasets containing terabytes of mathematical documents, processing them would far exceed our resource capacity. Therefore, we constrain our dataset to papers published over a selected months window.

For training, we generate synthetic question–answer pairs using the LLaMA 3.2 model, enabling us to use the `SentenceTransformer` framework to teach the model sentence embeddings in a supervised way.

2 Literature Review

The effectiveness of Large Language Models (LLMs), particularly in the context of domain-specific applications such as mathematics, has been extensively discussed in recent literature. Models like BERT (Devlin et al., 2018), GPT-3 (Brown et al., 2020), and ChatGPT have demonstrated remarkable capabilities in generating human-like text, engaging in conversations, and providing explanations. These models and approaches became a development of the transformer architecture (Vaswani et al., 2017) upended sequence modelling by replacing recurrence with purely attention-based encoder–decoder blocks. However, if Vaswani et al.'s original transformer has both an encoder stack (for reading input) and a decoder stack (for generating output), BERT discards the decoder entirely and uses only the transformer encoder layers. Apart from that if the original transformer was trained autoregressively (predict the next token) on translation tasks; BERT instead uses Masked Language Modeling: 15 percent of tokens are masked, and the model must reconstruct them using full left+right context. This forces truly bidirectional self-attention rather than the left-to-right or right-to-left attention in GPT-style models. In the context of large language models development, fine-tuning became an important paradigm emphasizing that once pretrained, BERT may become a state-of-the-art model capable to solve a wide range of problems. However, despite these advances, several studies have identified persistent limitations, especially when dealing with technical and specialized information, including mathematical content Lewis et al., 2020; Raffel et al., 2020.

Retrieval-Augmented Generation (RAG), first proposed by Lewis et al., 2020, presents a promising solution to mitigate the limitations of traditional LLMs. The RAG framework integrates external knowledge bases during the generation process, significantly improving response accuracy and relevance, especially in specialized contexts. This integration has been shown to enhance the factual correctness of generated responses and reduce hallucinations (false statements)—a common issue with LLMs where plausible but incorrect information is produced (Shuster et al., 2021).

An essential component within RAG systems is the embedding model, which maps queries and documents into a shared semantic vector space. The process is usually arranged as: RAG encodes a user's query into a dense vector and retrieves the top k passages from an external corpus using a dense retriever. Then, it fuses those passages with the original query to build an enriched context. Finally, sequence-to-sequence transformer conditions on this combined input to generate its final response. By explicitly grounding generation in retrieved facts, RAG substantially reduces hallucinations and keeps the output more factually consistent with the external knowledge base. Embedding models such as Dense Passage Retrieval (DPR) Karpukhin et al., 2020 - which use two separate encoders (one for queries, one for passages) and allow a very fast nearest-neighbor search, as well as Sentence-BERT Reimers and Gurevych, 2019 - which encodes any two sentences with the same network and captures overall sentence meaning - have gained popularity due to their superior retrieval capabilities. However, general-purpose embedding models often fall short in capturing domain-specific semantics, particularly within mathematics, where the language structure significantly differs from standard text.

To address these challenges, domain-specific embedding models have been explored. Notably, MathBERT (Shen et al., 2021), a variant of BERT specifically pretrained on a mixture of mathematical textbooks, online course materials, and actual LaTeX source from arXiv, demonstrated improved performance in various mathematical natural language processing tasks. Shen et al., 2021 showed that domain-specific pretraining substantially enhances the semantic representation of mathematical expressions, resulting in better downstream task performance. Peng et al., 2021 , by contrast, focus squarely on the challenge of formula understanding rather than general text. They design a specialized pretraining task that jointly masks out entire substructures of mathematical expressions and ask the model to reconstruct them from context. This "mask-the-formula" strategy forces MathBERT (their version) to internalize the syntactic and semantic patterns of the formulas themselves, delivering state-of-the-art performance in tasks such as formula topic classification and headline generation. Before that, less sophisticated approaches were considered - Gao et al., 2017 treated formulas as "sentences" over a vocabulary of symbols (numbers, operators, function names) and train a symbol2vec/formula2vec embedding. Their aim was purely information retrieval, to see if formulas can be retrieved by semantic similarity in the same way that natural-language queries can, without ever touching a heavy Transformer model. Mansouri et al., 2019 extend this line with lightweight, high-quality embeddings for mathematical formulas by treating each formula as a graph and applying word-embedding techniques to it. An interest to the analysis of the math text/information - which lasts for 8 years - resulted in the development of the MathBert model. However, our aim to achieve similar results with the MathBert, but using ModernBert and a number of techniques, which might improve our results.

ModernBert introduced by Warner et al., 2024 became a new updated instrument for NLP tasks. It incorporates a number of "modern" optimizations—such as memory-efficient layer implementations, and support for native 8 192-token inputs (compared to 512 tokens in BERT) and is pretrained on over two trillion tokens. The result is a model that exceeds BERT's accuracy across a wide range of tasks. Using the ModernBert as a foundation we are focused on adapting it to represent domain-specific terminology and symbolic patterns more effectively.

In order to enhance domain adaptation, we applied Domain-Adaptive Pre-Training (DAPT). It has emerged as an effective intermediate step between general pretraining and task-specific fine-tuning. Gururangan et al., 2020 demonstrated the efficacy of DAPT across multiple specialized domains, including biomedical and computer science texts. Their results suggest that exposing pretrained models to domain-specific data before task-specific fine-tuning significantly improves the models' ability to encode and retrieve relevant information. In practice, DAPT consists of taking a pretrained model and continuing masked-language modeling on a large corpus of unlabeled, in-domain texts—randomly masking a fraction of tokens and training the model to predict them using only the neighboring context.

One of the most recent and relevant contributions to mathematical NLP is the work by Horowitz and Hathaway, 2024, which focused on the task of definition extraction from mathematical texts written in LaTeX. The authors fine-tuned three BERT-based models and developed a dedicated preprocessing pipeline that became a key reference for our own data preparation. Their approach involved standardizing math delimiters, preserving the integrity of in-line formulas, and introducing a custom spaCy component called detextor, designed to prevent incorrect sentence segmentation within mathematical expressions.

In summary, the literature spans from architectural innovations (Transformer, ModernBERT) through specialized embedding techniques for formulas (formula2vec, Tangent-CFT), to domain-targeted pre-training (MathBERT, DAPT) and fine-tuning (definition extraction). Yet, few works have joined these strands specifically for mathematical-document retrieval: this thesis aims to bridge that gap by evaluating and integrating these advances under a unified RAG framework tailored for math.

3 Data and Preprocessing

The dataset used for the finetuning was built through a two-step generation. In the first one, we scraped mathematical papers from latex and extracted statements. In the second one, we synthetically generated questions that could be answered by each of the statements, to meet the demands of a fine tuning dataset (pairs of queries and positive passages) .

(a) arXiv Data

The 2025 math-papers dataset consists of mathematical research articles published between *January and March 2025*, sourced from the arXiv preprint server. The dataset includes a total of *12,000 documents*, covering a range of mathematical disciplines such as algebra, topology, analysis, and logic.

These papers are provided in their original LaTeX source format, preserving the mathematical structure and notation used by the authors. Each document typically contains rich mathematical content, including:

- Theorem-like environments such as `\begin{theorem}`, `\begin{lemma}`, `\begin{definition}`, and their variants,
- Display-style math expressions using standard LaTeX math delimiters (`$...$`, `\[...\]`, `\begin{equation}`, etc.),
- Metadata such as paper ID, title, and arXiv subject categories.

On average, we have around 3.7 theorems, 4.6 lemmas, 2.4 definitions, 1.2 corollaries and 2.9 propositions per paper.

(b) Preprocessing

To effectively fine-tune embedding models on mathematical content, an extensive preprocessing pipeline was developed. Initially, raw archives provided by arXiv in the formats of `.tar.gz` or `.gz` were systematically processed. Each archive typically contained one or more files with extensions such as `.tex`, `.ltx`, or `.texx`. The preprocessing commenced with the extraction of these files from the archives. The primary LaTeX source file within each archive was identified through a structured heuristic approach. Specifically, the preprocessing pipeline first checked each LaTeX file for the presence of a `\documentclass` directive. If this directive was absent, the largest LaTeX file was selected, or, if necessary, the first successfully decodable file was chosen as a fallback. Upon identifying the primary document, its content was decoded for subsequent processing.

Following extraction, the LaTeX source was cleaned to remove non-semantic constructs and stylistic artifacts. We stripped out comments, hyperlink commands, and visual formatting macros such as `\textbf{}`, `\emph{}`, and `\section{}` using regular expressions. Mathematical environments were then normalized to a single format, converting all variants—such as `$$...$$`, `\[...\]`, and `\(...\)`—into a consistent inline math representation using dollar signs, i.e., `$...$`.

To extract meaningful mathematical content, we applied a rule-based recognizer focused on theorem-like environments. Specifically, we targeted standard structures including `theorem`, `lemma`, `definition`, `proposition` and `corollary`. These were detected either directly or through user-defined macros via the `\newtheorem` declaration. The content of each recognized environment was stored in a structured format for downstream processing.

A particular challenge arose in sentence segmentation due to the presence of embedded math expressions. To address this, we employed the `spaCy` NLP library with all processing modules disabled except the sentence segmenter. A custom pipeline component—named `detextor`—was applied to ensure that sentence boundaries were not introduced inside mathematical expressions enclosed in dollar signs. For example, consider the following sentence from a LaTeX source:

```
"Let $f(x) = x^2$ be a function. Then $f'(x) = 2x$."
```

Without careful handling, a naive sentence segmenter might erroneously split the expression at `$f(x) = x^2$` or interpret mathematical symbols as punctuation, thereby corrupting the structure and meaning of the sentence. The `detextor` component ensures that everything between a pair of dollar signs is treated as a single atomic unit during sentence splitting. Additionally, the extremely long statements were dropped to avoid memory-related issues.

The cleaned and segmented documents were serialized into a `.json` format, where each line corresponded to a single paper. Each record included the paper's identifier, metadata, and lists of extracted environments. This structured dataset enabled consistent downstream embedding and model fine-tuning, while preserving the syntactic and semantic integrity of the original mathematical expressions.

(c) Query Creation

To automatically generate natural language questions from mathematical statements, we designed a pipeline using a large language model. The goal was to produce one well-formed question per theorem that could be used in a question-answering system. This is a very delicate part, as it is widely known that the quality of a dataset largely impacts the performance of the model it is fed to.

We chose to use `LLaMA 3.2 Instruct 3B`, because it is instruction-tuned (meaning it can follow prompts that specify a task), relatively lightweight for a large model, and capable of understanding mathematical language and LaTeX notation. Unlike larger models, it can be run on accessible hardware such as the GPUs provided in Google Colab Pro. Using the GPU L4 with 22.5 GB RAM, the model generated questions at a rate of approximately 1.08 s/it. This means that, to generate 100.000 questions, the model took around 30.5 hours to run.

Each statement is inserted into a carefully constructed prompt that uses LLaMA's chat format: a system message enclosed in «SYS» ... «/SYS» to define the expected behavior, and a user instruction enclosed in [INST] ... [/INST] that describes the specific task (in our case, "generate one question answerable by this theorem"). We ask the model to return its answer in a strict JSON format — a single object with a "questions" key containing one question as a string. This makes parsing and post-processing more reliable.

We run inference in batches for efficiency, using Hugging Face's text-generation pipeline. To keep the output consistent and reduce unnecessary verbosity (e.g., phrases like "Sure! Here's a question…"), we use deterministic decoding and set the temperature to 0. This prevents the model from generating random or overly chatty responses. Naturally, it also affects the diversity of the generated questions.

Despite these efforts, we encountered some typical issues. For example, many mathematical expressions use LaTeX syntax with backslashes (like leq, ker, etc.), which are not valid in JSON unless the backslash is doubled. If we try to parse such output as JSON directly, it fails. To handle this, we post-process the model's output by escaping all backslashes before attempting to parse the JSON. We also use regular expressions to reliably extract the JSON portion of the response in case the model adds extra text before or after. After filtering out the failed queries, we are left with 88,775 question-statement pairs, ready to parse into the Fine Tuning trainer. Below is an example of what a typical output of the query generator looks like:

Question Generator LLM output

Sure! Here is a question that could be answered by citing the theorem:
{ "questions": ["What is the value of $f(c)$ for $c \in (a, b)$ if f is a continuous function on $[a, b]$?"] }
Explanation: The theorem states that there exists a value $c \in (a, b)$ such that: $f(c) = \frac{1}{b-a} \int_a^b f(x)\,dx$. Therefore, if we know this value, it gives us the average value of f on the interval $[a, b]$.

Note: Both in the "questions" and in the "Explanation", the LLM output format is in LaTeX.
It is worth mentioning that relying on a large language model to generate questions introduces potential limitations. Since all training questions follow the linguistic patterns and styles of the LLaMA-generated prompts, the fine-tuned model may underperform during inference, when faced with human-written queries that differ in structure, tone, or specificity.

Hard Negative Mining

To improve the effectiveness of our retriever during fine-tuning, we explore the employment of hard negative mining: selecting passages that are lexically and semantically similar to the query but are incorrect answers. These hard negatives play a crucial role in contrastive learning, helping the model learn to differentiate between truly relevant passages and deceptively similar distractors.

The mining pipeline consists of two key stages: lexical filtering using BM25 and semantic reranking using embedding similarity.

- **Lexical Filtering (BM25)**

In the first stage, we use the BM25 ranking function to retrieve a shortlist of candidate passages. Crucially, we use the query (question) text as the BM25 query, not the corresponding positive passage. This design choice reflects the model's intended deployment setting: at inference time, the retriever will receive a natural language question and must retrieve relevant content from a large passage index. By using the question for BM25, we ensure that the negatives we mine reflect realistic retrieval errors—passages that a lexical retriever might mistakenly rank highly for a given question.

We restrict BM25 retrieval to passages that share at least one top-level category with the query's associated passage, grouping rare categories into a generic "math.other" bucket to ensure broad coverage. We also explicitly exclude passages from the same paper as the ground truth, preventing the model from learning trivial lexical similarities within the same document.

- **Semantic Re-ranking (Cosine Similarity)**

After BM25 filtering, we re-rank the candidate passages using cosine similarity between the question embedding and each candidate's embedding. The embeddings are precomputed from the model being fine-tuned and are L2-normalized, so that the cosine similarity reduces to a dot product.

This semantic filtering ensures that the final set of negatives includes those passages that are closest to the query in embedding space, despite being incorrect. These are the most informative mistakes—passages that the model currently finds confusing—and are therefore ideal for contrastive training.

To employ this hard negative selection in our FT pipeline, we use a loss function called `TripletLoss`, which only uses a single hard negative. Following this approach we observe that results certainly improve all base models, yet the performance gap between base and fine tuned models is even greater when using multiple hard negatives, for which we employ a different loss function (specified in the Loss Function section) that randomly selects the negative passages.

Therefore, we leave this approach as a potential strategy to improve the hard negative selection for future work.

Train-Test split

To evaluate our fine-tuned retriever in a way that reflects its intended use, we design a train/test split focused on measuring how well the model **generalizes to new queries**, while ensuring that test queries come from papers the model has had sufficient exposure to during training. This aligns with our goal of building a domain-specific retrieval system that will be incrementally fine-tuned as new documents are added.

We begin by assigning all papers with fewer than 4 statements to the training set only. These low-statement papers are not suitable for evaluation because they don't provide enough context for meaningful testing. By excluding them from the test set, we ensure that all test queries originate from papers that contain a richer set of statements—allowing the model to have trained sufficiently on that content and thus making the evaluation of query generalization more reliable.

Additionally, we select a small group of papers with approximately 14 statements, close to the average paper length, and reserve them entirely as an unseen-doc test set. This allows us to assess how well the model performs when retrieving from completely new documents—a relevant scenario when the corpus is expanded and the model has not yet been updated.

From the remaining data, we sample entries to ensure that **90% of the total dataset** (including the low-statement papers) is used for training, and **10% is held**

out for testing. This final test set includes both the held-out average-length papers (to measure document-level generalization) and queries from seen papers (to evaluate query-level generalization).

By structuring the split this way, we ensure that:

- The model learns from a representative subset of the domain.
- We can separately measure query-level and document-level generalization.
- Our evaluation aligns with the practical use case of incrementally updating the retriever as new mathematical documents are introduced.

The final structure of our corpus can be found in the table below, where *Count* refers to the count of statements that fall into each category.

Split	**Count**	**Percentage**
Train	79 897	90.0%
Regular Test	8 668	9.8%
Unseen Test	210	0.2%
Total Test	8 878	10.0%
Total	88 775	100%

Table 1: Statement Count for the split of our final dataset

(d) Final dataset structure

The final dataset structure resulting from the preprocessing and query creation is detailed in Table 2, and includes the paper identifiers, titles, subject categories, and categorized extracted mathematical content, as well as the mathematical statement and the generated question that could be answer by it. Note that the sentences in *Statement* and *Question* columns contain parsed LaTeX code, which appears as raw unrendered text in our dataset.

The final structured dataset consists of 88,775 statements spread across 4843 unique papers, systematically capturing mathematical content across the extensive corpus of arXiv mathematical papers from the first quarter of 2025. Below can be found the distribution of statements per paper.

Figure 1 shows the slight skewness of statements distribution across papers, which can also be reflected in Table 3: a median of 13 against a mean of 17.31 suggests that many papers have fewer statements, while a minority contain large counts. The 25th–75th range captures exactly half of all papers, indicating that 50% of them have between 7 and 23 statements. However, a standard deviation of 14.64 underscores that paper lengths vary widely, which means that some metrics need to adapt correctly.

Table 2: Sample of Query–Statement Pairs for Fine-Tuning

ID	Paper ID	Title	Categories	Statement	Question
53216	2502.00867	partitions of an eulerian digraph into circuits	[math.co]	Let $\mathcal{A}$ be a finite set. We define the partition lattice $\Pi\mathcal{A} = \pi\mathcal{A}, \leq$...	What is the relationship between refinement and concurrence in the partition lattice $\Pi\mathcal{A}$?...
41436	2501.15784	constructing stable hilbert bundles via diophantine approximation	[math.dg, math.ag, math.ph]	For $i = 0, 1$, we have $L_i(\theta) \geq 1$, and equality implies attainability...	Can $L_i(\theta)$ be greater than 1 for some θ?...
61254	2502.04123	localizing invariants of inverse limits	[math.at, math.ct, math.nt]	Let $(_n)$ be a strongly Mittag-Leffler sequence. Then $((^\kappa_n))$ and...	What is the connection between the properties of $(_n)$ and the induced sequences?...
36828	2501.14069	bounded toeplitz products on the hardy space	[math.fa]	$u \in H(\mathbb{T}), v \in L^2(\mathbb{T})$ form an admissible Toeplitz pair if...	Does admissibility imply the poles of u and v coincide?...
26721	2501.10495	cohomology and deformations of nonabelian embedding tensors	[math.ra]	Let θ be a coherent action of a Lie triple system on another...	Does the nonabelian hemisemidirect product satisfy the Leibniz rule?...

Statistic	Value
Count	4795
Mean	17.31
Std. Deviation	14.64
Min	1
25th Percentile	7
50th Percentile	13
75th Percentile	23
Max	89

Table 3: Distribution of statements per paper.

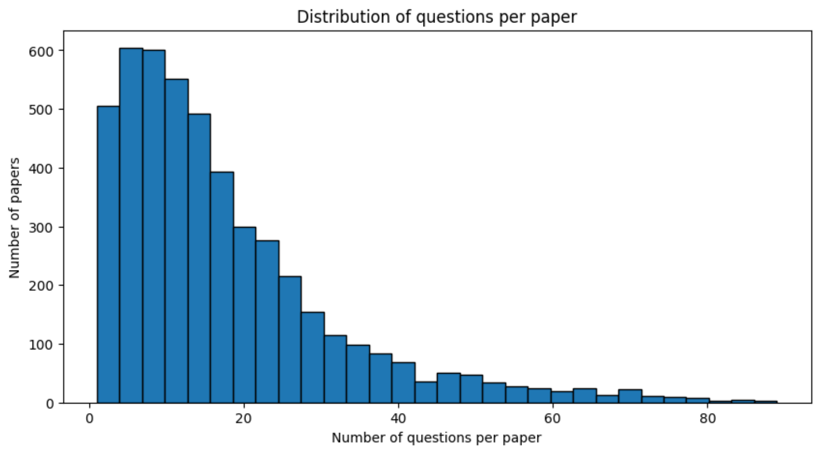


Figure 1: Distribution of Statements per Paper.

4 Methodology

This study aimed at evaluating the embedding model performance within retrieval-augmented generation (RAG) systems for mathematical documents through a pipeline, which includes: data preparation, model pretraining, fine-tuning, and retrieval evaluation. The objective of this research was to enhance the relevance of retrieved mathematical statements (e.g., theorems, lemmas) and analyzing their relevance to natural language queries. This is accomplished by fine-tuning transformer-based sentence embedding models on a domain-specific dataset.

The methodology comprises following core stages: query construction, domain-adaptive pretraining (DAPT), model fine-tuning, and retrieval evaluation.

- Query construction - the generation of query–text pairs. Each query is a natural language question automatically generated to correspond with a specific mathematical statement. The paired passage is an excerpt (e.g., a theorem or lemma) extracted from LaTeX-formatted research papers. The questions were generated using the LLaMA 3.2 Instruct 3B, a very lightweight yet stable model, accessible for low memory GPU's.
- Domain-Adaptive Pretraining (DAPT) was applied prior to fine-tuning to enhance the base model's proficiency with mathematical language. This involved continued pretraining of the ModernBERT encoder using masked language modeling (MLM) on a large corpus of unlabeled `arXiv` papers. This step allows the model to learn mathematical vocabulary, LaTeX notation, and formal logic, providing a more suitable initialization for subsequent fine-tuning.
- Fine-tuning : The encoder is then fine-tuned on the query–passage dataset using a contrastive learning objective (Multiple Negatives Ranking Loss).
- Retrieval evaluation: The trained model is evaluated on its ability to retrieve relevant statements, using standard information retrieval metrics such as Recall@k, Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain (nDCG).

(a) Model architecture

In this section we aim to understand how modern language models represent and structure mathematical statements, that is why our analysis proceeds from foundational architectural principles to detailed comparisons of tokenization, attention behavior, and embedding geometry. In general, we adopted a layered approach, beginning with a base model and gradually introducing domain adaptation and task-specific fine-tuning. As we mentioned before, our study is centered around three transformer-based models:

- **ModernBERT (Base)** — a domain-adapted BERT model, which is a baseline encoder, exposed to technical vocabulary, but not fine-tuned.
- **Fine-Tuned ModernBERT** — a version of the base model further trained using a retrieval objective.

- **MathBERT** — a publicly available BERT model pretrained from scratch on mathematical corpora, such as arXiv. While MathBERT is tailored to the domain, it is not fine-tuned for document-level retrieval. This model is considered as a reference point for understanding how pretraining helps with mathematical language, and whether it´s possible to achieved similar results through fine-tuning.

These models reflect a continuum of modeling strategies: from domain adaptation without task-specific supervision (Base), to task-aligned fine-tuning (Fine-Tuned), to broad mathematical pretraining without structural optimization for retrieval (MathBERT). Table 4 represents the basic architecture of considered models.

Table 4: Architectural comparison of the models

Component	**FT ModernBERT**	**Base ModernBERT**	**MathBERT**
Embedding Class	`ModernBertEmbed-s`	`ModernBertEmbed-s`	`BertEmbeddings`
Token Embeddings	50368	50368	30522
Position Embeddings	—	—	512
Segment Embeddings	—	—	2
Dropout (Embeddings)	0.0	0.0	0.1
LayerNorm (Embeddings)	Yes	Yes	Yes
Transformer Layers	22	22	12
Attention Type	Rotary QKV	Rotary QKV	Multi-head QKV
Attn Norm (Layer 0)	Identity	Identity	LayerNorm
Attn Norm (Other Layers)	LayerNorm	LayerNorm	LayerNorm
MLP Dimensions	$2304 \to 1152 \to 768$	$2304 \to 1152 \to 768$	$3072 \to 768$
MLP Norm	LayerNorm	LayerNorm	LayerNorm
Dropout (FFN/Attn)	0.0	0.0	0.1
Activation Function	GELU	GELU	GELU
Final Norm	LayerNorm	LayerNorm	LayerNorm
Pooling Strategy	Mean + Norm	Mean	Pooler
Special Modules			
Rotary Embedding	✓	✓	—
Identity Dropout	✓	✓	—
Explicit Pooler Layer	—	—	✓
Pos. & Seg. Embeddings	—	—	✓
BertSelfOutput	—	—	✓

In general, considered models have classical Transformer architecture. This architecture is based on self-attention, which enables BERT to calculate the importance of each word by considering its context — both the words that come before and after it. At a high level, the process includes the following stages: tokenization, embedding, encoding via self-attention, and pooling to generate sentence-level representations. While this overall structure remains consistent across models, specific components and design choices can vary. In our case, for instance, both Fine-Tuned and a baseline `modernBERT` share a newer architecture, consisting of 22 transformer layers and using rotary position embeddings, which make the `modernBERT` models deeper than Vanilla BERT (with only 12 layers). Additional layers allow `modernBERT` to learn more complex patterns, which is especially helpful in understanding detailed mathematical expressions.

Instead of using fixed position information (like "this token is at position 3"), `modernBERT` uses rotary embeddings, which help the model focus on the relative position between tokens — which is very useful when working with equations or symbolic text, where meaning often depends on token order and structure. In contrast, MathBERT follows the original BERT design, with only 12 layers, and uses absolute position embeddings along with segment embeddings and a classification pooler (which is often used for classification tasks). `modernBERT` removes these and instead relies on simpler strategies like averaging the output vectors (mean pooling), which often works better for sentence embeddings and retrieval tasks.

Both ModernBERT models use very little dropout (a method to prevent overfitting), which helps preserve more information. They also apply normalization differently: the first layer in `modernBERT` uses no normalization at all (just an "Identity" function), while MathBERT applies LayerNorm consistently. All in all, despite the simpler structure of MathBERT, its strength lies in its domain-specific pretraining on mathematical texts. `modernBERT`, in contrast, adopts a more flexible and deeper architecture that becomes highly effective when fine-tuned for mathematical retrieval. Hence, fine-tuning is considered not only to align the model's parameters with the target task (in our case retrieval of mathematical documents) but also compensates for architectural elements, such as the lack of training for sentence embeddings.

In order to further clarify how these architectural choices work in practice, we investigate each model's behavior during the main stages of input processing: tokenization, embedding construction, and contextual encoding through self-attention - see Table 5. These components reveal how each model interprets and transforms mathematical language into internal representations.

Stage	**Function**
Tokenization	Converts raw text into subword units using domain-sensitive tokenizers
Embedding	Maps each token (plus its position and segment) into a high-dimensional vector space
Self-Attention	Learns contextual dependencies among tokens across the entire sequence
Pooling	Aggregates token-level embeddings into a fixed-size vector representing the entire sentence or theorem

Table 5: Principal stages of the BERT-based embedding pipeline

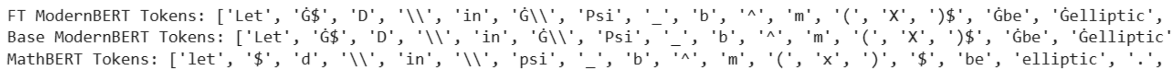
(i) Tokenization

It is the first step in any transformer-type architecture: it defines how input is broken down before being passed to the model for processing. What we know from the theory - transformers like BERT models cannot read raw text; instead, they process inputs as sequences of discrete units called *tokens*. These tokens may correspond to full words, subwords, or even single characters, depending on the tokenizer's vocabulary and underlying strategy. All models in this study rely on variants of the WordPiece tokenizer, which is specifically designed to handle rare and compound words by decomposing them into smaller, more manageable subword fragments. However, in our case, the inputs consist of mathematical theorems and definitions, often written in LaTeX-style notation, which poses a unique challenge to standard NLP tokenizers.

Let consider this example:

```
Let $D \in \Psi_bm̂(X)$ be elliptic.
```

When this sentence is processed by a model, it is first converted into a sequence of tokens. This step also includes the addition of special tokens, such as `[CLS]` at the beginning and `[SEP]` at the end of a sequence, which serve structural purposes in encoding.



```
FT ModernBERT Tokens: ['Let', 'Ġ$', 'D', '\\', 'in', 'Ġ\\', 'Psi', '_', 'b', '^', 'm', '(', 'X', ')$', 'Ġbe', 'Ġelliptic',
Base ModernBERT Tokens: ['Let', 'Ġ$', 'D', '\\', 'in', 'Ġ\\', 'Psi', '_', 'b', '^', 'm', '(', 'X', ')$', 'Ġbe', 'Ġelliptic'
MathBERT Tokens: ['let', '$', 'd', '\\', 'in', '\\', 'psi', '_', 'b', '^', 'm', '(', 'x', ')', '$', 'be', 'elliptic', '.',
```

Figure 2: Tokenization output for the same LaTeX-style sentence across models. MathBERT splits into more subwords, while Fine-Tuned and Base ModernBERT preserve symbolic tokens.

While these stages are shared across all models, the tokenizers themselves exhibit substantial differences. This is evident in how they segment the same mathematical input. For example, MathBERT tokenizes more aggressively than both variants of ModernBERT, producing a sequence of 231 tokens versus 211 for both the Base and Fine-Tuned ModernBERT models. The increased length reflects a more fine-grained parsing of mathematical subwords in MathBERT.

Beyond token counts, the vocabulary divergence is even more pronounced. While the Base and Fine-Tuned ModernBERT models share 91 unique tokens for the input text—unsurprising given their shared backbone—the overlap with MathBERT drops to just 23 tokens. This suggests not only a different token segmentation scheme but also a fundamentally different lexical structure encoded during pretraining. These differences are visualized in Figure 3, which shows the distribution of the 30 most frequent tokens produced by each tokenizer.

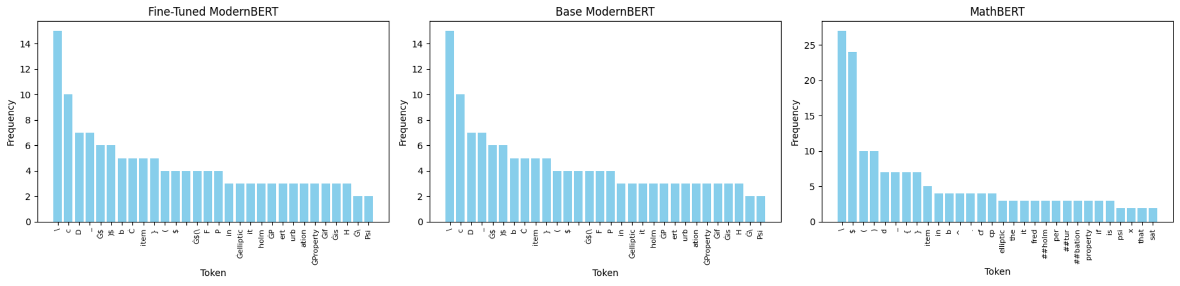


Figure 3: Token frequency comparison across models.

ModernBERT models reveal a vocabulary heavily influenced by technical formatting tokens. Common entries include `\`, `$`, `D`, `Psi`, and `item`, suggesting that these models are relatively conservative in their subword segmentation. Their top tokens are largely syntactic. In contrast, MathBERT exhibits a substantially different token profile. The most frequent tokens include not only standard symbols such as `$` and `\`, but also more fragmented lexical items such as `##tur`, `##bation`, `##holm`, and `fred`. This granularity may allow MathBERT to better generalize over previously unseen or morphologically complex mathematical terms, but it comes at the cost of increased sequence length and potentially noisier attention dynamics.

Taken together, the differences in token length, vocabulary overlap, and token frequency profiles reflect divergent modeling principles. The ModernBERT variants emphasize domain adaptation via continued pretraining on mathematical corpora while preserving standard token boundaries. MathBERT, by contrast, adopts a more data-driven segmentation from the outset.

(ii) Embedding

This is the next step after tokenization. The procedure is arranged in the following way: each token produced by the tokenizer is mapped to a fixed-dimensional vector through an embedding layer - it includes three components:

- **Token embeddings**: The model uses a lookup table to map each token ID to a vector of fixed size (for example, 768 numbers). These vectors are not random; they are learned during pretraining so that they capture semantic properties of tokens (e.g., the token `$` or \ will always be associated with the same vector, and the model learns what these symbols mean based on how they are used across many training examples).
- **Position embeddings**: Transformer models process the entire input at once (not step by step); hence, they need an extra signal to understand the order of the tokens. This is what position embeddings do. They assign each token a vector that corresponds to its position in the input. As a result, the model knows not just what the word is, but where it appears in the sentence - used only in MathBert model.
- **Segment embeddings:** These are used when the model is handling a pair of inputs, like two sentences in a question-answering problem. Segment embeddings help the model identify which token belongs to which segment (e.g., sentence A or sentence B) - used only in the MathBert model.

The sum of these three components is passed as input to the transformer encoder stack. Models like ModernBert do not use position and segment embeddings. Instead, ModernBert adopts rotary embeddings, which better identify the relative position.

To assess how token representations differ between models, we analyzed the L2 norms (Euclidean norms) of token embedding vectors. The L2 measures the magnitude of the vector or how far the vector is from the origin in the embedding space $\mathbf{v} = [v_1, v_2, \ldots, v_n]$ as:

$$\|\mathbf{v}\|_2 = \sqrt{\sum_{i=1}^{n} v_i^2}$$

For example, the embedding of the token `"math"` is represented by the vector:

$$\mathbf{v}_{\texttt{math}} = [0.21,\ -0.13,\ \ldots,\ 0.48] \in \mathbb{R}^{768}$$

In the context of transformer-based models such as BERT, L2 norms show the length of a token's embedding vector. While the direction of a vector governs semantic similarity, its magnitude reflects other properties, such as the frequency or informativeness of a token. Typically, tokens that are more general or frequent (e.g., punctuation or stopwords) have lower norms, while some specific or rarer tokens (e.g., "theorema") may have higher norms.

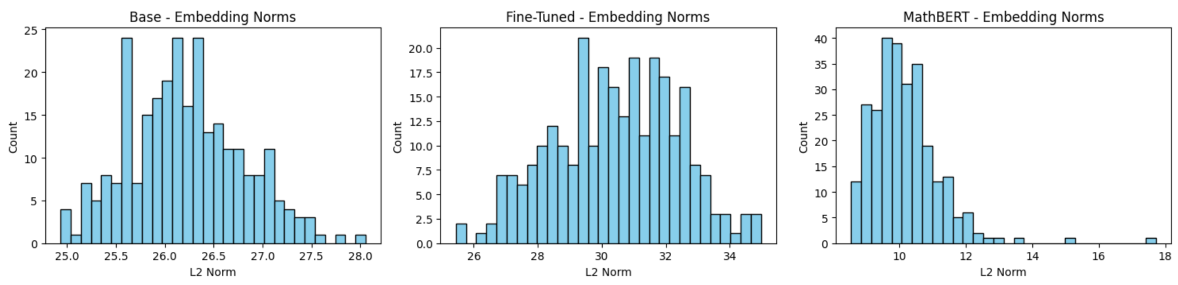


Figure 4: Distribution of L2 norms of token embeddings across models. Note:"Base ModernBERT" denotes the ModernBERT model after DAPT.

To visualize the embedding space structure across models, we computed the L_2 norm for our three models. The results are presented in Figure 4, which shows histograms of the token embedding norms for each model.

The Base model exhibits a relatively compact distribution, centered on norms of approximately 26 to 27. This suggests a uniform embedding space in which most token vectors have similar magnitudes. After fine-tuning, the Fine-Tuned ModernBERT model shows a shifted and broader distribution, with many tokens having norms in the range of 30–33. This increase in vector length may show that the model now makes the representations of different tokens stand out more from each other, which can help the model recognize and use these differences better in later stages.

In contrast, MathBERT demonstrates a markedly different distribution: most token embeddings cluster around small norms between 9 and 12. This suggests a different encoding strategy, possibly related to its pretraining on formal mathematical texts where symbols, operators, and frequently occurring mathematical constructs receive compressed, low-magnitude embeddings.

These differences highlight how fine-tuning and domain-specific pretraining reshape the geometry of the embedding space, potentially affecting how tokens are distinguished and retrieved in later stages of the pipeline.

(iii) Transformer encoder: self-attention layers

After formulating how each token in the input sequence is mapped to a high-dimensional embedding vector, it´s important to consider a core mechanism that enables transformer models to identify complex dependencies and contextual information: the encoder and, more specifically, the self-attention mechanism.

In transformer architectures, the sequence of token embeddings forms the initial input to a stack of encoder layers. Each encoder layer is designed to refine these representations by integrating information from other tokens in the sequence. This is achieved through self-attention, a process in which every token can "attend to" (i.e., consider and weigh) every other token, regardless of their position in the sequence.

Mathematically, the attention mechanism is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

where Q, K, and V are the query, key, and value matrices derived from the input embeddings, and d_k is the key dimensionality. Intuitively, this formula enables the model to compute, for each token, a weighted sum of all token representations in the sequence, with higher weights given to more relevant tokens.

This allows the model to assign higher importance to certain tokens when computing a new representation for each token. The result of self-attention is a new set of vectors for each token, which are called contextualized embeddings. Each encoder layer repeats this process, allowing the model to gradually build more context-related representations.

To analyze how different models represent self-attention, we might see from attention heatmaps (Figure 5).

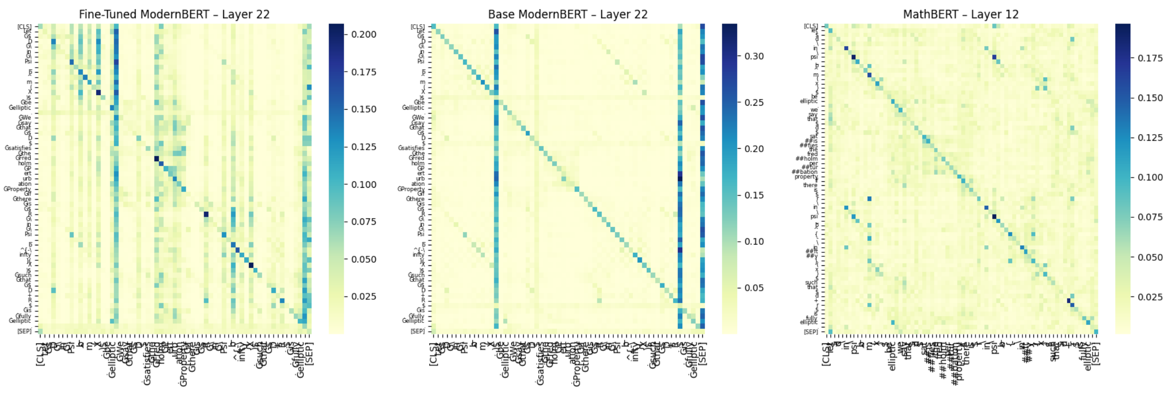


Figure 5: Mean attention per token across models. Note:"Base ModernBERT" denotes the ModernBERT model after DAPT.

These show the average attention weights between all pairs of tokens for the final encoder layer in each model (layer 12 for MathBERT, layer 22 for ModernBERT). Each heatmap is a matrix where the (i, j) entry represents how much attention token i pays to token j. As we can see from these plots, all three models show dominant diagonal attention, which implies that tokens mostly attend to themselves or immediate neighbors and some string related to specific tokens - like [SEP]. However, both the Fine-Tuned ModernBERT and MathBERT exhibit more complex off-diagonal structures, indicating their capacity to capture longer-range dependencies — which is an important feature for understanding any documents, and mathematical text in particular.

From the attention heatmaps, we can analyze the global structure of attention. However, to explore how attention is distributed to specific tokens, let us consider the [CLS] token. We chose this token because [CLS] is typically used as a summary embedding for the entire sequence. This means that examining the attention weights from [CLS] to other tokens can reveal which parts of the input the model considers most important for the overall meaning or task. To visualize this, we plotted the average [CLS] attention for each token across the last five layers of each model, as shown in Figure 6.

From this analysis we can say that in both ModernBERT models, [CLS] attention becomes highly concentrated on a small number of tokens in the last layers—typically those carrying the main mathematical or logical content. MathBERT, in contrast, shows a more distributed [CLS] attention pattern, integrating information from a wider range of input tokens.

By examining both the mean attention maps and these [CLS]-centric plots, we can conclude that global attention heatmaps show that the fine-tuned ModernBERT learns

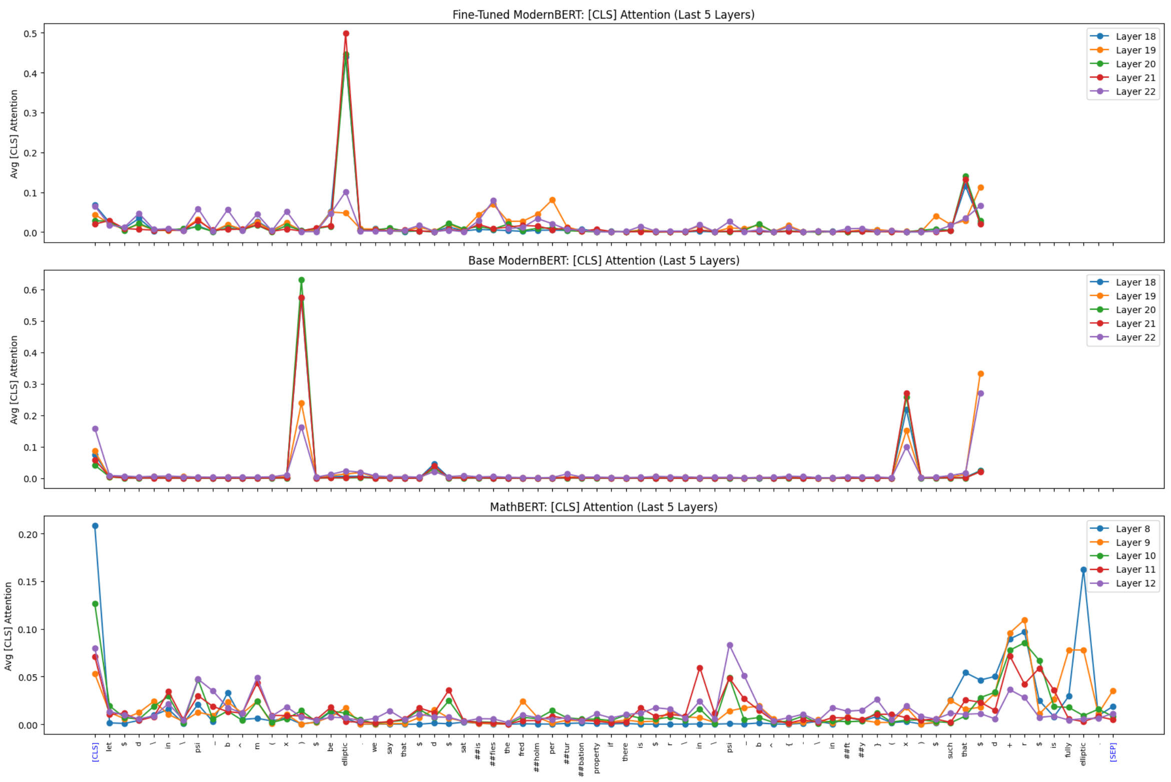


Figure 6: Attention across models. Note:"Base ModernBERT" denotes the ModernBERT model after DAPT.

richer structural relationships among tokens, indicating successful domain adaptation. However, when analyzing [CLS] attention, MathBERT more effectively identifies and attends to key mathematical tokens, reflecting its specialized training. This suggests that fine-tuning improves the model's general capacity to capture complex context, while domain-specific pretraining (MathBERT) can capture relationships that are specific to the target domain.

(iv) Pooling layer: sentence-level representation

After the final transformer (self-attention) layer, the model generates a sequence of contextualized token embeddings, one for each token in the input. However, for most downstream tasks—such as retrieval, clustering, or classification—a single, fixed-length vector representation of the entire sequence (sentence, theorem) is needed. This is achieved via a pooling operation.

In our experiments, mean pooling is used for ModernBERT models, while MathBERT employs [CLS] pooling via its explicit pooler layer. Mathematically, mean pooling can be defined as:

$$\mathbf{z}_{\text{sentence}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{z}_i$$

where $\mathbf{z}_i$ is the embedding of token i, and n is the number of tokens.

For retrieval and similarity tasks, an additional L2 normalization step is often applied to the pooled sentence embeddings to ensure they lie on the unit hypersphere, which is desirable when computing cosine similarity.

To investigate the effect of pooling strategies on sentence embeddings, we compared the L2 norms of all token embeddings against the norm of the resulting sentence embedding, using both mean pooling (for ModernBERT models) and [CLS] pooling (for MathBERT) see Figure 7.

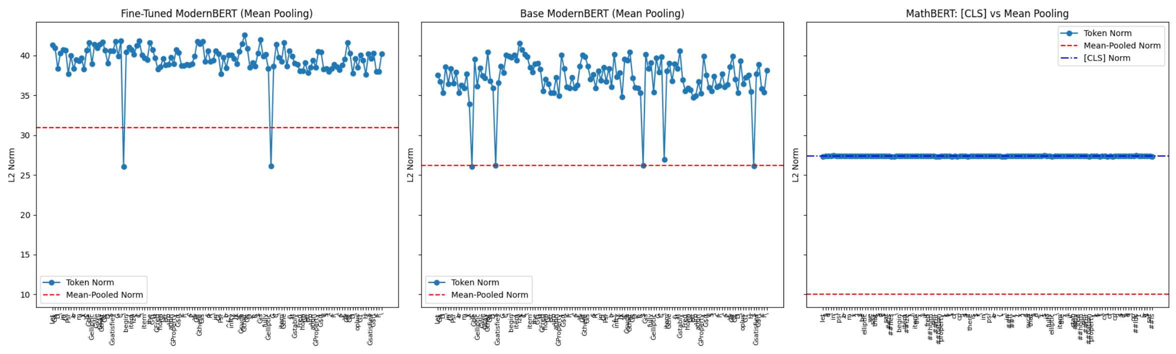


Figure 7: Token embedding norms vs. sentence embedding ([CLS] for MathBERT). Note:"Base ModernBERT" denotes the ModernBERT model after DAPT.

The resulting plots reveal that in ModernBERT models, the mean-pooled sentence embedding is consistently shorter than most token vectors, indicating that information is averaged and possibly compressed. In MathBERT, the [CLS] embedding has a much larger norm than the mean-pooled vector, highlighting the architectural difference in how sentence meaning is aggregated. These differences may influence retrieval performance and the expressiveness of the sentence representation.

(v) Sentence embedding construction

After contextual encoding and aggregation by pooling and normalization, each model maps mathematical statements to sentence embeddings in a shared high-dimensional space. The geometry and distribution of these sentence representations, which are determined by the preceding architectural choices, are directly observable through pairwise cosine similarity matrices, which is the key metric for assessing, how closely related different theorems are in the embedding space:

$$\text{cos\_sim}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \, \|\mathbf{b}\|} \tag{1}$$

where **a** and **b** are sentence embedding vectors.

Figure 8 shows the cosine similarity matrices for all three models. Each heatmap displays the pairwise cosine similarities between theorem embeddings, with brighter diagonal patterns reflecting high self-similarity. The base ModernBERT model produces embeddings that are uniformly similar, which is proved by the generally high off-diagonal values. In contrast, the fine-tuned ModernBERT demonstrates greater separability between different theorems, resulting in a sparser, more blue-dominated off-diagonal structure. MathBERT, while also domain-pretrained, shows an intermediate pattern: exhibiting more structure than the baseline, but less separability than the fine-tuned model. These observations highlight how pooling, normalization, and fine-tuning jointly shape the expressiveness and retrieval capacity of embedding spaces in mathematical language models.

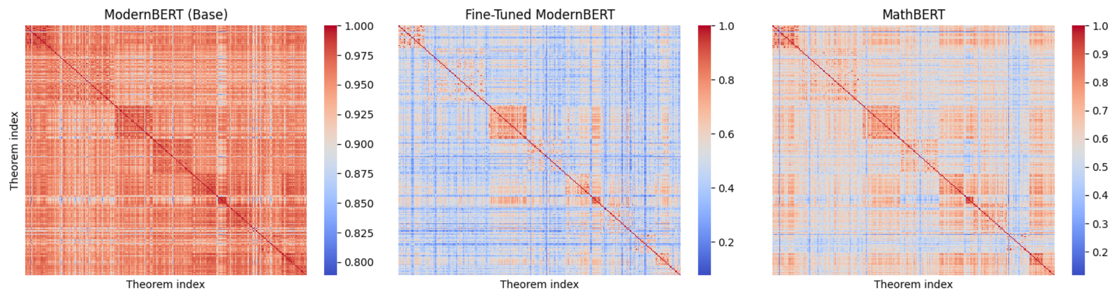


Figure 8: Cosine similarity heatmaps of theorem embeddings for the base ModernBERT, Fine-Tuned ModernBERT, and MathBERT. Note:"Base ModernBERT" denotes the ModernBERT model after DAPT.

(b) DAPT

As mentioned earlier, one of the challenges we aim to address is the limited capability of language models to handle domain-specific mathematical and LaTeX vocabulary. Models such as BERT and ModernBERT have not been pre-trained on data that includes mathematical expressions or LaTeX syntax.

For example, consider the following sentence:
"Let n be an integer and let $\frac{\log^{15} n}{n} \le p \le \varepsilon$ and $q = \sqrt{\frac{p}{n \log^8 n}}$. "

Where with Latex is written as:

```
Let $n$ be an integer and let $\frac{\log^{15}n}{n}\le p \le
\varepsilon$ and $q = \sqrt{\frac{p}{n \log^8 n}}$.
```

If we remove the LaTeX structure (meaning `\begin`, `\item` and more structural LaTeX) and preserve only the content and contextual elements, a standard model not trained on LaTeX-formatted data would tokenize this sentence in a suboptimal way, failing to capture the mathematical semantics accurately.

In table 13 we can observe that tokenization not only expands a single formula into many subword tokens, but also results in a loss of semantic meaning for certain mathematical symbols and expressions—such as Greek letters—that are not part of BERT's or ModernBERT's original vocabulary. Expressions that are not included in the token matrix are split into smaller subcomponents (e.g., `\sqrt` becomes `\`, `s`, `##q`, `##rt`).

A particularly **important observation** is that punctuation characters such as `$`, `\`, `{`, `}`, `/`, `^`, `=`, `>`, `(`, `)`, and `.` are treated as individual tokens. However, in the **context of mathematical or LaTeX expressions, such punctuation should not be interpreted in the same way as in natural language**. Their roles and meanings are structurally and semantically distinct within mathematical environments.

This issue is not limited to punctuation; the same applies to complete mathematical expressions, which often lose coherence when improperly tokenized. For this reason, and to enhance the performance of Retrieval-Augmented Generation (RAG) systems in mathematical domains, we explored the use of Domain-Adaptive Pretraining (DAPT) techniques to specialize the model in mathematical vocabulary and structure.

Domain-Adaptive Pretraining (DAPT) consists of resuming the pretraining phase of a language model in our case ModernBERT—on domain-specific data. For our oun part, this domain consists of mathematical documents (e.g., research papers) with appropriate preprocessing applied. The goal is to adapt the model to the vocabulary, structure, and semantics specific to the mathematical domain.

DAPT serves as an intermediate step between the original pretraining and the task-specific fine-tuning, allowing the model to better capture domain-specific patterns before it is adapted to downstream tasks.

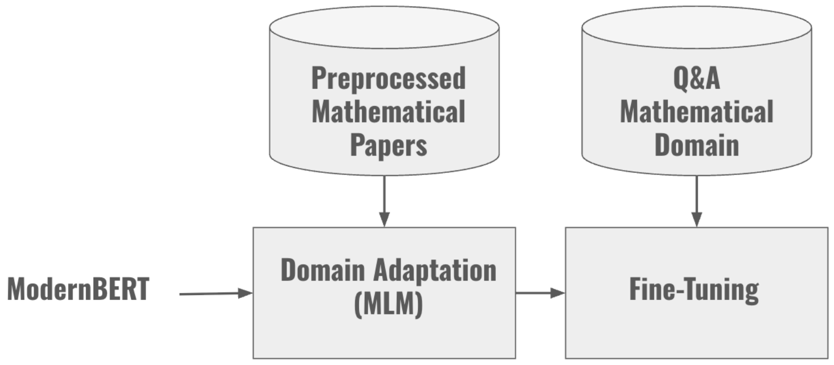


Figure 9: Representation of the DAPT flow with the ModernBERT

This stage is sometimes referred to as:

- **Domain-Adaptive Pretraining (DAPT)** when the model is pretrained on unlabeled data from a broad domain such as economics or medicine.
- **Task-Adaptive Pretraining (TAPT)** when the model is pretrained on unlabeled data that closely resembles the specific downstream task (e.g., economic abstracts prior to a classification task).

By applying DAPT, we aim for the model to update its weights and parameters in a way that aligns more closely with the mathematical domain. In particular, embeddings are sensitive to positional and contextual relationships between tokens. In mathematical text, combinations of punctuation and symbols (e.g., `\`, `^`, `=`, `{`, `}`) are common and carry semantic meaning, whereas such combinations are rare or nonexistent in general corpora like Wikipedia. As a result, models pretrained only on general-domain data tend to perform poorly when exposed to mathematical input.

This additional DAPT step is thus crucial for making the model more robust and effective in handling domain-specific structures, ultimately leading to improved document retrieval performance.

As previously mentioned, we apply Domain-Adaptive Pretraining (DAPT) using a corpus of mathematical papers. However, these are not just raw papers in LaTeX format. When writing a paper in LaTeX, many elements are unrelated to the actual content or topic. For instance, commands such as library imports or structural commands like `\textbf`, `\indent`, `\texttt` or `\table` serve purely visual or organizational purposes and do not carry meaningful contextual information.

With this in mind, we performed specific preprocessing on the mathematical papers. Our goal was to select only those sections that combine narrative text with mathematical equations, since we do not want the model to learn exclusively from mathematical formulas. Instead, we aim for the model to learn the relationship between natural language and formal notation—how equations are typically embedded within or explained by surrounding text.

Selected Sections: From the full documents, we extracted sections such as: `introduction`, `methodology`, `conclusion`, `model`, `results`, `proves`, `proof`, `models`.

Where we make sure that they include semantically rich environments such as `definition`, `theorem`, `lemma`, `corollary`, and `proof`.

All selected content was then merged and underwent the same preprocessing described in **Section 3.b**, to maintain coherence in the structure of the data used to train the model.

And once we have the data prepared, we continue the pretraining process on domain-specific data using **Masked Language Modeling (MLM)**. The goal is for ModernBERT to **adapt its vocabulary and internal representations** to better capture the style, terminology, and structural characteristics of the **mathematical domain**.

To do this efficiently, we adopt a fixed-length input strategy (required by the model's constant-shape tensor interface). Although ModernBERT can process sequences up to 8,192 tokens in length, we limit our input window to 1,024 tokens to manage RAM usage. Any segment exceeding this limit is split into consecutive windows of exactly 1,024 tokens with a 126-token overlap (stride) to retain contextual continuity. For shorter segments, we apply padding using the [PAD] tokens. This windowing strategy is applied consistently across the entire corpus. We then prepare inputs for MLM by using a data collator randomly replace 15 % of tokens with [MASK], generating input–label pairs. Training is carried out over three epochs on an NVIDIA A100 GPU, taking approximately four hours. And we upload the model into Hugging Face for facilitating later use.

While the commonly recommended dataset size for models of this scale is around 1GB, we are constrained by limited computational resources and available GPU memory, so we use approximately 500MB of mathematical text's data. As a result, the DAPT phase is performed on half the recommended data volume.

This limitation suggests that the performance of the ModernBERT + DAPT model could potentially be further improved with access to larger datasets and more training epochs.

(c) Fine-tuning objective: Multiple Negatives Ranking Loss

To fine-tune our sentence embedding models for mathematical question–statement retrieval, we use **MultipleNegativesRankingLoss (MNRL)**, a contrastive loss designed in the sentence-transformers Python library (Reimers, 2020) for efficient and scalable retrieval training. This loss is widely used in `sentence-transformers` (Reimers and Gurevych, 2019) and related models thanks to its effectiveness, and specially due to its ability to leverage in-batch negatives without requiring explicit negative sampling.

Given a batch of *(query, positive passage)* pairs (ie., a mathematical question and the statement that answers it), MNRL treats all other positives in the batch as negatives for a given query. This strategy enables strong supervision from just positive pairs, as long as the batch is sufficiently diverse.

The goal of this loss function is for the model to learn embeddings such that each query is closer to its corresponding positive passage, and further away from all other (non-matching) passages. See the figure below for a visual representation.

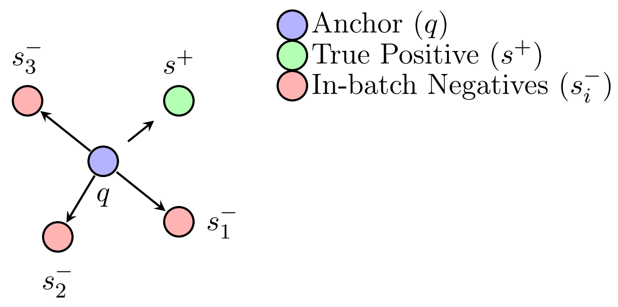


Figure 10: Visual representation of the loss function.

(i) Mathematical Formulation

Assume a batch of N query–positive pairs:

$$\{(q_1, p_1), (q_2, p_2), \ldots, (q_N, p_N)\}$$

We encode the queries and passages into L2-normalized embeddings:

$$q_i, p_i \in \mathbb{R}^d, \quad \|q_i\| = \|p_i\| = 1$$

We then compute a *similarity matrix* $S \in \mathbb{R}^{N \times N}$ using cosine similarity, scaled by a factor $\alpha = 20$:

$$S_{i,j} = \alpha \cdot \cos(q_i, p_j)$$

Below can be found a table with an example of what the cosine similarity matrix looks like.

	pos$_0$	**pos**$_1$	**pos**$_2$	**pos**$_3$
query$_0$	**14.8**	12.3	11.2	10.4
query$_1$	11.0	**15.1**	13.6	12.0
query$_2$	12.8	13.3	**16.2**	11.5
query$_3$	10.5	11.2	12.1	**14.9**

Table 6: Scaled cosine similarities between queries and positive passages.

The correct passage for q_i is p_i, so we use the cross-entropy loss over each row:

$$\mathcal{L}_i = -\log\left(\frac{e^{S_{i,i}}}{\sum_{j=1}^{N} e^{S_{i,j}}}\right)$$

The total loss is:

$$\mathcal{L}_{\text{MNRL}} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_i$$

This formulation encourages the diagonal scores $S_{i,i}$ (matching pairs) to be higher than all off-diagonal scores $S_{i,j}$ where $j \neq i$.

This loss function is very efficient for our purpose, as it requires only a similarity matrix and a softmax operation, and is "retrieval-aligned", as it optimizes the embedding space for cosine similarity, which is also used in RAG during inference.

In our setting, questions and statements are embedded using a shared encoder, and the loss pulls matching pairs closer while pushing others apart. This aligns directly with our retrieval objective: given a query, retrieve the most semantically appropriate passage from the corpus.

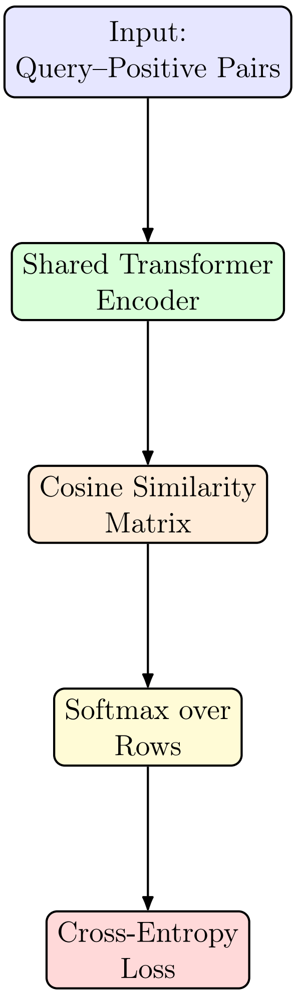


Figure 11: Training flow for MultipleNegativesRankingLoss

(d) Evaluation Metrics

To determine whether fine-tuning improves the retrieval performance of our embedding model, `modernBERT`, we rely on a set of standard evaluation metrics widely used in retrieval-augmented generation (RAG) tasks. These metrics allow us to quantify changes in retrieval quality across different experimental setups.

At a high level, our evaluation process proceeds as follows: for each query in the test set—unseen during training—the fine-tuned embedding model encodes the query and retrieves the top-k most similar statements from the corpus based on embedding cosine similarity. Relevance is defined at the paper level: all statements that appear in the same paper as the query statement are considered relevant.

This definition aligns with the ultimate goal of our RAG system, which is to retrieve entire papers that are useful for answering mathematically oriented questions. In practice, we are not only interested in recovering the exact statement corresponding to the query, but also in identifying other semantically related results within the same paper.

In mathematical texts, it is often the case where multiple theorems in a single work build on a common theme, thus their embeddings hopefully grow closer after we fine tune the model. However, some statements may differ substantially in content or purpose, and we do not expect all top-ranked results to belong exclusively to the target paper—especially in the top-10.

The following metrics are used to assess performance: Accuracy@k, Precision@k, Recall@k, Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain (nDCG@k). These are computed for each query individually and then averaged across the test set to yield global performance measures.

The retrieval pipeline to evaluate our results is the following:

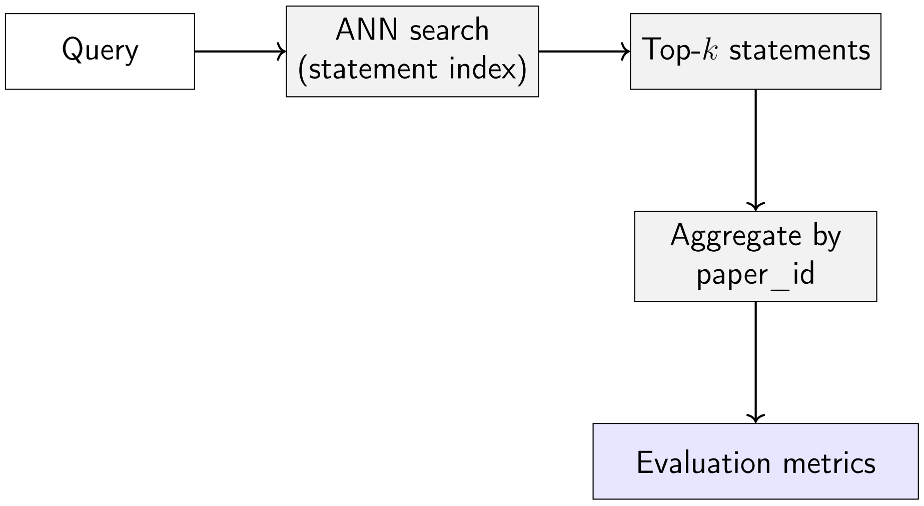


Figure 12: Retrieval pipeline used for evaluation: the query retrieves the top-k statements using an approximate nearest neighbor (ANN) index, which are then aggregated by paper ID to compute the evaluation metrics.

More details on the retrieval metrics can be found below:

- **Accuracy@k:** Assesses the percentage of queries for which at least one relevant document is retrieved in the top-k results. We will use k values of 1, 3 and 5.
- **Precision@k:** Measures the proportion of relevant theorems among the top-k retrieved results. It reflects how accurate the top-k suggestions are, without considering their rank order.

$$\text{Precision@}k = \frac{1}{k}\sum_{i=1}^{k} \mathbb{I}(r_i \in \text{Rel}(q))$$

where r_i is the i-th retrieved item, $\text{Rel}(q)$ is the set of relevant items for query q, and $\mathbb{I}(\cdot)$ is the indicator function.

- **Recall@k:** Assesses the proportion of all relevant theorems that are retrieved within the top-k results.

$$\text{Recall@}k = \frac{|\{r_1, r_2, \ldots, r_k\} \cap \text{Rel}(q)|}{|\text{Rel}(q)|}$$

  For low values of k (eg. $k = 5$), this metric can be misleading. Even if all 5 retrieved statements are relevant, our relevant paper might have 20 relevant statements, so $\text{Recall@}k = 0.25$. To address this, recall with higher values of k are considered as more relevant. Additionally, this metric can be complemented by precision and nDCG@k, which reward highly ranked results, regardless of the total amount of potentially relevant statements.

- **Normalized Discounted Cumulative Gain (nDCG@k):** This metric captures both the relevance and ranking position of retrieved theorems by applying a logarithmic discount to lower-ranked items.

$$\text{DCG@}k = \sum_{i=1}^{k} \frac{2^{\text{rel}_i} - 1}{\log_2(i+1)} \qquad \text{nDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k}$$

  where rel_i is the graded relevance of the item at rank i, and IDCG@k is the ideal DCG (i.e., the DCG of the optimal ranking).

  nDCG@k discounts lower ranks and automatically normalizes, making it robust across different numbers of relevant items. For example, a paper with 25 theorems gets nDCG@10 = 1 if the most relevant ones appear near the top — regardless of how many total it has.

- **Mean Reciprocal Rank (MRR):** Calculates the inverse of the rank at which the first relevant result appears, averaged over all queries. This metric emphasizes the "early retrieval" of relevant items, rewarding models that rank correct results near the top. Therefore, it is a good metric to complement other like Accuracy, Recall@k or Precision@k, which measure how many relevant items are retrieved but do not account for where those items appear in the ranking.

$$\text{MRR} = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{\text{rank}_q}$$

  where rank_q is the rank position of the first relevant item for query q.

5 Results and Discussion

In this section, we present the results achieved by each of our models individually, followed by an overall comparison and a detailed discussion of how we interpret the evolution of retrieval metrics after fine-tuning.

The main model of interest is `modernBERT base`, as it represents the most direct and unadapted version of our approach: fine-tuning a general-purpose pretrained embedding model to assess how much performance can be gained for domain-specific retrieval through fine tuning adaptation alone.

We then evaluate the `DAPT modernBERT` model, which allows us to explore whether additional domain-adaptive pretraining on mathematical texts can help the model better handle domain-specific terminology -which is unfamiliar with- and improve retrieval performance.

Finally, we present the results for `nomicAI modernBERT`, a model already optimized for sentence embeddings, to assess whether starting from a checkpoint tailored for semantic similarity leads to even greater improvements when fine-tuned on our specific retrieval task.

All models were fine tuned with a GPU L4 of 22.5 GB RAM. Fine tuning each model took around 140 minutes, which accounts for a total GPU usage of approximately 7 hours.

Recall that, in our test dataset, for each question we have the relevant statement that could answer it, and we do not only consider that statement as correct, but also all other statements from the same paper.

(a) ModernBERT base

Metric	Base Value	Fine-Tuned Value	% Improvement	Unseen Test
Accuracy@1	0.1777	0.8655	387.14%	0.8190
Accuracy@3	0.2147	0.9150	326.17%	0.8619
Accuracy@5	0.2350	0.9282	295.00%	0.8857
Recall@30	0.0260	0.1904	633.07%	0.3190
Precision@3	0.1064	0.6053	469.09%	0.6063
Precision@5	0.0793	0.4863	513.00%	0.4924
Precision@10	0.0511	0.3406	566.00%	0.3434
nDCG@10	0.0733	0.4428	504.00%	0.4316
MRR@10	0.2016	0.8932	342.94%	0.8483

Table 7: Retrieval metrics for modernBERT base: Base vs. Fine-Tuned vs. Unseen Test.

The results show impressive retrieval performance gain, showing that fine tuning not only significantly improves the metrics, but actually reaches near "Deployment Ready" results. Looking at Accuracy we can observe how the fine tuned model retrieves a statement from the correct paper in the very first position around 87% of the time. If the RAG system only parsed the first three results, the correct paper would be retrieved for above 91% of queries.

A good metric to compliment the different values of Accuracy is MRR. A fine tuned value of 89% means that , on average, the first relevant result for each query appears in position 1.12. (value obtained by inverting 0.89. This tells us that, across all queries, when there is at least one retrieved statement from the correct paper, it's almost always in the very top ranks (often rank 1, sometimes rank 2).

Recall@30 tells us that, on average, we retrieve around 20% of all possible correct statements in the top 30. This metric yields the lowest absolute value, yet shows the greatest relative improvement. There are several reasons for why this value could be lower than the others.

The first and possibly most relevant reason is that, despite many statements in a single paper are linked to each other and build up on a common theme, it is common for papers (especially those with many statements) include theorems and lemmas that cover different topics, with very little semantic overlap across the paper. This makes it much harder for the model to retrieve as relevant all statements in each paper.

Below we provide an example:

Example: Semantic Diversity Within a Single Paper

Statement 1:
An absolute connectivity lattice is a connectivity lattice in which the connectivity adjunction is an isomorphism.

Question:
What does it mean for the connectivity adjunction to be an isomorphism in the context of absolute connectivity lattices?

Statement 2:
Call a subposet Γ' *of a chainmail* Γ *a subchainmail of* Γ *when it is closed in* Γ *under joins of mail-connected subsets of the subposet* Γ'.

Question:
What is the relationship between a subchainmail and a chainmail in terms of closure under joins of mail-connected subsets?

If a single paper has ten theorems on five distinct subtopics, the model might only retrieve the two or three it recognizes as 'close enough' to a given query. Hence, out of ten relevant statements, at most three might land in the top 30—capping Recall@30 at 0.30 for that paper.

Another supporting argument comes from the imbalance present in the number of statements per paper. The k value for this metric was explicitly chosen so that most papers would be able to have full coverage in it. However, there is still a proportion of papers that have over 30 statements. This means that, for a paper with 60 statements, even if all 30 retrieved documents were actually relevant, we would reach a Recall value of 0.5. Despite this not being the most influential factor, many papers with large amounts of statements in it are more prone to having different topics covered in it, exacerbating the mentioned issue.

In fact, the *UnseenTest* supports this latter argument. Recall that this test set is composed of 210 statements coming from 15 different papers that have exactly 14 statements each, none of which have been seen by the model during training. The more stable number of statements per paper results in a higher Recall value.

nDCG@10 evaluates not only whether the retrieved statements are relevant, but also how highly they are ranked within the top 10. Unlike Precision@10, which simply measures the proportion of relevant items among the top 10, this metric assigns higher importance to relevant statements that appear earlier in the list. In this way, it provides a more nuanced view of retrieval quality, rewarding models that place relevant results near the top.

For instance, in our fine-tuned ModernBERT model, we observe a Precision@10 of 0.3406 and an nDCG@10 of 0.4428. The relatively high nDCG compared to precision indicates that not only are 3 to 4 relevant statements retrieved in the top 10, but many of them also appear in the very top positions, further confirming the MRR value. This reinforces the conclusion that fine-tuning not only improves the number of correct statements retrieved, but also improves their relative ranking, which is critical in practical applications where users are most likely to read the first few results.

The *Unseen Test* shows slightly worse results in all metrics (except for the already commented Recall), which suggests that our model has learned to generalize for new papers, but it would benefit from being further trained on when new papers are added.

(b) ModernBERT with DAPT

Metric	**Base Model**	**DAPT Model**	**% Improvement**
Accuracy@1	0.1777	0.1915	7.77%
Accuracy@3	0.2147	0.2304	7.31%
Accuracy@5	0.2350	0.2488	5.87%
Recall@30	0.0260	0.0287	10.38%
Precision@3	0.1064	0.1174	10.34%
Precision@5	0.0793	0.0877	10.59%
Precision@10	0.0511	0.0571	11.75%
nDCG@10	0.0733	0.0808	10.24%
MRR@10	0.2016	0.2167	7.50%

Table 8: Comparison of retrieval performance between the Base Model and the DAPT Model both without FT, including relative improvements.

From Table 8, we observe the impact of Domain-Adaptive Pretraining (DAPT) compared to the base ModernBERT model without DAPT. All retrieval metrics show **consistent improvements**, ranging from approximately 5% to 12%. These results indicate that DAPT was effective: the model successfully adapted its internal representations, enabling it to better capture the style, terminology, and structural characteristics of mathematical language—ultimately enhancing retrieval performance.

It is worth noting that DAPT was performed using only 500MB of domain-specific data. While larger-scale pretraining (typically 1GB or more) is recommended for models of this size, our data volume was constrained by computational limitations. Nevertheless, the performance gains observed suggest that even greater improvements could be achieved with access to more training data.

From Table 9, we observe the impact of fine-tuning (FT) on the model, which leads to improved performance across all evaluation metrics. It is important to emphasize that applying DAPT prior to fine-tuning is fully compatible: the parameter updates

Metric	Base Value	Fine-Tuned Value	Improvement	Unseen Test
Accuracy@1	0.1915	0.8686	353.68%	0.8238
Accuracy@3	0.2304	0.9186	298.74%	0.8667
Accuracy@5	0.2488	0.9330	274.84%	0.8952
Recall@30	0.0287	0.1944	576.91%	0.3163
Precision@3	0.1174	0.6121	421.54%	0.6111
Precision@5	0.0877	0.4932	461.94%	0.4886
Precision@10	0.0571	0.3476	508.74%	0.3454
nDCG@10	0.0808	0.4491	456.01%	0.4251
MRR@10	0.2167	0.8966	313.79%	0.8535

Table 9: Retrieval metrics for the DAPT ModernBERT model: Base vs. Fine-Tuned vs. Unseen Test.

introduced during the DAPT phase do not interfere with the fine-tuning process or degrade its effectiveness. On the contrary, they serve as a strong initialization that enhances downstream retrieval performance.

The results obtained after fine-tuning the DAPT model are comparable to those of the fine-tuned base model in terms of relative metric improvements. Notably, we again observe the most significant gain in **Recall@30**, where nearly 20% of the relevant theorems from a paper are retrieved within the top 30 retrieved statements—an encouraging result for information retrieval in mathematical domains.

While accuracy-based metrics show strong performance, we acknowledge that the current model may still fall short of being “deployment-ready” in high-stakes applications. In our case, the primary objective is to retrieve the source paper of a given theorem or to surface related prior work. Retrieval errors in this context do not pose safety or financial risks; they result primarily in time loss, as users would realize the error upon reading the retrieved paper.

Therefore, our system does not aim for near-perfect values (e.g., 99.9% Accuracy@1 or Precision), as might be expected in mission-critical domains. However, we do aim for strong and reliable performance, targeting around **95% Accuracy@1** and competitive scores in metrics such as Precision and MRR, which would make the system practically useful in academic or exploratory research contexts.

(c) Nomic AI ModernBERT

This model shows results that drift further from `modernBERT base` than those from the DAPT model, showing the impact of optimizing an encoder model for sentence embeddings. The baseline values already interestingly demonstrate that fine tuning on a general topic corpus significantly increases retrieval metrics.

The relative improvements are largely smaller than those achieved in previous models, as these represent the improvement over the second part of a two-step fine tuning. All finetuned metrics grow in parallel to the base model, achieving slightly better results in each, reaching the overall highest results. Values like a 95% accuracy show that this fine tuning takes the baseline NomicAI model from "very good", to nearly perfect.

Metric	Base Value	Fine-Tuned Value	Improvement	Unseen Test
Accuracy@1	0.7715	0.9129	18.34%	0.8905
Accuracy@3	0.8379	0.9455	12.88%	0.9476
Accuracy@5	0.8603	0.9544	10.93%	0.9619
Recall@30	0.1857	0.2204	18.67%	0.3840
Precision@3	0.5536	0.6632	19.80%	0.6825
Precision@5	0.4522	0.5432	20.13%	0.5686
Precision@10	0.2751	0.3731	35.62%	0.3812
nDCG@10	0.4132	0.4952	19.85%	0.4972
MRR@10	0.8096	0.9311	15.01%	0.9210

Table 10: Retrieval metrics for the nomic-ai/ModernBERT model: Base vs. Fine-Tuned vs. Unseen Test.

A 66% value in Precision@3 shows that, on average, 2 out of the top 3 retrieved statements belong to the correct paper. An MRR value above 93% further confirms that the first correctly retrieved statement is either first or second time in the vast majority of cases.

Unseen Test's results are more interesting in this model, as the gap between the test with seen and unseen papers is much tighter, which provides evidence that the further pre-training provides a more robust performance for new papers in the RAG system.

(d) Discussion

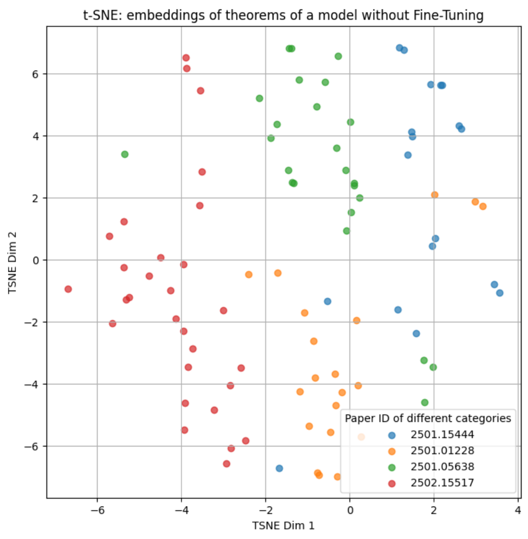
Metric	Baseline Model	modernBERT Base	DAPT	NomicAI
Accuracy@1	0.1777	0.8655	0.8686	0.9129
Accuracy@3	0.2147	0.9150	0.9186	0.9455
Accuracy@5	0.2350	0.9282	0.9330	0.9544
Recall@30	0.0260	0.1904	0.1944	0.2204
Precision@3	0.1064	0.6053	0.6121	0.6632
Precision@5	0.0793	0.4863	0.4932	0.5432
Precision@10	0.0511	0.3406	0.3476	0.3731
nDCG@10	0.0733	0.4428	0.4491	0.4952
MRR@10	0.2016	0.8932	0.8966	0.9311

Table 11: Overall comparison of retrieval metrics across all models. The Baseline Model is ModernBERT base without fine-tuning.

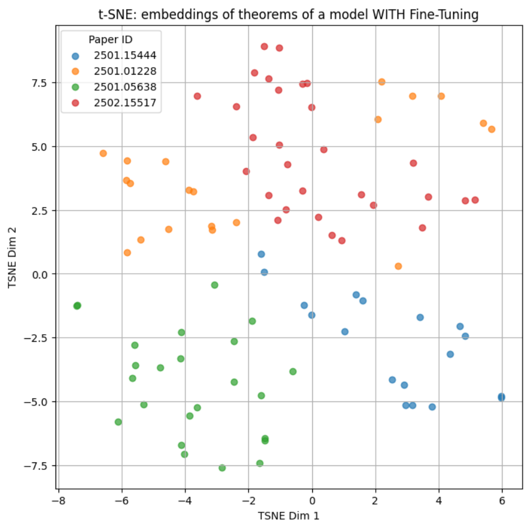
As can be observed above, Table 11 shows the overall comparison of the main retrieval metrics across our Baseline Model (`modernBERT Base`) and all fine tuned models. The raw fine tuned version of `modernBERT Base` makes the largest relative jump over the baseline, whilst the stable results improvements from DAPT and NomicAI show how additional techniques can further boost the model's performance.

Recall seems to have hit a low ceiling- probably due to reasons unrelated to the models-, which makes clear that even the top-performing model leaves room for improvement when the goal is to retrieve all relevant passages. Other metrics such as Precision and nDCG emphasize how NomicAI's sentence embedding pretraining helps not only retrieve more relevant documents, but also place them at the very top.

In conclusion, the assessed models help clarify that fine tuning on a general embedding model significantly boosts retrieval performance for domain-specific tasks, at little economic and environmental cost. Furthermore, we leave the door open to study how much DAPT implementation can further improve performance at the cost of a few more GPU hours for an additional unsupervised training stage.



(a) Embeddings of 4 mathematical papers with ModernBERT with DAPT



(b) Embeddings of 4 mathematical papers with a fine-tuning ModernBERT with DAPT

Figure 13: Comparative of the embeddings of theorems of 4 papers from different categories before and after Fine-tuning

These images provide a more visual understanding of how fine-tuning (FT) affects the embeddings of theorems from papers in different mathematical categories. Through this example, we aim to highlight a few interesting observations and gain deeper insight into how the loss function influences embedding structure from a visual perspective.

We selected randomly four papers, each belonging to a different arXiv category. For each paper, we embed the theorems using the model both before and after fine-tuning. Once the embeddings were computed, we applied **t-SNE** (t-distributed Stochastic Neighbor Embedding), a non-linear dimensionality reduction technique commonly used to visualize high-dimensional data in two dimensions.

While a detailed explanation of t-SNE is beyond the scope of this project, it is important to note that we use it solely to reduce the embedding dimensionality (from 768 to 2) for visualization purposes.

In the table 12 we can see the categories of the papers of the analysis and a short description of this categories, that can helps us understand more the distribution of the embeddings.

In the initial embedding space (prior to fine-tuning), we observe a certain degree of clustering among theorems from the same paper. While the clustering is not particularly strong in all cases, it aligns with our initial intuition: theorems from the same

Paper	Categories	Description
`2501.15444`	`[math.CO]`	Focuses on combinatorics, including counting problems, graph theory, and discrete structures.
`2501.01228`	`[math.OC]`	Deals with optimization and control theory, especially finding optimal solutions under constraints in dynamic systems.
`2502.15517`	`[math.AG, math.RT]`	Combines algebraic geometry (studying shapes defined by polynomial equations) and representation theory (studying group actions via matrices).
`2501.05638`	`[cs.CC, cs.DM, cs.DS, math.CO]`	A multidisciplinary blend of computational complexity, discrete mathematics, data structures, algorithms, and combinatorics.

Table 12: Overview of selected arXiv category groups and their thematic focus.

source document tend to be embedded more closely together, while those from different papers—especially from distinct mathematical categories—should ideally appear more separated.

However, some deviations from this expectation are noteworthy. For example, theorems from paper `2501.15444` appear relatively dispersed, even though this paper focuses on a single category. In contrast, paper `2501.05638`, which covers multiple topics (primarily computer science, but also including combinatorics), exhibits a tighter cluster. This cluster includes a small group of three theorems at the bottom of the embedding plot, which may correspond to the combinatorics-related content. These mixed-topic overlaps suggest that topic boundaries in the embedding space are influenced not only by document-level labels but also by the semantic interplay between closely related subfields.

It is important to note that this **analysis is purely observational**, based on visual inspection of the embedding plots. Our aim is not to draw generalized conclusions, but rather to explore specific cases and understand how individual theorems are positioned within the representation space.

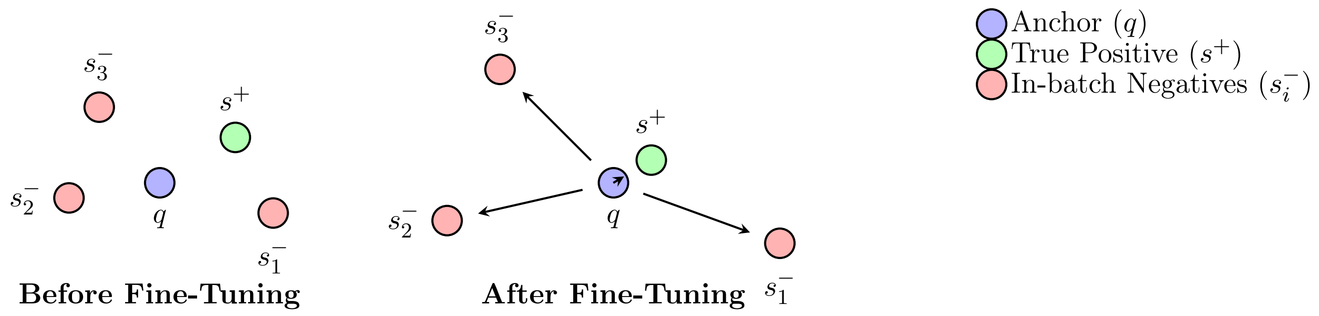


Figure 14: Embedding space before and after fine-tuning. After contrastive learning, positive examples are pulled closer to the query, and negatives are pushed farther away. Reminder: We considered in-batch negatives theorems from another Paper

After applying fine-tuning, we observe that inter-paper distances have increased, and clustering becomes more pronounced. This outcome is consistent with the training objective: the contrastive loss function used during fine-tuning explicitly encourages positive pairs (queries and their corresponding passages) to be pulled closer together, while pushing negative pairs (non-matching passages) farther apart.

This push-pull mechanism is mediated through the query embedding. Positive passages are drawn into tighter alignment with their associated query, while negative passages are repelled. The result is a representation space in which queries tend to cluster around their correct passages and remain distant from incorrect ones—**collectively shaping a more structured and semantically meaningful embedding space**.

Important note: This clustering effect would likely be more pronounced if the theorems analyzed in this section had appeared in the same training batch. In that case, theorems from different papers would have been treated explicitly as negative passages, thereby reinforcing inter-document separation during training.

Even though the four papers analyzed may not have appeared in the same training batches, the observed clustering can be explained by the indirect effects of contrastive fine-tuning across the entire dataset. During training, the **model learns generalizable patterns** about which types of content tend to co-occur or correspond to similar queries. As positive pairs (query–passage) are repeatedly reinforced and negatives are pushed away, the model implicitly learns to organize the embedding space around semantic similarity.

This results in local structuring, where theorems that share similar linguistic or mathematical properties—even across different batches—gradually cluster together. In effect, the contrastive loss induces a global geometry that reflects recurring patterns in the training data. Even if the specific theorems from these four papers never directly interacted during training, their relationships to other similar or contrasting examples throughout the corpus help shape their eventual positions in the space.

6 Conclusions and Recommendations

(a) Conclusions

In this work, we enhance the retrieval capabilities of a base embedding model using techniques adapted to our limited computational resources. These improvements can be directly applied to the retrieval component of a Retrieval-Augmented Generation (RAG) system—ultimately leading to higher-quality answers.

We show that fine-tuning not only benefits a pretrained ModernBERT model, but also yields gains when applied after domain-adaptive pretraining (DAPT). Moreover, fine-tuning further improves a ModernBERT variant that was already fine-tuned as a general-purpose sentence encoder on a different domain. In this case, additional fine-tuning helps the model acquire the mathematical vocabulary and structure specific to our target domain.

Across all of our experiments, fine-tuning produced consistently strong results at relatively low computational cost.

Furthermore, our results show that for improving retrieval, it is more effective to start from a model that already knows how to generate sentence embeddings for question–answer pairs (e.g., a ModernBERT model fine-tuned on a general sentence embedding task) and then adapt it to the mathematical domain. This way, the model only needs to learn domain-specific vocabulary. By contrast, beginning with a model that has already been exposed to DAPT on mathematics but has not yet been trained to produce sentence embeddings forces it to learn the embedding task, resulting in lower retrieval performance.

A helpful analogy: is not the same to teach mathematics to someone who already knows how to teach, than to teach someone who knows mathematics how to teach it.

(b) Further Study

While this study provides promising results, it also opens several avenues for further research and improvement. One natural extension would be to test the system on a much broader corpus—not just a few months of papers, but possibly the last ten years of mathematical publications. This would allow for evaluating the model's scalability and robustness across time.

Another important direction involves developing methods to improve generalization to unseen data. In realistic scenarios with continuously growing datasets, it is not computationally feasible to fine-tune the model every time a new paper is published. Daily updates would impose a significant computational burden. Therefore, identifying techniques that increase robustness to domain shifts—reducing performance degradation on unseen data—could be critical. For instance, fine-tuning could be triggered only when performance on new data drops below a certain threshold, using accumulated unseen data.

Moreover, when scaling to massive datasets, full fine-tuning becomes even less practical. In such cases, it would be valuable to explore parameter-efficient fine-tuning methods such as LoRA, QLoRA, or DoRA. These techniques introduce small, structured adjustments to a model's parameters, allowing for efficient training without updating the entire model. In some settings, they have even outperformed full fine-tuning by mitigating issues such as catastrophic forgetting.

Studying the presence and impact of catastrophic forgetting in our current model would also be an interesting research direction, as it may affect long-term retention of previously learned knowledge during incremental updates.

Finally, the most important future goal is to further improve retrieval performance to the point that the system becomes *Deployment-Ready*. Achieving this would make it feasible to integrate the model into real-world applications such as Retrieval-Augmented Generation pipelines or platforms like arXiv for semantic search and recommendation.

References

Brown, T., Mann, B., Ryder, N., Subbiah, M., et al. (2020). Language models are few-shot learners. *https://arxiv.org/pdf/2005.14165.*

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805.*

Gao, L., Jiang, Z., Yin, Y., Yuan, K., Yan, Z., & Tang, Z. (2017). Preliminary exploration of formula embedding for mathematical information retrieval: Can mathematical formulae be embedded like a natural language? *arXiv preprint arXiv:1707.05154*. https://arxiv.org/pdf/1707.05154

Gururangan, S., Marasovic, A., Swayamdipta, S., et al. (2020). Don't stop pretraining: Adapt language models to domains and tasks. *ACL*. https://arxiv.org/abs/2004.10964

Horowitz, L., & Hathaway, R. (2024). Fine-tuning berts for definition extraction from mathematical text. https://arxiv.org/abs/2406.13827

Karpukhin, V., Oguz, B., Min, S., et al. (2020). Dense passage retrieval for open-domain question answering. *EMNLP*, 6769–6781. https://aclanthology.org/2020.emnlp-main.550/

Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. https://arxiv.org/abs/2005.11401

Mansouri, B., Rohatgi, S., Oard, D. W., Wu, J., Giles, C. L., & Zanibbi, R. (2019). Tangent-cft: An embedding model for mathematical formulas. https://pure.psu.edu/en/publications/tangent-cft-an-embedding-model-for-mathematical-formulas/

Peng, S., Yuan, K., Gao, L., & Tang, Z. (2021). Mathbert: A pre-trained model for mathematical formula understanding. *arXiv preprint arXiv:2105.00377*. https://arxiv.org/abs/2105.00377

Raffel, C., Shazeer, N., Roberts, A., et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research 21*, 1–67.

Reimers, N. (2020). Sentence-transformers: State-of-the-art sentence and image embeddings.

Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *EMNLP-IJCNLP*. https://arxiv.org/pdf/1908.10084

Shen, J. T., Yamashita, M., Prihar, E., Heffernan, N., Wu, X., Graff, B., & Lee, D. (2021). MathBERT: A pre-trained language model for general nlp tasks in mathematics education. *arXiv preprint arXiv:2106.07340*. https://arxiv.org/abs/2106.07340

Shuster, K., Smith, Poff, S., Chen, M., Kiela, D., & Weston, J. (2021). Retrieval augmentation reduces hallucination in conversation. *https://arxiv.org/abs/2104.07567.*

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 5998–6008. https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Warner, B., Chaffin, A., Clavié, B., Weller, O., Hallström, O., Taghadouini, S., Gallagher, A., Biswas, R., Ladhak, F., Aarsen, T., Cooper, N., Adams, G., Howard, J., & Poli, I. (2024). Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. https://arxiv.org/html/2412.13663v2

7 Appendix

#	WordPiece	Comment
0	[CLS]	Start-of-sequence token
1	let	Full token from vocabulary
2	$	Punctuation sign
3	n	Single letter(known
4	$	Dollar sign
5	be	Full token from vocabulary
6	an	Full token from vocabulary
7	integer	Full token from vocabulary
8	and	Full token from vocabulary
9	let	Repeated word
10	$	Dollar sign
11	\	Backslash, not part of a known token
12	f	First character of unknown command
13	##ra	WordPiece subword
14	##c	WordPiece subword (completes “frac”)
15	{	Left brace token
16	\	Backslash
17	log	Token from vocabulary
18	^	Caret symbol (superscript indicator)
19	{	Left brace
20	15	Number token
21	}	Right brace
22	n	Variable token
23	}	Right brace
24	/	Slash token (fraction line)
25	n	Variable token
26	$	Dollar sign
27	[UNK]	Unknown token (e.g., Unicode symbol)
28	p	Variable token
29	[UNK]	Another unknown (likely again)
30	[UNK]	Unknown token (e.g., Greek epsilon)
31	$	Dollar sign
32	and	Full token from vocabulary
33	$	Dollar sign
34	q	Variable token
35	=	Equals sign
36	\	Backslash starting command
37	s	Subword (first part of "sqrt")
38	##q	Subword
39	##rt	Subword (completes "sqrt")
40	{	Left brace
41	\	Backslash
42	f	Subword of “frac”
43	##ra	Subword
44	##c	Subword
45	{	Left brace
46	p	Variable token
47	}	Right brace
48	/	Slash
49	n	Variable token
50	\	Backslash
51	log	Token from vocabulary
52	^	Caret
53	8	Number
54	n	Variable token
55	}	Right brace
56	}	Right brace
57	$	Final dollar sign
58	.	Period (sentence end)
59	[SEP]	End-of-sequence token

Table 13: Example of WordPiece tokenization applied to a LaTeX-style mathematical expression.

Model	**Acc@1**	**@3**	**@5**	**@10**	**Prec@1**	**@3**	**@5**	**@10**	**Recall@1**	**@3**	**@5**	**@10**	**NDCG@10**	**MRR@10**	**MAP@100**
ModernBERT BASE (no FT)	0.1777	0.2147	0.2350	0.2597	0.1777	0.1064	0.0793	0.0511	0.0079	0.0130	0.0157	0.0192	0.0733	0.2016	0.0174
ModernBERT + DAPT (no FT)	0.1915	0.2304	0.2488	0.2779	0.1915	0.1174	0.0877	0.0571	0.0083	0.0142	0.0170	0.0209	0.0808	0.2167	0.0194
ModernBERT + DAPT (with FT)	0.8680	0.9183	0.9325	0.9496	0.8680	0.6119	0.4935	0.3476	0.0419	0.0832	0.1074	0.1421	0.4493	0.8964	0.1638
nomicai/ModernBERT (no FT)	0.7715	0.8379	0.8603	0.8846	0.7715	0.5536	0.4522	0.3255	0.0368	0.0745	0.0976	0.1318	0.4132	0.8096	0.1557
nomicai/ModernBERT (with FT)	0.9128	0.9455	0.9544	0.9657	0.9128	0.6628	0.5434	0.3914	0.0442	0.0905	0.1184	0.1593	0.4952	0.9311	0.1922
MathBert (no FT)	0.4210	0.4873	0.5171	0.5601	0.4210	0.2853	0.2264	0.1566	0.0184	0.0352	0.0457	0.059	0.2060	0.4630	0.0600

Table 14: Retrieval performance of various ModernBERT configurations.

Metric	**MathBERT**	**MathBERT + FT**	**% Improvement**
Accuracy@1	0.421	0.787	+86.9%
Accuracy@3	0.487	0.857	+76.1%
Accuracy@5	0.517	0.880	+70.1%
Precision@3	0.285	0.545	+91.2%
Precision@5	0.226	0.437	+93.3%
Recall@3	0.035	0.073	+108.5%
Recall@5	0.045	0.094	+109.7%
NDCG@10	0.206	0.399	+93.7%
MRR@10	0.463	0.827	+78.6%
MAP@100	0.060	0.142	+135.7%

Table 15: Comparison of MathBERT vs MathBERT + Fine-Tuning (FT) on cosine similarity metrics.