

# LeanExplore

## Semantic Search for Lean 4 Mathematical Libraries

Justin Asher

Axiom Math

September 16, 2025

# Outline

- 1 The Problem
- 2 The Search Algorithm
- 3 Applications
- 4 Conclusion

# The Challenge: Missing Documentation in Formal Libraries

Lean 4 libraries are large in size:

- Mathlib: **112,407** definitions
- Mathlib: **230,392** theorems
- Plus PhysLean, Batteries, Std, etc.

**Critical problem:**

- Existing search interfaces can be hard to navigate
- They lack the ability to search things both formally, informally, by name, etc.
- **Why not use all of the data available to us?**

## Example: Mathlib docs

Try searching for:

*"fundamental theorem of calculus"*

**Result:** Nothing found!

**Actual name:**

```
intervalIntegral.integral_hasDerivAt_of_
tendsto_ae_left
```

# Our Approach

- 1 **Inductively informalize** all Lean top-level declarations
  - Use LLM with dependency context for consistency
  - Generate human-readable descriptions for formal code
- 2 **Create 4 embeddings** for each declaration:
  - **Name** (e.g., `Real.sqrt`)
  - **Source code** (formal Lean definition)
  - **Docstring** (if available)
  - **Informalization** (AI-generated description)
- 3 **Add all embeddings to a unified index** (FAISS + BM25+)
- 4 **Use hybrid search algorithm** combining semantic, lexical, and structural signals

# The Search Algorithm

# Hybrid Search Algorithm Details

## Embedding Model: BAAI bge-base-en-v1.5 (109M parameters)

- Lightweight yet effective for mathematical content
- Embeds formal code, informalizations, names, docstrings

## BM25+ for Lexical Matching:

- Extension of BM25 with proper lower-bounding
- Excellent for keyword matching: distinguishes "groups" vs "rings"
- Handles exact term relevance that embeddings might miss

## PageRank on Dependency Graph:

- Computed on dependencies between declarations
- Identifies foundational/important theorems

## Signal Weighting: $1.0 \times \text{semantic} + 1.0 \times \text{BM25+} + 0.2 \times \text{PageRank}$

(Semantic score = max over embeddings: name, docstring, composite informal statement.)

# Example: Single Lean Construct

## User writes one structure definition:

```
structure Scheme extends LocallyRingedSpace where
  local_affine :
     $\forall x : \text{toLocallyRingedSpace},$ 
     $\exists (U : \text{OpenNhds } x) (R : \text{CommRingCat}),$ 
    Nonempty
    (toLocallyRingedSpace.restrict U .isOpenEmbedding  $\cong$ 
     Spec.toLocallyRingedSpace.obj (op R))
```

## Fragments into multiple constants:

- AlgebraicGeometry.Scheme (inductive type)
- AlgebraicGeometry.Scheme.mk (constructor)
- AlgebraicGeometry.Scheme.local\_affine (the local affine theorem)
- AlgebraicGeometry.Scheme.toLocallyRingedSpace (projection)
- AlgebraicGeometry.Scheme.rec, .noConfusion, etc.

# AI-Generated Informalization: The Prompt

**Goal:** Translate formal Lean 4 statements into clear, concise informal English descriptions suitable for mathematical textbooks.

## Key Instructions:

- Reference only the formal statement and docstring
- Focus on core mathematical concept or assertion
- Write in natural mathematical English
- **No repetition** of Lean syntax, keywords, or declaration names
- Use standard informal mathematical terminology
- Be brief and mathematically rigorous

## Context Provided:

- Primary declaration name and type
- Docstring (if available)
- Formal statement text block
- **Prerequisites context** (informal descriptions of dependencies)



# BM25+ Impact: Example Query "finite morphism schemes"

## With BM25+ (full hybrid):

- 1 A **finite** morphism of schemes is locally of **finite** type.

Score: 1.97

- 2 A **finite** morphism of schemes is an integral morphism.

Score: 1.83

- 3 A **finite** morphism of schemes is proper.

Score: 1.72

- 4 A **finite** morphism of schemes is proper.

Score: 1.69

- 5 A morphism of schemes is **finite** if and only if it is an integral morphism and locally of **finite** type.

Score: 1.66

## Without BM25+ (semantic only):

- 1 A **finite** morphism of schemes is locally of **finite** type.

Score: 1.01

- 2 A morphism of schemes from  $X$  to  $Y$  can be viewed as a rational map...

Score: 0.92

- 3 Given a morphism  $f$  of schemes, we can view it as a partial map...

Score: 0.90

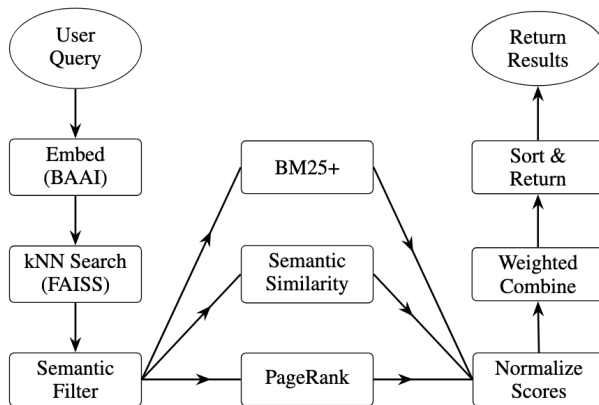
- 4 A **finite** morphism of schemes is an integral morphism.

Score: 0.89

- 5 Given two composable morphisms  $f$  and  $g$  between covers...

Score: 0.82

# Hybrid Search Algorithm Pipeline



# Evaluation Setup

## Experimental Design:

- **300 AI-generated queries** reflecting typical Mathlib search tasks
- Tested against **two leading search engines**: LeanSearch and Moogle
- **Blind evaluation** by Google's Gemini 2.5 Flash model
- **Top 5 results** from each engine presented to evaluator

## Evaluation Criteria:

- LLM evaluator ranks 1st, 2nd, 3rd place for each query
- Based on accuracy and relevance to natural language query
- Ties permitted when results are of equal quality
- **3 independent runs** to ensure reliability

## Competitors:

- **LeanSearch**: E5mistral-7B model (7 billion parameters)
- **Moogle**: Existing semantic search for Mathlib

# Evaluation Results

## Overall Comparison:

Search Engine	1st Place	2nd Place	3rd Place
<b>LeanExplore</b>	<b>55.4%</b>	33.8%	10.8%
LeanSearch	46.3%	34.1%	19.6%
Moogler	12.0%	24.8%	63.2%

## Head-to-Head Comparison:

Comparison	LeanExplore Wins	Competitor Wins	Ties
vs LeanSearch	<b>50.0%</b>	39.4%	10.6%
vs Moogler	<b>79.2%</b>	15.9%	4.9%

Pairwise win rates computed head-to-head; tables show 3-way ranking frequencies.

## Key Results:

- Very rarely came in last (only 10.8% 3rd place)
- Achieved with **lightweight** 109M parameter model
- LeanSearch uses 7B parameter model but still loses overall

- **Lightweight model limitations:** Our 109M parameter model can miss very broad queries (e.g., *"vector space axioms"*)
- **Variable name sensitivity:** BM25+ can be overly sensitive to specific variable names in queries
- **PageRank bias:** Popular theorems may rank highly even when loosely related to queries
- **Limited package coverage:** Currently indexes 6 packages (Batteries, Init, Lean, Mathlib, PhysLean, Std); broader ecosystem coverage needed
- **Static indexing:** No nightly updates; data is becoming outdated as libraries evolve

# Applications

# Making Search Accessible: Multiple Interfaces

## Web Interface:

- Public search at [leanexplore.com](https://leanexplore.com)
- Dependency exploration
- API key management

## Python Library:

- `pip install lean-explore`
- Local & remote backends

## Command Line Interface:

- Direct search commands
- Data management
- AI chat integration

## Model Context Protocol:

- Tools for AI agents
- Standardized integration

## LeanExplore

square root of real number

Packages



Web search entry and initial results.



# Web Interface Demo (2/5)

## Real.sqrt

In: [Mathlib/Data/Real/Sqrt.lean](#)

### LEAN CODE:

```
noncomputable def sqrt (x : ℝ) : ℝ :=  
  NNReal.sqrt (Real.toNNReal x)
```

### DOCSTRING:

The square root of a real number. This returns 0 for negative inputs.

This has notation  $\sqrt{x}$ . Note that  $\sqrt{x^{-1}}$  is parsed as  $\sqrt{(x^{-1})}$ .

### GEMINI 2.0 FLASH:

The square root of a real number  $x$ , denoted by  $\sqrt{x}$ , is defined to be the square root of the nonnegative real number obtained by taking the maximum of  $x$  and 0.

[Find similar](#)

[View Dependencies ▼](#)

Result details view: code, docstring, informal, dependencies.

# Web Interface Demo (3/5)

Hide Dependencies ▲

Init:

- [LE](#)

Mathlib:

- [DFunLike](#)
- [NNReal](#)
- [NNReal.sqrt](#)
- [NNReal.toReal](#)
- [OrderIso](#)
- [PartialOrder](#)
- [Preorder](#)
- [Real](#)
- [Real.toNNReal](#)
- [RelIso.instFunLike](#)

Dependency exploration and navigation context.

# Web Interface Demo (4/5)

Dependency Details

NNReal.sqrt

In: [Mathlib/Data/Real/Sqrt.lean](#)

LEAN CODE:

```
noncomputable def sqrt : ℝ≥0 → ℝ≥0 :=  
  OrderIso.symm <| powOrderIso 2 two_ne_zero
```

[Show all](#)

DOCSTRING:

Square root of a nonnegative real number.

GEMINI 2.0 FLASH:

The square root function is defined as an order isomorphism between the nonnegative real numbers and itself. It is the inverse of the function that maps a nonnegative real number  $x$  to its square  $x^2$ , where this squaring function is viewed as an order isomorphism.

[Find similar](#)

[View Dependencies ▼](#)

## API Keys

Manage your API keys for accessing LeanExplore programmatically.

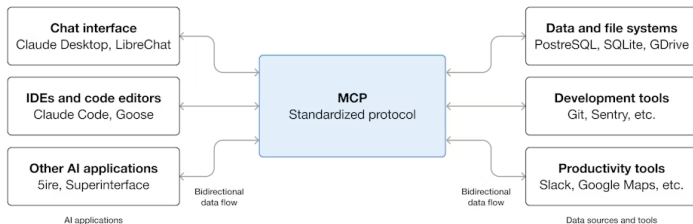
+ Create API Key

LABEL	KEY PREFIX	LAST FOUR	CREATED AT	LAST USED	ACTIONS
Demo	lean_	...KzMs	9/16/2025	Never	<a href="#">Edit</a> <a href="#">Revoke</a>

# What is the Model Context Protocol (MCP)?

A standardized protocol enabling AI systems to connect to external tools and data

- **Universal interface:** Like USB-C for connecting AI applications to external systems
- **Tool integration:** Allows AI agents to use specialized tools during conversations
- **Live data access:** AI can query real-time information from connected services
- **Standardization:** One protocol works across different AI applications (Claude, ChatGPT, etc.)



# Model Context Protocol (MCP): AI Agent Integration

**Standardized interface for AI agents to access LeanExplore**

## Three Core Tools:

- `search`: Semantic search functionality
- `get_by_id`: Retrieve specific statements
- `get_dependencies`: Explore relationships

## Integration Examples:

- Claude Desktop
- Claude Code
- ChatGPT
- etc.

**JA** how a group defined in lean 4

## Thought process

The user is asking about how a group is defined in Lean 4. This is a specific question about the Lean theorem prover and its mathematical library. I should search for information about group definitions in Lean 4 using the lean explore tool to find the actual definitions and structure.

Let me search for "group" in the Lean database to see how groups are defined.

# Claude Desktop with LeanExplore MCP (2/9)

I'll help you understand how a group is defined in Lean 4 by searching through the Lean mathematical library.

L Search

Response

```
{
  "query": "group",
  "packages_applied": [],
  "results": [
    {
      "id": 8438,
      "primary_declaration": {
        "lean_name": "Group"
      }
    }
  ]
}
```



## Thought process

Perfect! Now I have the core definition of Group in Lean 4. Let me also look at what DivInvMonoid is, since Group extends it. I should search for that to give a complete picture.

# Claude Desktop with LeanExplore MCP (4/9)

## L Search

### Request

```
{  
  `limit`: 5,  
  `query`: `DivInvMonoid`  
}
```

### Response

```
{  
  "query": "DivInvMonoid",  
  "packages_applied": [],  
}
```

- Thought process

Now I have a good understanding of how groups are defined in Lean 4. Let me also check what a Monoid is to give the complete picture, since DivInvMonoid extends Monoid.

# Claude Desktop with LeanExplore MCP (6/9)

L Search

```
{  
  "query": "Monoid",  
  "packages_applied": [],  
  "results": [  
    {  
      "id": 8363,  
      "primary_declaration": {  
        "lean_name": "Monoid"  
      },  
      "source_file": "Mathlib/Algebra/Group/Defs.lean",  
      "range_start_line": 561,  
    }  
  ]  
}
```

## Thought process

Great! Now I have a complete picture of how a group is defined in Lean 4. Let me explain the hierarchical structure:

1. Monoid - has associative multiplication, identity element, and natural number powers
2. DivInvMonoid - extends Monoid with inverse operation and division ( $a/b = a * b^{-1}$ ), plus integer powers
3. Group - extends DivInvMonoid with the key axiom that  $a^{-1} * a = 1$

This is a nice hierarchical design that builds up the structure step by step.

# Claude Desktop with LeanExplore MCP (8/9)

## Group Definition in Lean 4

A `Group` in Lean 4 is defined as a class that extends `DivInvMonoid` with one additional axiom:

```
lean

class Group (G : Type u) extends DivInvMonoid G where
  protected inv_mul_cancel : ∀ a : G, a⁻¹ * a = 1
```

This means a group has:

1. **All the structure of a `DivInvMonoid`**, which includes:

- **Monoid structure:** Associative multiplication, identity element `1`, and natural number powers
- **Inverse operation** `⁻¹`
- **Division operation** `/` defined as `a / b = a * b⁻¹`
- **Integer powers** `a ^ z` for `z : ℤ`

2. **The key group axiom:** `inv_mul_cancel : ∀ a : G, a⁻¹ * a = 1`

- This states that for every element `a` in the group, its inverse times itself equals the identity

## Hierarchical Structure

The definition follows this hierarchy:

- **Semigroup** → **Monoid** → **DivInvMonoid** → **Group**

Each level adds more structure:

- **Semigroup**: Associative multiplication
- **Monoid**: Adds identity element and natural number powers
- **DivInvMonoid**: Adds inverse operation, division, and integer powers
- **Group**: Adds the cancellation property  $a^{-1} * a = 1$

## Alternative Constructors

The library also provides `Group.ofLeftAxioms` and `Group.ofRightAxioms` to define group structures with minimal proof obligations, allowing you to prove either left or right cancellation and have the other properties derived automatically.

This design is elegant because it only requires proving one axiom (left inverse cancellation) to get the full group structure, with all other group properties being derived automatically from this and the underlying monoid structure.

- 250-500+ search requests per day
- 100+ registered users

Growing momentum in the Lean community



## Questions & Discussion

**Website:** [leanexplore.com](https://leanexplore.com)  
**Documentation:** [leanexplore.com/docs](https://leanexplore.com/docs)  
**Try it:** `pip install lean-explore`  
**Contact:** [justinchadwickasher@gmail.com](mailto:justinchadwickasher@gmail.com)