
NATURALPROOFS: Mathematical Theorem Proving in Natural Language

Sean Welleck^{1,2}, Jiacheng Liu¹, Ronan Le Bras²,
Hannaneh Hajishirzi^{1,2}, Yejin Choi^{1,2}, Kyunghyun Cho^{3,4}

¹Paul G. Allen School of Computer Science & Engineering, University of Washington

²Allen Institute for Artificial Intelligence

³New York University

⁴CIFAR Fellow in Learning in Machines & Brains
wellecks@uw.edu

Abstract

Understanding and creating mathematics using natural mathematical language – the mixture of symbolic and natural language used by humans – is a challenging and important problem for driving progress in machine learning. As a step in this direction, we develop NATURALPROOFS, a multi-domain corpus of mathematical statements and their proofs, written in natural mathematical language. NATURALPROOFS unifies broad coverage, deep coverage, and low-resource mathematical sources, allowing for evaluating both in-distribution and zero-shot generalization. Using NATURALPROOFS, we benchmark strong neural methods on mathematical reference retrieval and generation tasks which test a system’s ability to determine key results that appear in a proof. Large-scale sequence models show promise compared to classical information retrieval methods, yet their performance and out-of-domain generalization leave substantial room for improvement. NATURALPROOFS opens many avenues for research on challenging mathematical tasks.¹

1 Introduction

Solving the problem of understanding and creating mathematics using *natural mathematical language* – the mixture of symbolic and natural language used by humans – is a path towards developing agents capable of reasoning. The mixture of symbolic and natural text, along with the existence of a formal counterpart, offers a unique setting for studying reasoning that complements research involving natural language alone or purely within a formal system. Constructing a mathematical proof involves symbolic manipulation, logical and analogical reasoning, as well as knowledge retrieval. Common sense and natural language abilities are needed to articulate the proof in a concise, comprehensible form. Moreover, systems that operate on mathematical text have applications in education and scientific discovery, while bridging informal and formal mathematics can be a key driver of progress in automated reasoning [5, 20, 37].

Recently, techniques from natural language processing have driven advances in *formalized mathematics* (e.g. Polu and Sutskever [30], Rabe et al. [31], Wu et al. [47]), in which mathematics is written in a verifiable formal language that resembles source code, such as Mizar [41], Lean [7], or Metamath [26]. However, this setting does not directly address the *informal* aspect of human mathematics, which is conveyed with a mixture of symbolic and natural language [13]. This aspect is crucial, since advancing *human understanding* is a goal of mathematics [40], and a significant fraction of mathematical knowledge is in natural language text [37].

¹Dataset and code available at <https://github.com/wellecks/naturalproofs>.

Source	ProofWiki
Theorem	Category of Monoids is Category Let Mon be the category of monoids. Then Mon is a metacategory.
Proof	Let us verify the axioms (C1) up to (C3) for a <u>metacategory</u> . We have Composite of Homomorphisms on Algebraic Structure is Homomorphism, verifying (C1). We have monoid (S, \circ) . Now, (C2) follows from <u>Identity Mapping is Left Identity</u> and <u>Identity Mapping is Right Identity</u> . Finally, (C3) follows from <u>Composition of Mappings is Associative</u> . Hence Mon is a <u>metacategory</u> .
Source	Textbook: Real Analysis
Theorem	Suppose that f is continuous on the closed interval $[a, b]$ and differentiable on the open interval (a, b) , and $f(a) = f(b)$. Then $f'(c) = 0$ for some c in the open interval (a, b) .
Proof	Since f is continuous on $[a, b]$, f attains a maximum and a minimum value on $[a, b]$ (Theorem 2.2.9). If these two extreme values are the same, then f is constant on (a, b) , so $f'(x) = 0$ for all x in (a, b) . If the extreme values differ, then at least one must be attained at some point c in the open interval (a, b) , and $f'(c) = 0$, by Theorem 2.3.7.

Table 1: Example theorems and their proofs from NATURALPROOFS. Given a theorem, the mathematical retrieval task consists of retrieving the references (underlined) that occur in its proof. NATURALPROOFS contains data from ProofWiki, Stacks, and two textbooks; we show two sources here and two other sources in Table 12. See Figure 2 and Figure 3 for data format details.

In this paper, we describe NATURALPROOFS, a multi-domain corpus of mathematical statements and their proofs, written in natural mathematical language. NATURALPROOFS contains *broad-coverage* data from ProofWiki,² *deep-coverage* data from the Stacks project,³ and *low-resource, real-world* data from mathematics textbooks. NATURALPROOFS unifies these sources in a common schema and is made publicly available as a resource to drive progress on tasks involving informal mathematics, complementing existing work in this direction (e.g. [11, 12, 43]).

Using NATURALPROOFS, we consider *mathematical reference retrieval*, an analogue of premise selection [1, 12]: given a mathematical claim, retrieve the set of references (theorems, lemmas, definitions) that occur in its proof. This task represents a crucial facet of mathematical reasoning, in which a mathematician determines the key results that appear in a proof. As a bridge towards generative tasks using NATURALPROOFS, we consider *mathematical reference generation*, which requires additionally recovering the order and number of references in each proof. Progress on either task could enable educational applications, such as providing a student with hints or guidance.

In addition to standard *in-distribution* evaluation, the multi-domain nature of NATURALPROOFS allows for evaluating *out-of-distribution*, zero-shot generalization. We design an evaluation protocol that tests a system’s ability to retrieve references for *novel* theorems in each setting, and benchmark methods based on large-scale neural sequence models [8, 21], including a strong *joint retrieval* method that better refines the top of the ranked list, as well as an *autoregressive* variant for reference generation. The neural methods are effective for in-domain retrieval compared to classical techniques, yet out-of-distribution generalization, leveraging symbolic mathematical content, and fully recovering a proof’s references remain as fundamental challenges. NATURALPROOFS opens many possibilities for developing and evaluating machine learning methods on challenging mathematical tasks.

2 Related Work

Machine learning for mathematical theorem proving. A large portion of work integrating machine learning with mathematical reasoning has focused on formalized mathematics. Early work by Urban [41] used machine learning for selecting relevant premises in the Mizar mathematical library that are passed to an automated theorem prover, which was later explored with deep neural networks [1]. Bansal et al. [3] developed the HOLList benchmark based on the HOL Light theorem prover, while

²<https://proofwiki.org/>

³<https://stacks.math.columbia.edu/>

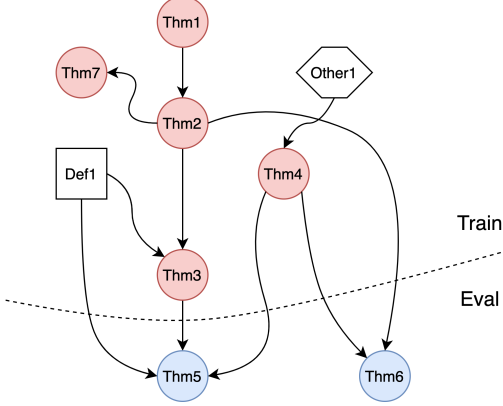


Table 2: The reference graph. Nodes are *statements* and edges are *reference* links. An edge pointing from A to B means that the proof for *theorem* B refers to *statement* A. Edges can start from any type of *statement*, but they always end at a *theorem*. In our tasks, the dataset is split so that all theorems in the evaluation sets are *leaf* nodes in the reference graph.

	Source	All	PWiki	Stacks	RA	NT
Theorem	N	32,579	19,734	12,479	298	68
	Tokens	46.7	38.2	60.6	33.6	23.7
	Lines	5.9	3.6	9.7	8.4	4.5
	Refs	1.8	2.8	0.2	0.0	0.0
Proof	N	32,012	19,234	12,479	235	64
	Tokens	181.5	199.3	155.5	128.9	97.2
	Lines	24.9	25.8	23.4	36.1	16.1
	Refs	5.6	7.4	3.0	1.6	0.9
Definition	N	14,230	12,420	1,687	86	37
	Tokens	48.4	45.0	73.2	58.6	32.6
	Lines	5.0	4.2	10.7	13.3	5.1
	Refs	2.9	3.3	0.4	0.0	0.0
Other	N	1,974	1,006	968	–	–
	Tokens	212.1	286.1	135.2	–	–
	Lines	34.4	46.7	21.7	–	–
	Refs	5.7	9.2	2.0	–	–

Table 3: NATURALPROOFS dataset statistics. Numbers represent mean value, except for "N" rows which represent count. **RA** is the Real Analysis textbook; **NT** is the Number Theory textbook. See Table 14 for detailed statistics.

others use the Coq [19, 48], Metamath [44, 42, 30], Isabelle [24], or Lean [14, 49] environments. These formalized settings differ from NATURALPROOFS, which uses mathematical language as humans write it. Szegedy [37] argues for leveraging both informal and formal mathematics through autoformalization. Wang et al. [43] explore translating between informal and formal mathematics, including via a dataset based on ProofWiki, though their dataset is not made available. Ferreira and Freitas [11, 12] propose a classification-based natural language premise selection task and a dataset based on ProofWiki, while NATURALPROOFS covers multiple domains and provides evaluation and benchmarks for full retrieval and generative tasks. The multiple informal domains, evaluation protocol, joint retrieval model, and reference generation task distinguish our work from previous work on ProofWiki and formalized mathematics.

Mathematics and language benchmarks. Several datasets evaluate a model’s ability to solve multiple-choice algebraic word problems [35, 25, 2] or arithmetic problems [36] with varying degrees of natural language. Lample and Charton [22] evaluate neural sequence models on symbolic integration problems, while Hendrycks et al. [16] propose a benchmark based on math competition problems. NATURALPROOFS focuses on theorem proving rather than calculation, which we hypothesize evaluates different skills, and may prove useful in bridging formal and informal settings.

Large-scale neural language models. Large-scale unsupervised pretraining of language models has led to significant advances in many natural language processing domains (e.g. [8, 32, 33, 4]). Recent work suggests that these models store knowledge in their parameters [29], are capable of reasoning in mathematical [31, 47] and language [6, 38] domains, and are effective for information retrieval tasks [27, 28]. These advances motivate our work, which explores mathematical reasoning in natural language with large-scale language models through a retrieval task.

3 The NATURALPROOFS Dataset

The NATURALPROOFS Dataset is a large-scale, multi-domain dataset for studying mathematical reasoning in natural language. NATURALPROOFS consists of 32k theorem statements and proofs, 14k definitions, and 2k other types of pages (e.g. axioms, corollaries). Table 3 shows dataset statistics.

Multi-domain. NATURALPROOFS contains data derived from three domains:

1. *Broad-coverage* data that encompasses many mathematical topics (e.g. Set Theory, Analysis). We source data from ProofWiki (<https://proofwiki.org/>), an online compendium of mathematical proofs written by a community of contributors. Table 1 shows a theorem and its proof from ProofWiki. There are 31 top-level topic categories in this domain (§A.2).
2. *Deep-coverage* data that focuses on a single topic. We use the Stacks project (<https://stacks.math.columbia.edu/>), a collaborative web-based textbook of algebraic geometry written for graduate students and researchers. See Appendix Table 12 for an example.
3. *Low-resource, real-world* data that poses generalizability challenges to informal theorem proving systems. We use two open-source math textbooks with rich theorem-proof structures and reference links, specifically *Introduction to Real Analysis* (**RA** in short) by William F. Trench and *Elementary Number Theory: Primes, Congruences, and Secrets* (**NT** in short) by William Stein. See Table 1 and Appendix Table 12 for examples from each textbook.

NATURALPROOFS provides a common schema for mathematical statements, proofs, and the references that appear in each domain. Its multiple domains provide a challenging evaluation setting for models and opens opportunities for investigating domain transfer, out-of-distribution generalization, and methods for low-resource settings. This differs from existing resources that focus only on ProofWiki [11, 12], and reflects shifts in natural language processing towards multi-domain settings [45, 18], out-of-distribution generalization [23, 15, 39], and few- or zero-shot generalization in resource-constrained settings [4, 9].

Structure. Each *statement* in NATURALPROOFS is either a theorem or a definition. NATURALPROOFS provides the statement’s title, contents, and references. The *contents* is a list of sequences, where each sequence contains one line of mixed text and \LaTeX , with reference links displayed in their natural language forms. A *theorem* is associated with one or more proofs when available. A *proof* contains a title, contents, and references in the same format as a statement. Finally, we collect *other* pages (e.g. axioms, corollaries). A *reference* is a theorem, definition, or other page that is linked to within the contents of a statement or proof. Figure 3 shows the data format for theorems, definitions, and proofs in NATURALPROOFS. All statements and the reference links connecting them form a *reference graph*, shown in Table 2. The reference graph can contain cycles, e.g. Pythagoras’s Theorem and Sum of Squares of Sine and Cosine refer to each other in their proofs.

Data sources and preprocessing. We describe how we retrieve data from each source and give an overview of preprocessing; for full details see Appendix A.1 and the Jupyter notebooks we release.

- **ProofWiki.** We download the public ProofWiki XML dump,⁴ which contains a snapshot of all pages on ProofWiki. We filter pages according to manual rules (e.g. redirects, files, categories), and determine page type, title, contents, and references using each page’s Wikimedia data structure.
- **Stacks.** We pull the Stacks GitHub repo,⁵ which contains multiple \LaTeX files for various sub-topics in algebraic geometry. We extract statements and proofs by \LaTeX environment names. For example, the content enclosed by $\text{\textbackslash begin\{theorem\}}$ and $\text{\textbackslash end\{theorem\}}$ would be considered a theorem.
- **Textbooks.** We downloaded the \LaTeX source of the **RA**⁶ and **NT**⁷ textbooks, and similarly extracted statements and proofs by environment names. In both textbooks, every statement is either a theorem or a definition – there are no statements that fall under "others".

4 NATURALPROOFS Reference Retrieval and Generation Tasks

NATURALPROOFS opens many possible machine learning tasks that involve natural mathematical language. We consider **mathematical reference retrieval**: given a theorem x , retrieve the set of references y that occur in its proof. An example is shown in Table 1, where the task is to retrieve the underlined references given the title and contents of the theorem *Category of Monoids is*

⁴<https://proofwiki.org/xml dump/latest.xml>. We use the November 12, 2020 version. ProofWiki is licensed under CC BY-SA 3.0.

⁵<https://github.com/stacks/stacks-project>. We use the April 15, 2021 version (commit 4df67b8). Stacks is licensed under GNU Free Documentation License.

⁶<https://digitalcommons.trinity.edu/mono/7/>. Retrieved on April 15, 2021. We did not use the supplementary materials. This textbook is licensed under CC BY-NC-SA 3.0.

⁷<https://github.com/williamstein/ent>. Retrieved on April 15, 2021. We provide a script to download and format the publicly available latex source.

	Split	P+S	ProofWiki	Stacks	RA	NT
Examples $ \mathcal{E} $	total	25,271	14,698	10,573	167	40
	train	21,446	12,424	9,022	–	–
	valid	1,914	1,139	775	–	–
	test	1,911	1,135	776	167	40
Refs $ \mathcal{R} $	train	42,056	28,473	13,583	–	–
	valid	45,805	30,671	15,134	–	–
	test	45,805	30,671	15,134	384	105
Refs/Ex $ \mathbf{y} $	train	5.9	7.5	3.6	–	–
	valid	5.6	7.5	2.9	–	–
	test	5.6	7.4	2.9	2.2	1.5

Table 4: NATURALPROOFS retrieval dataset statistics. **P+S** refers to the combined dataset from the ProofWiki and Stacks sources. **RA** (Real Analysis) and **NT** (Number Theory) are data from textbook sources that we use for zero-shot evaluation.

Category. As a proof is ultimately written as an ordered collection of statements with references often occurring more than once, we also consider **mathematical reference generation**: generate the *sequence* of references that occur in a given theorem’s proof. These tasks represent a crucial aspect of theorem proving, in which a mathematician determines the key results that appear in a proof. Each task also enables educational applications, such as providing hints to a student about which previous results to use, or guidance on how to structure a proof.

Reference retrieval and generation. Each theorem \mathbf{x} has a proof containing a sequence of references $\mathbf{y} = (\mathbf{r}_1, \dots, \mathbf{r}_{|\mathbf{y}|})$, where each reference $\mathbf{r}_m \in \mathcal{R}$ is either a theorem, definition, or other statement (see §3). We consider two tasks: *retrieval* and *generation*.

In the *retrieval* task, given an input theorem \mathbf{x} , a model assigns a score to each reference in \mathcal{R} , inducing a ranked list $\hat{\mathbf{r}}^{(1)}, \dots, \hat{\mathbf{r}}^{(|\mathcal{R}|)}$. These ranked references are evaluated against the ground-truth reference set using standard retrieval metrics such as mean average precision (MAP), recall ($\text{REC}@k$), and full recovery ($\text{FULL}@k$), which checks whether all references in the proof are in the top- k predicted rankings. This reflects the goal of fully proving a theorem using a fixed number of results. For further details, refer to Appendix B.4.

In the *generation* task, a model produces a variable-length sequence of references $(\hat{\mathbf{r}}_1, \dots, \hat{\mathbf{r}}_{|\hat{\mathbf{y}}|})$ given an input \mathbf{x} , with the goal of exactly matching the ground-truth reference sequence $(\mathbf{r}_1, \dots, \mathbf{r}_{|\mathbf{y}|})$. Unlike retrieval, generation requires the model to correctly predict the total number of references, the number of occurrences of each unique reference, and their orders in the proof.

Input-output examples. Using NATURALPROOFS, we derive examples of the form (\mathbf{x}, \mathbf{y}) , where $\mathbf{x} = (x_1, \dots, x_T)$ is a theorem, and $\mathbf{y} = (\mathbf{r}_1, \dots, \mathbf{r}_{|\mathbf{y}|})$ is the sequence of references that occur in the proof of \mathbf{x} . For retrieval, we transform each sequence into a set $\mathbf{y} = \{\mathbf{r}_1, \dots, \mathbf{r}_{|\mathbf{y}|}\}$. The set of all references, \mathcal{R} , consists of theorems, definitions, and other statements (see §3). We use theorems with at least one proof that has at least one reference, resulting in a dataset with roughly 25k examples and a reference set \mathcal{R} with 46k unique references. We partition the dataset into ProofWiki-only, Stacks-only, and textbook-only datasets. Table 4 summarizes the size, total references, and average references per example in each dataset.

Training and evaluation splits. We design training and evaluation splits that reflect the real-world scenario of proving *newly seen* theorems at evaluation time. This requires careful attention, since naively sampling evaluation examples would yield evaluation theorems that appear as references in the training set. To ensure that the theorems in the evaluation set have no overlap with the references in the training set, we form an evaluation set using a randomly sampled subset of *reference graph leaf nodes*, and use the remaining nodes as the training set (Table 2). We use roughly half of the evaluation set for validation and the other half for testing. Since evaluation theorems are not referred to in training examples, the reference set for training is smaller than that for evaluation (Table 4).

5 Methods

As benchmark methods for our tasks, we introduce two *parallel retrieval* methods, and a *sequential retrieval* method trained for sequence generation. See Appendix B for further implementation details.

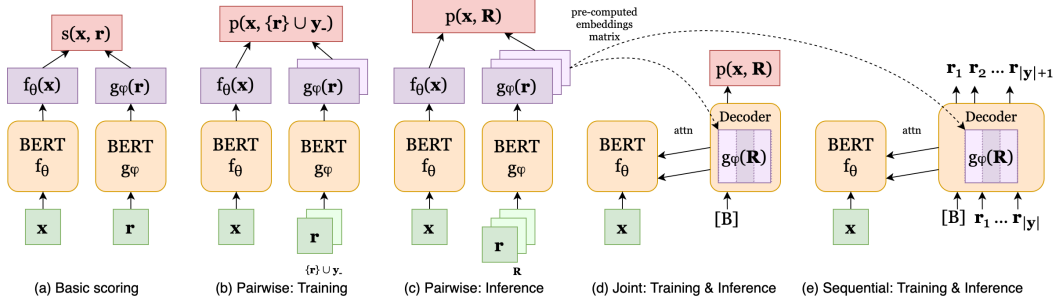


Figure 1: The pairwise, joint and sequential methods for mathematical reference retrieval.

Parallel retrieval. Given a theorem \mathbf{x} , a retrieval model should assign high scores to references in the proof of \mathbf{x} and low scores to all other references, which corresponds to minimizing,

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = \text{KL}(p_*(\mathcal{R}|\mathbf{x}) \| p_\theta(\mathcal{R}|\mathbf{x})) \quad (1)$$

$$\propto - \sum_{\mathbf{r} \in \mathbf{y}} \log \frac{\exp(s_\theta(\mathbf{x}, \mathbf{r}))}{\sum_{\mathbf{r}' \in \mathcal{R}} \exp(s_\theta(\mathbf{x}, \mathbf{r}'))} + \text{const}, \quad (2)$$

where each distribution is over reference indices (i.e. in $\Delta^{(|\mathcal{R}|)}$), and $p_*(\mathbf{r}|\mathbf{x}) \propto \mathbb{I}[\mathbf{r} \in \mathbf{y}]$. The denominator requires scores $s_\theta(\mathbf{x}, \mathbf{r})$ for all $|\mathcal{R}|$ references, making backpropagation too expensive when a large-scale neural model is used to compute reference representations. As a result we consider two variants: a *pairwise* model that approximates Equation 1, and a *joint* model that computes Equation 1 but with implicit vector representations of each reference.

Pairwise parameterization. The pairwise model scores a reference \mathbf{r} against a theorem \mathbf{x} independent of other references, $s_\theta(\mathbf{x}, \mathbf{r}) = f_{\theta_1}^{\text{thm}}(\mathbf{x})^\top g_{\theta_2}^{\text{ref}}(\mathbf{r})$. The model is trained to contrast each positive reference with a set of negative references,

$$\mathcal{L}(\mathbf{x}, \mathbf{r}, \mathbf{y}_-) = - \log \frac{\exp(s_\theta(\mathbf{x}, \mathbf{r}))}{\exp(s_\theta(\mathbf{x}, \mathbf{r})) + \sum_{\mathbf{r}_- \in \mathbf{y}_-} \exp(s_\theta(\mathbf{x}, \mathbf{r}_-))}, \quad (3)$$

where \mathbf{r} is a reference that occurs in the proof of \mathbf{x} , and \mathbf{y}_- is a (small) set of negative references. At inference time, every reference is scored against an input theorem \mathbf{x} , inducing a ranking $\mathbf{r}^{(1)}, \dots, \mathbf{r}^{(|\mathcal{R}|)}$. The scoring, training, and inference are illustrated in Figure 7 (a-c), respectively.

In practice, we use two instances of BERT [8] and in-batch negatives as in [21]. The pairwise model represents retrieval methods such as the dense passage retriever [21] and similar methods [27], and allows for evaluating large-scale sequence models on mathematical reference retrieval.

Joint parameterization. The joint model scores all references in a single pass,

$$p_\theta(\mathcal{R} | \mathbf{x}) = \text{softmax}(\mathbf{R}f_\theta(\mathbf{x})), \quad (4)$$

where $\mathbf{R} \in \mathbb{R}^{|\mathcal{R}| \times d}$ is a reference embedding matrix and $f_\theta(\mathbf{x}) \in \mathbb{R}^d$ is a neural theorem encoder. This model has the advantage of computing the loss denominator in Equation 1 over *all* references rather than a subset of negatives. However, it must learn implicit representations of each reference without observing reference contents. To give the model access to representations that were learned using reference contents, we populate its embedding matrix as,

$$\mathbf{R} = \begin{bmatrix} - & g^{\text{ref}}(\mathbf{r}_1) & - \\ & \vdots & \\ - & g^{\text{ref}}(\mathbf{r}_{|\mathcal{R}|}) & - \end{bmatrix}, \quad (5)$$

where $g^{\text{ref}}(\mathbf{x})$ is from a trained pairwise model. The joint model is illustrated in Figure 7 (d).

Sequential generation and retrieval. Finally, we consider an autoregressive model (Figure 7 (e)),

$$p_\theta(\mathbf{r}_1, \dots, \mathbf{r}_{|\mathbf{y}|} | \mathbf{x}) = \prod_{t=1}^{|\mathbf{y}|+1} p_\theta(\mathbf{r}_t | \mathbf{r}_{<t}, \mathbf{x}), \quad (6)$$

ProofWiki						Stacks				
	mAP	R@10	R@100	Full@10	Full@100	mAP	R@10	R@100	Full@10	Full@100
Random	0.04	0.00	0.19	0.00	0.00	0.07	0.05	0.60	0.00	0.13
Frequency	3.38	5.90	24.30	0.44	2.29	0.91	1.76	11.27	0.13	2.45
TF-IDF	6.19	10.27	23.09	4.14	9.43	13.64	25.46	47.36	18.94	37.76
BERT (P+S)										
+pair	13.54	20.10	58.75	6.17	31.28	18.58	34.42	71.80	28.48	65.21
+joint	32.71	37.59	73.72	17.71	48.90	26.88	35.71	72.68	28.99	66.11
BERT (P/S)										
+pair	16.82	23.73	63.75	7.31	38.50	20.93	37.43	74.21	30.03	66.37
+joint	36.75	42.45	75.90	20.35	50.22	28.32	39.10	73.61	31.96	65.59

Table 5: *In-domain* performance on the mathematical reference retrieval task (test set). **BERT (P/S)** is finetuned on the part of dataset with the same source as the evaluation set, whereas **BERT (P+S)** is finetuned on the combined dataset from ProofWiki and Stacks sources. Recall is micro-averaged.

where $\mathbf{r}_{|y|+1}$ is a special $\langle \text{eos} \rangle$ token denoting the end of the reference sequence. The autoregressive model is trained to maximize the log-likelihood of ground-truth reference sequences. Unlike the parallel retrieval models, this model predicts the order and total number of references and can predict multiple occurrences of each reference. It also adjusts its predictions based on preceding predictions.

For generation, a standard decoding algorithm (e.g. beam search) is used to generate a reference sequence $\hat{\mathbf{y}} = (\hat{\mathbf{r}}_1, \dots, \hat{\mathbf{r}}_{|\hat{\mathbf{y}}|} \langle \text{eos} \rangle)$. For retrieval, we populate a ranked list using generations $\{\hat{\mathbf{r}}_1, \dots, \hat{\mathbf{r}}_{|\hat{\mathbf{y}}|}\}$ followed by references ordered according to the first step’s probabilities, $p_\theta(\mathbf{r}_1|\mathbf{x})$.

6 Experiments

First, we benchmark the neural retrieval methods (§5) on mathematical reference retrieval in terms of their *in-domain* performance (Table 5) and their *out-of-domain* performance on an evaluation set formed from the textbooks in NATURALPROOFS (Table 7). We perform several analyses to better understand each method’s strengths, weaknesses, and the factors that contribute to their performance.

Experimental setup. For the pairwise model, we use separate instances of bert-base-cased to parameterize the theorem encoder $f_{\theta_1}^{\text{thm}}$ and reference encoder $g_{\theta_2}^{\text{ref}}$. We implement the autoregressive model as a sequence-to-sequence encoder-decoder model. Following Rothe et al. [34], we parameterize the encoder and decoder using separate instances of bert-base-cased. This allows for initializing with pairwise model components. The joint retrieval model is implemented as a one-step variant of the autoregressive model. Pairwise models are trained for 500k steps (50 epochs for the autoregressive and joint models), evaluated every 5k steps (or 5 epochs) and the model with the highest validation mAP is selected for final evaluation. Refer to Appendix B for further details.

Retrieval metrics. We evaluate with standard retrieval metrics – mean average prevision (mAP) and recall@ k (R@ k) – and a Full@ k metric measuring whether *all* true references are retrieved in the top- k results. We use $k = 10$ and $k = 100$ for our evaluation. Refer to Appendix B.4 for definitions.

In-domain performance. The BERT-based retrieval models show strong in-domain performance compared to the classical TF-IDF and naive baselines in terms of average precision, recall, and the ability to fully recover all true references within the top- k results, as seen in Table 5. On both ProofWiki and Stacks, the pairwise models outperform TF-IDF, with improvements that are consistent across reference types (Appendix Table 17).

Joint parameterization substantially improves over the pairwise models that are the starting point of joint training. On ProofWiki, the joint model ranks roughly 4 out of every 10 true references within its top 10 rankings (R@10 42.45) compared to 1 out of 10 for TF-IDF, and an impressive 75% within its top 100. For roughly half of the theorems, the joint model’s top 100 references contain *all* of the references needed to prove the theorem (Full@100 50.22). On Stacks the recall@10 is similar at roughly 40%, with a higher full recovery rate of 66% for the top 100 results.

The gains from the joint parameterization are most prominent on ProofWiki, e.g. increasing mAP from 16.82 to 36.75. Joint parameterization particularly excels at refining the top of the ranked list compared to pairwise parameterization; the percentage improvement in the @10 metrics are larger than those for @100 metrics. On Stacks, the improvements are more modest: though mAP improves by 40%, the other metrics are relatively close, suggesting that advances beyond the joint

Source	ProofWiki		
Theorem	Category of Monoids is Category Let Mon be the category of monoids. Then Mon is a metacategory.		
	Ground-Truth Reference	Rank (Pairwise)	Rank (Joint)
	Metacategory	1	1
	Identity Mapping is Left Identity	4	5
	Identity Mapping is Right Identity	5	4
	Monoid	11	2
	Composition of Mappings is Associative	21	8
	Identity Mapping is Automorphism	117	64
	Composite of Homomorphisms is Homomorphism	261	54
Rank	Reference (Pairwise)	Reference (Joint)	
1	<i>Metacategory</i>	<i>Metacategory</i>	
2	Monoid Category is Category	<i>Monoid</i>	
3	Monoid Category	Identity Morphism	
4	<i>Identity Mapping is Left Identity</i>	<i>Identity Mapping is Right Identity</i>	
5	<i>Identity Mapping is Right Identity</i>	<i>Identity Mapping is Left Identity</i>	
6	Category	Associative	
7	Composition of Morphisms	Identity (Abstract Algebra)/Two-Sided Identity	
8	Dual Category is Category	<i>Composition of Mappings is Associative</i>	
9	Identity Morphism	Composition of Morphisms	
10	Morphism Category	Semigroup	

Table 6: Retrieval for a representative theorem. Top: predicted ranks for ground-truth references using the pairwise (left) and its joint (right) BERT models. Bottom: top 10 retrievals from the pairwise (left) and joint (right) models. A retrieved reference is italicized when it is a ground-truth reference.

	Real Analysis			Number Theory		
	mAP	R@10	Full@10	mAP	R@10	Full@10
TF-IDF	15.79	34.65	27.54	16.42	39.62	30.00
BERT-pair (P)	13.24	24.01	19.16	15.12	41.51	35.00
+joint	11.24	20.97	16.77	15.85	41.51	35.00
BERT-pair (S)	11.56	21.28	14.97	12.58	26.42	20.00
+joint	7.04	11.55	9.58	14.88	26.42	20.00

Table 7: *Zero-shot* retrieval performance on out-of-domain textbooks.

model are needed. This demonstrates the importance of evaluating on multiple domains: each domain presents novel challenges for driving advances in modeling. Finally, the BERT models trained on both ProofWiki and Stacks (**BERT (P+S)**) show the possibility of training a single multi-domain model, albeit with lower per-domain performance than the models trained individually on each domain.

Qualitative evaluation. Table 6 shows model predictions for a representative theorem, *Category of Monoids is Category*. The pairwise model retrieves three out of seven true references within its top 50 results, while the joint model retrieves five out of seven. The top 10 results for both models are comprised of references that are related to category theory, which is the subject of the theorem. This illustrates the model’s ability to retrieve *relevant* references, while highlighting its inability to always perform the fine-grained distinction between a relevant reference and one that occurs in the ground-truth proof(s). Arguably, such a system is still useful for providing hints to a user, so long as the user is confident that all of the true references are in a reasonably small set of results.

Out-of-domain performance. While strong in-domain performance drives applications in scenarios where training data is available, an ambitious goal is building a system with mathematical retrieval skills that automatically generalize to new resources. To evaluate the retrieval methods in this zero-shot, out-of-domain setting, we use each textbook from NATURALPROOFS as an evaluation set. This tests situations where the same theorem is expressed using different language (e.g. Table 13), generalization across data formats, and whether retrieval ability from in-domain training transfers.

Table 7 shows the results. The pairwise BERT model trained on ProofWiki underperforms TF-IDF on the Real Analysis textbook, and has comparable performance on the Number Theory textbook. Joint training did not improve out of domain performance, despite its favorable in-domain impact.

	Model	Sequence					Multiset		Set		
		EM	Edit(↓)	BLEU ₄	BLEU ₂	Len	EM	F1	EM	F1	BLEU ₁
Stacks	*-set	51.74	35.70	9.75	47.73	0.97	89.03	97.04	100.0	100.0	94.09
	*-multiset	49.42	38.13	9.71	47.71	1.00	100.0	100.0	100.0	100.0	100.0
	*-halfseq	0.00	70.49	6.13	12.08	0.30	0.00	56.86	0.65	58.01	16.87
	Joint	0.00	98.81	0.00	3.42	2.82	0.00	19.24	0.00	19.65	15.15
	Autoregressive	3.87	90.65	0.00	2.59	0.97	4.00	13.14	4.90	15.04	10.06
ProofWiki	*-set	18.09	58.51	7.18	29.50	0.83	49.96	82.57	100.0	100.0	65.57
	*-multiset	19.23	58.09	16.68	52.89	1.00	100.0	100.0	100.0	100.0	100.0
	*-halfseq	0.00	58.84	25.88	29.17	0.41	0.00	63.33	4.21	70.26	30.55
	Joint	0.00	93.03	0.00	6.88	1.42	0.09	25.30	0.18	30.76	19.27
	Autoregressive	3.69	84.30	5.48	11.90	1.18	3.78	25.61	4.65	28.97	20.81

Table 8: In-domain *generation* results. We show the autoregressive model, a retrieval-only baseline using the top-5 predictions from the joint retrieval model, and oracle benchmarks for correctly predicting the first half of the sequence (*-halfseq), the full multiset with randomized order (*-multiset), and the full set with randomized order (*-set). The best model-based method is in bold.

Training BERT on ProofWiki outperforms training on Stacks, showing that the training domain impacts out-of-domain generalization. ProofWiki’s broad coverage of mathematics may help the model generalize better than the deep, single-topic coverage in Stacks.

The BERT models show some evidence of generalizing to out-of-domain mathematical sources, yet they do not show an advantage over traditional retrieval methods despite strong in-domain performance. This aligns with recent findings about neural retrieval models in various zero-shot settings [39]. An exciting research direction is using NATURALPROOFS to develop and evaluate methods which improve not only in-domain performance, but out-of-domain generalization.

6.1 Reference Generation

Next, we establish a benchmark for recovering the *sequence* of references occurring in the proof of each theorem via the reference generation task (§4).

Metrics. We evaluate predicted reference sequences against ground-truth sequences using order-aware **sequence** metrics, as well as unordered **multiset** and **set**-based metrics. Sequence metrics include exact match (**EM**), edit-distance (**Edit**), standard **BLEU₄** score which uniformly weights 1-4 gram precision, **BLEU₂** with only 1-2 gram precision, and average length ratio $\frac{\text{predicted}}{\text{true}}$ (**Len**). Unordered metrics include exact match, **F1**-score (corpus level), and 1-gram precision **BLEU₁**.

Methods. We use the autoregressive model to generate a reference sequence for each theorem using beam search. As a retrieval-only baseline, we form a sequence using the joint retrieval model’s top-5 predictions, ordered by retrieval score. To judge performance and provide a benchmark for future work, we provide three oracle baselines: correctly predicting the first half of the sequence (*-halfseq), the full multiset of references with random order (*-multiset), and the set with random order (*-set).

Results. Table 8 shows the in-domain generation results. The task is challenging, with the autoregressive model exactly matching the ground-truth sequence roughly 3% of the time. The autoregressive model improves over the retrieval-only baseline on order-aware metrics, aside from **BLEU₂** on Stacks. It does length-prediction reasonably well, with length-ratios of 0.97 and 1.18, yet the multiset and set metrics indicate that the autoregressive model struggles to correctly predict the correct references, even after discarding order. The oracle baselines indicate substantial room for future improvement—for instance, predicting only half of each sequence correctly would move ProofWiki **BLEU₄** from 5.48 to 25.88. Developing models along the full spectrum from set-based retrieval, to reference generation, to full proof generation is an exciting use-case for NATURALPROOFS.

6.2 Ablation Studies

Initialization and autoregressive retrieval. As shown in Table 9, the autoregressive model trained for sequence generation substantially improves over the pairwise retrieval model, yet underperforms

Init	Model	mAP
–	Pairwise	16.99
–	Autoregressive	17.77
f^{thm}	Autoregressive	25.07
$f^{\text{thm}}, \mathbf{R}$	Autoregressive	35.37
–	Joint	18.71
f^{thm}	Joint	28.95
$f^{\text{thm}}, \mathbf{R}$	Joint	37.51

Table 9: Initializing with pairwise components, and autoregressive retrieval (ProofWiki).

Train		Eval	
Lang.	NatProof	PW	Stacks
✓	✗	0.14	0.30
✗	✓	0.04	0.86
✓	✓	16.99	21.21

Table 10: Language pretraining and NATURALPROOFS finetuning (pairwise retrieval, mAP).

	Title	Content	PW	Stacks
TF-IDF	✗	✓	4.97	12.34
	✓	✗	8.10	12.69
	✓	✓	6.33	13.45
BERT	✗	✓	16.19	19.12
	✓	✗	24.48	19.15
	✓	✓	16.99	21.21

Table 11: Excluding (✗) the title or content of theorems and references (pairwise retrieval, mAP).

the joint model, which is trained specifically for retrieval. Initializing the joint and autoregressive models using the pairwise model was necessary for achieving high performance; in particular, the reference information conveyed through the embedding matrix (Equation 5) was crucial.

Language pretraining and NATURALPROOFS training. The BERT model has two learning phases: pretraining on language data, and finetuning on NATURALPROOFS. As seen in Table 10, relying on language-pretraining alone without fine-tuning on NATURALPROOFS (top row) led to poor performance. Conversely, training from scratch on NATURALPROOFS (middle row) was unsuccessful, suggesting that language pretraining served as an effective initialization for mathematical retrieval.

Title and content ablation. Each theorem statement and reference consists of a title, as well as contents that is a mixture of symbolic mathematics and natural language. As seen in Table 11, ProofWiki’s titles contain a large amount of useful information for retrieval—TF-IDF and the pairwise BERT model performed better with only access to titles. In principal, the title+content model could learn to ignore the contents if needed, so its lower performance shows a deficiency in the pairwise model. On Stacks, the model performs best with both sources of information, though the degree of improvement suggests that leveraging the mathematical content remains as a fundamental challenge.

7 Conclusion

Building agents that understand and create mathematics using *natural mathematical language* is a challenging research direction, providing a means for evaluating and developing machine learning methods capable of symbolic reasoning and natural language understanding. As a step in this direction, we develop NATURALPROOFS, a multi-domain dataset for studying mathematical reasoning in natural language. NATURALPROOFS allows for evaluating *in-domain* performance, and *out-of-domain* generalization in broad and deep coverage mathematics, as well as real-world, low-resource settings. We establish benchmarks for retrieval and generation tasks that represent key steps in real-world theorem proving, and are tractable, yet challenging, for current large-scale neural sequence models. NATURALPROOFS opens many promising avenues for future research.

Broader Impacts Statement Our work pertains to use of the natural language in mathematical theorem proving, and more generally reasoning in artificial intelligence. Although a general reasoning agent may present negative societal impacts, we do not foresee any immediate negative societal impact from the domain, dataset, tasks, and study that we present here. Instead, we foresee positive societal impacts through applications in education and scientific discovery that are enabled by systems which understand and create natural mathematical content.

Acknowledgements This work was funded in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) (funding reference number 401233309), DARPA MCS program through NIWC Pacific (N66001-19-2-4031), and the Allen Institute for AI. We also thank Google Cloud Compute, as well as OpenAI. The authors would like to thank Deborah Ferreira, Jesse Han, Christian Szegedy, and Josef Urban for helpful discussions and feedback on earlier drafts.

References

- [1] A. A. Alemi, F. Chollet, N. Eén, G. Irving, C. Szegedy, and J. Urban. DeepMath - Deep sequence models for premise selection. In *Advances in Neural Information Processing Systems*, pages

2243–2251, 2016.

- [2] A. Amini, S. Gabriel, S. Lin, R. Koncel-Kedziorski, Y. Choi, and H. Hajishirzi. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference, volume 1, 2019.
- [3] K. Bansal, S. Loos, M. Rabe, C. Szegedy, and S. Wilcox. Holist: An environment for machine learning of higher-order theorem proving. In 36th International Conference on Machine Learning, ICML 2019, volume 2019-June, 2019.
- [4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language Models are Few-Shot Learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL <https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf>.
- [5] N. C. Carter and K. G. Monks. Lurch: a word processor that can grade students’ proofs. In C. Lange, D. Aspinall, J. Carette, J. H. Davenport, A. Kohlhase, M. Kohlhase, P. Libbrecht, P. Quaresma, F. Rabe, P. Sojka, I. Whiteside, and W. Windsteiger, editors, Joint Proceedings of the MathUI, OpenMath, PLMMS and ThEdu Workshops and Work in Progress at CICM, Bath, UK, volume 1010 of CEUR Workshop Proceedings. CEUR-WS.org, 2013. URL <http://ceur-ws.org/Vol-1010/paper-04.pdf>.
- [6] P. Clark, O. Tafjord, and K. Richardson. Transformers as Soft Reasoners over Language. In C. Bessiere, editor, Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, pages 3882–3890. International Joint Conferences on Artificial Intelligence Organization, 2020.
- [7] L. M. de Moura, S. Kong, J. Avigad, F. van Doorn, and J. von Raumer. The lean theorem prover (system description). In A. P. Felty and A. Middeldorp, editors, CADE, volume 9195 of Lecture Notes in Computer Science, pages 378–388. Springer, 2015. ISBN 978-3-319-21400-9. URL <http://dblp.uni-trier.de/db/conf/cade/cade2015.html#MouraKADR15>.
- [8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL <https://www.aclweb.org/anthology/N19-1423>.
- [9] A. Ebrahimi, M. Mager, A. Oncevay, V. Chaudhary, L. Chiruzzo, A. Fan, J. Ortega, R. Ramos, A. Rios, I. Vladimir, G. A. Giménez-Lugo, E. Mager, G. Neubig, A. Palmer, R. A. C. Solano, N. T. Vu, and K. Kann. Americasnli: Evaluating zero-shot natural language understanding of pretrained multilingual models in truly low-resource languages, 2021.
- [10] European Organization For Nuclear Research and OpenAIRE. Zenodo, 2013. URL <https://www.zenodo.org/>.
- [11] D. Ferreira and A. Freitas. Natural language premise selection: Finding supporting statements for mathematical text. In Proceedings of the 12th Language Resources and Evaluation Conference, pages 2175–2182, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL <https://www.aclweb.org/anthology/2020.lrec-1.266>.
- [12] D. Ferreira and A. Freitas. Premise selection in natural language mathematical texts. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7365–7374, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.657. URL <https://www.aclweb.org/anthology/2020.acl-main.657>.

- [13] T. Gowers, J. Barrow-Green, and I. Leader. The Princeton Companion to Mathematics. Princeton University Press, USA, illustrated edition edition, 2008. ISBN 0691118809.
- [14] J. M. Han, J. Rute, Y. Wu, E. W. Ayers, and S. Polu. Proof artifact co-training for theorem proving with language models, 2021.
- [15] D. Hendrycks, X. Liu, E. Wallace, A. Dziedzic, R. Krishnan, and D. Song. Pretrained transformers improve out-of-distribution robustness, 2020.
- [16] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the math dataset, 2021.
- [17] S. Holland, A. Hosny, S. Newman, J. Joseph, and K. Chmielinski. The dataset nutrition label: A framework to drive higher data quality standards. arXiv preprint arXiv:1805.03677, 2018.
- [18] J. Hu, S. Ruder, A. Siddhant, G. Neubig, O. Firat, and M. Johnson. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In H. D. III and A. Singh, editors, Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 4411–4421. PMLR, 13–18 Jul 2020. URL <http://proceedings.mlr.press/v119/hu20b.html>.
- [19] D. Huang, P. Dhariwal, D. Song, and I. Sutskever. Gamepad: A learning environment for theorem proving. In International Conference on Learning Representations, 2019. URL <https://openreview.net/forum?id=r1xwKoR9Y7>.
- [20] D. Kang, A. Head, R. Sidhu, K. Lo, D. Weld, and M. A. Hearst. Document-Level Definition Detection in Scholarly Documents: Existing Models, Error Analyses, and Future Directions. In Proceedings of the First Workshop on Scholarly Document Processing, pages 196–206, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.sdp-1.22. URL <https://www.aclweb.org/anthology/2020.sdp-1.22>.
- [21] V. Karpukhin, B. Oğuz, S. Min, L. Wu, S. Edunov, D. Chen, and W.-t. Yih. Dense passage retrieval for open-domain question answering. arXiv preprint arXiv:2004.04906, 2020.
- [22] G. Lample and F. Charton. Deep learning for symbolic mathematics. In International Conference on Learning Representations, 2020. URL <https://openreview.net/forum?id=S1eZYeHFDS>.
- [23] R. Le Bras, S. Swayamdipta, C. Bhagavatula, R. Zellers, M. E. Peters, A. Sabharwal, and Y. Choi. Adversarial filters of dataset biases, 2020. ISSN 23318422.
- [24] W. Li, L. Yu, Y. Wu, and L. C. Paulson. Isarstep: a benchmark for high-level mathematical reasoning. In International Conference on Learning Representations, 2021. URL <https://openreview.net/forum?id=Pzj6fzU6wkj>.
- [25] W. Ling, D. Yogatama, C. Dyer, and P. Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers), volume 1, 2017. doi: 10.18653/v1/P17-1015.
- [26] N. D. Megill and D. A. Wheeler. Metamath: A Computer Language for Mathematical Proofs. Lulu Press, Morrisville, North Carolina, 2019. <http://us.metamath.org/downloads/metamath.pdf>.
- [27] R. Nogueira and K. Cho. Passage re-ranking with bert, 2020.
- [28] C. Nogueira dos Santos, X. Ma, R. Nallapati, Z. Huang, and B. Xiang. Beyond [CLS] through ranking by generation. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1722–1727, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.134. URL <https://www.aclweb.org/anthology/2020.emnlp-main.134>.

- [29] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel. Language models as knowledge bases? In EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference, 2020. doi: 10.18653/v1/d19-1250.
- [30] S. Polu and I. Sutskever. Generative language modeling for automated theorem proving, 2020.
- [31] M. N. Rabe, D. Lee, K. Bansal, and C. Szegedy. Mathematical reasoning via self-supervised skip-tree training. In International Conference on Learning Representations, 2021. URL <https://openreview.net/forum?id=YmqAnYOCMEy>.
- [32] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.
- [33] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21(140):1–67, 2020. URL <http://jmlr.org/papers/v21/20-074.html>.
- [34] S. Rothe, S. Narayan, and A. Severyn. Leveraging pre-trained checkpoints for sequence generation tasks. Transactions of the Association for Computational Linguistics, 8:264–280, 2020. doi: 10.1162/tac1_a_00313. URL <https://www.aclweb.org/anthology/2020.tac1-1.18>.
- [35] S. Roy and D. Roth. Solving general arithmetic word problems. In Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing, 2015. doi: 10.18653/v1/d15-1202.
- [36] D. Saxton, E. Grefenstette, F. Hill, and P. Kohli. Analysing mathematical reasoning abilities of neural models. In International Conference on Learning Representations, 2019. URL <https://openreview.net/forum?id=H1gR5iR5FX>.
- [37] C. Szegedy, editor. A Promising Path Towards Autoformalization and General Artificial Intelligence, 2020.
- [38] O. Taffjord, B. D. Mishra, and P. Clark. Proofwriter: Generating implications, proofs, and abductive statements over natural language. ArXiv, abs/2012.13048, 2020.
- [39] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. arXiv preprint arXiv:2104.08663, 4 2021. URL <https://arxiv.org/abs/2104.08663>.
- [40] W. P. Thurston. On proof and progress in mathematics. arXiv:math/9404236, Mar. 1994. URL <http://arxiv.org/abs/math/9404236>. arXiv: math/9404236.
- [41] J. Urban. Mptp 0.2: Design, implementation, and initial experiments. J. Autom. Reason., 37 (1–2):21–43, Aug. 2006. ISSN 0168-7433. doi: 10.1007/s10817-006-9032-3. URL <https://doi.org/10.1007/s10817-006-9032-3>.
- [42] M. Wang and J. Deng. Learning to Prove Theorems by Learning to Generate Theorems. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 18146–18157. Curran Associates, Inc., 2020. URL <https://proceedings.neurips.cc/paper/2020/file/d2a27e83d429f0dcae6b937cf440aeb1-Paper.pdf>.
- [43] Q. Wang, C. Brown, C. Kaliszyk, and J. Urban. Exploration of neural machine translation in autoformalization of mathematics in Mizar. In CPP 2020 - Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, co-located with POPL 2020, 2020. doi: 10.1145/3372885.3373827.
- [44] D. Whalen. Holophrasm: a neural automated theorem prover for higher-order logic, 2016.

- [45] A. Williams, N. Nangia, and S. R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference, 2018. ISBN 9781948087278. doi: 10.18653/v1/n18-1101.
- [46] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-art natural language processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 38–45, Online, Oct. 2020. Association for Computational Linguistics. URL <https://www.aclweb.org/anthology/2020.emnlp-demos.6>.
- [47] Y. Wu, M. Rabe, W. Li, J. Ba, R. Grosse, and C. Szegedy. Lime: Learning inductive bias for primitives of mathematical reasoning, 2021.
- [48] K. Yang and J. Deng. Learning to prove theorems via interacting with proof assistants. In 36th International Conference on Machine Learning, ICML 2019, volume 2019-June, 2019.
- [49] K. Zheng, J. M. Han, and S. Polu. Minif2f: a cross-system benchmark for formal olympiad-level mathematics. 2021.

Appendix

A Dataset Details

Table 12 shows example theorems and proofs from more data sources. Table 13 shows an example of the same theorem extracted from different sources. Table 14 gives more detailed statistics of the dataset. Figure 2 shows the JSON format of an example theorem, whereas Figure 3 shows the data schema we use to standardize data collected from different sources.

Source	Stacks
Theorem	Lemma 9.7 Let S be a scheme. Let $f : X \rightarrow S$ be locally of finite type with X quasi-compact. Then $\text{size}(X) \leq \text{size}(S)$.
Proof	We can find a finite affine open covering $X = \bigcup_{i=1, \dots, n} U_i$ such that each U_i maps into an affine open S_i of S . Thus by Lemma 9.5 we reduce to the case where both S and X are affine. In this case by Lemma 9.4 we see that it suffices to show $ A[x_1, \dots, x_n] \leq \max\{\aleph_0, A \}$. We omit the proof of this inequality.
Source	Textbook: Number Theory
Theorem	Proposition 2.1.13 If $\gcd(a, n) = 1$, then the equation $ax \equiv b \pmod{n}$ has a solution, and that solution is unique modulo n .
Proof	Let R be a complete set of residues modulo n , so there is a unique element of R that is congruent to b modulo n . By Lemma 2.1.12, aR is also a complete set of residues modulo n , so there is a unique element $ax \in aR$ that is congruent to b modulo n , and we have $ax \equiv b \pmod{n}$.

Table 12: Example theorems and their proofs from the Stacks and Number Theory textbook sources.

Source	ProofWiki
Theorem	Solution of Linear Congruence/Unique iff Coprime to Modulus If $\gcd\{a, n\} = 1$, then $ax \equiv b \pmod{n}$ has a <u>unique</u> solution.
Proof	From Solution of Linear Congruence: Existence: the problem of finding all integers satisfying the <u>linear congruence</u> $ax \equiv b \pmod{n}$ is the same problem as: the problem of finding all the x values in the <u>linear Diophantine equation</u> $ax - ny = b$. Let: $\gcd\{a, n\} = 1$ Let $x = x_0, y = y_0$ be one solution to the <u>linear Diophantine equation</u> : $ax - ny = b$ From <u>Solution of Linear Diophantine Equation</u> , the general solution is: $\forall k \in \mathbb{Z} : x = x_0 + nk, y = y_0 + ak$ But: $\forall k \in \mathbb{Z} : x_0 + nk \equiv x_0 \pmod{n}$ Hence $x \equiv x_0 \pmod{n}$ is the only solution of $ax \equiv b \pmod{n}$.
Source	Textbook: Number Theory
Theorem	Units If $\gcd(a, n) = 1$, then the equation $ax \equiv b \pmod{n}$ has a solution, and that solution is unique modulo n .
Proof	Let R be a complete set of residues modulo n , so there is a unique element of R that is congruent to b modulo n . By Lemma 2.1.12, aR is also a complete set of residues modulo n , so there is a unique element $ax \in aR$ that is congruent to b modulo n , and we have $ax \equiv b \pmod{n}$.

Table 13: Example of the same theorem extracted from two different sources.

A.1 Preprocessing Details

ProofWiki. The theorem, definition, and proof contents are contained in a WikiMedia section that is determined for each page type according to a hand-defined rule. Since the roughly 1,000 other pages have varying page structures, we use their entire contents instead of a single section’s contents.

Source		All				ProofWiki				Stacks				Textbook: RA				Textbook: NT			
Type	Attr	mean	25%	50%	75%	mean	25%	50%	75%	mean	25%	50%	75%	mean	25%	50%	75%	mean	25%	50%	75%
Theorem	N	32,579	-	-	-	19,734	-	-	-	12,479	-	-	-	298	-	-	-	68	-	-	-
	Chars	320.0	146	275	433	277.9	93	238	393	388.6	215	331	491	278.2	152	225	355	158.4	98	140	179
	Tokens	46.7	21	39	63	38.2	14	32	53	60.6	35	52	76	33.6	19	29	41	23.7	14	21	30
	Lines	5.9	2	4	8	3.6	1	3	5	9.7	4	8	12	8.4	4	7	11	4.5	2	4	5
	Refs	1.8	0	0	3	2.8	0	3	4	0.2	0	0	0	0.0	0	0	0	0.0	0	0	0
Proof	N	32,012	-	-	-	19,234	-	-	-	12,479	-	-	-	235	-	-	-	64	-	-	-
	Chars	1,123.8	388	770	1,449	1,170.0	444	810	1,470	1,053.1	280	705	1,422	1,231.0	442	876	1,634	655.7	327	551	732
	Tokens	181.5	57	121	236	199.3	68	134	254	155.5	36	101	211	128.9	50	92	165	97.2	47	87	115
	Lines	24.9	8	16	32	25.8	9	18	33	23.4	6	15	31	36.1	14	27	47	16.1	8	13	18
	Refs	5.6	2	3	7	7.4	2	5	9	3.0	1	2	4	1.6	0	1	2	0.9	0	1	1
Definition	N	14,230	-	-	-	12,420	-	-	-	1,687	-	-	-	86	-	-	-	37	-	-	-
	Chars	362.3	152	300	491	349.3	131	289	478	459.0	251	380	577	411.8	246	356	509	199.5	118	159	262
	Tokens	48.4	18	39	65	45.0	15	35	61	73.2	41	61	91	58.6	33	49	74	32.6	21	28	43
	Lines	5.0	1	4	6	4.2	1	3	6	10.7	5	9	13	13.3	8	11	17	5.1	3	4	7
	Refs	2.9	0	2	4	3.3	1	3	5	0.4	0	0	1	0.0	0	0	0	0.0	0	0	0
Other	N	1,974	-	-	-	1,006	-	-	-	968	-	-	-								
	Chars	1,399.8	712	1,109	1,680	1,836.5	1,018	1,431	2,131	945.9	480	802	1,198								
	Tokens	212.1	101	158	250	286.1	145	206	337	135.2	70	113	168								
	Lines	34.4	18	28	42	46.7	28	39	49	21.7	10	18	27								
	Refs	5.7	1	3	7	9.2	4	7	11	2.0	0	1	3								

Table 14: NATURALPROOFS dataset statistics (detailed).

In addition to well-formed axiom and corollary statements, the other pages include misformatted theorem or definition statements that occur as references elsewhere in the corpus.

Stacks and textbooks. The raw data we obtain from Stacks and textbook sources are \LaTeX source code. For each data source, we look up with a pre-defined list of environment names, and parse the contents enclosed in these environments into statements or proofs. Each proof is associated with the environment that immediately precedes it. As a result, each theorem has at most one proof. Table 15 lists the mapping from \LaTeX environment name to the data type in the NATURALPROOFS taxonomy.

A few misc notes:

- In Stacks, statements do not have titles, but each has a label with semantic meaning (e.g. `sets-lemma-bound-finite-type` for the example in Table 12), so we use it as a pseudo-title.
- In the Number Theory textbook, proofs are bounded by `(\proof, \bbox)` instead of `(\begin{proof}, \end{proof})`.

Source		Stacks		Textbook: RA		Textbook: NT	
\LaTeX env	Type	\LaTeX env	Type	\LaTeX env	Type	\LaTeX env	Type
theorem	theorem	theorem	theorem	theorem	theorem	theorem	theorem
lemma	theorem	lemma	theorem	lemma	theorem	lemma	theorem
proposition	theorem	corollary	theorem	corollary	theorem	corollary	theorem
definition	definition	definition	definition	definition	definition	definition	definition
remark	other	proof	proof			proof	proof
remarks	other						
proof	proof						

Table 15: Mappings from \LaTeX environment names to NATURALPROOFS data types for each data source. As an example, for Stacks, the mapping from *lemma* to *theorem* in row 2 means that an environment enclosed by `\begin{lemma}` and `\end{lemma}` is considered a theorem in NATURALPROOFS.

A.2 ProofWiki categories.

For ProofWiki, we also provide category tags for each statement. ProofWiki contains statements encompassing a broad coverage of mathematical topics (i.e. categories). In ProofWiki, each category has zero or more sub-categories, and sub-categories have sub-sub-categories, and so on, forming


```

{
  "id": 5480,
  "type": "theorem",
  "label": "Category of Monoids is Category",
  "categories": [ "Category of Monoids" ],
  "toplevel_categories": [ "Algebra", "Set Theory", "Abstract Algebra", "Category Theory" ],
  "recursive_categories": [
    "Category Theory",
    "Algebra",
    "Abstract Algebra",
    "Category of Monoids",
    "Set Theory",
    "Examples of Categories"
  ],
  "title": "Category of Monoids is Category",
  "contents": [
    "Let  $\mathbf{Mon}$  be the [[Definition:Category of Monoids|category of monoids]].",
    "Then  $\mathbf{Mon}$  is a [[Definition:Metacategory|metacategory]]."
  ],
  "refs": [
    "Definition:Category of Monoids",
    "Definition:Metacategory"
  ],
  "ref_ids": [ 22919, 21454 ],
  "proofs": [
    {
      "contents": [
        "Let us verify the axioms  $(C1)$  up to  $(C3)$  for a [[Definition:Metacategory|metacategory]].",
        "We have [[Composite of Homomorphisms on Algebraic Structure is Homomorphism]], verifying  $(C1)$ .",
        "We have [[Identity Mapping is Automorphism]] providing  $\operatorname{id}_S$  for every  

        [[Definition:Monoid|monoid]]  $S$   $\left( S, \circ \right)$ .",
        "Now,  $(C2)$  follows from [[Identity Mapping is Left Identity]] and  

        [[Identity Mapping is Right Identity]].",
        "Finally,  $(C3)$  follows from [[Composition of Mappings is Associative]].",
        "Hence  $\mathbf{Mon}$  is a [[Definition:Metacategory|metacategory]].",
        "{\qquad}",
        "[[Category:Category of Monoids]]",
        "sppgcr1pruam0jkf2euhyvt6y3jpnt0"
      ],
      "refs": [
        "Definition:Metacategory",
        "Composite of Homomorphisms is Homomorphism/Algebraic Structure",
        "Identity Mapping is Automorphism",
        "Definition:Monoid",
        "Identity Mapping is Left Identity",
        "Identity Mapping is Right Identity",
        "Composition of Mappings is Associative",
        "Definition:Metacategory"
      ],
      "ref_ids": [ 21454, 3852, 418, 19948, 217, 4387, 1494, 21454 ]
    }
  ]
}

```

Figure 2: NATURALPROOFS JSON for the theorem and proof shown in Table 1. Using the notation in section 4, an (x, y) example is formed where x is the concatenation of 'title' and 'contents', and y is a set formed with 'ref_ids' of one of the proofs.

a *category graph*.⁸ We recursively scrape the category pages starting from Category:Content Categories,⁹ and consider categories directly under Category:Proofs By Topic as top-level categories. Figure 4 shows the high-level structure of the ProofWiki category graph.

In the ProofWiki raw data, each statement page is tagged with several categories (the 'categories' field). In addition, we find the top-level categories (the 'toplevel_categories' field) as well as exhaustive categories (the 'recursive_categories' field) for each theorem by running flood-fill on the category graph. Figure 5 and Figure 6 show some statistics of the top-level categories.

⁸It is not strictly a tree or DAG, because there are several skip connections (e.g. Complex Analysis is both a top-level category and a sub-category under Analysis) and circular dependencies (e.g. Metric Spaces and Pseudometric Spaces are sub-category of each other)

⁹https://proofwiki.org/wiki/Category:Content_Categories

```

Dataset: {
  'dataset': {
    'theorems': [Statement],
    'definitions': [Statement],
    'others': [Statement],
    'retrieval_examples': [int], // deprecated
  },
  'splits': {
    'train': {
      'ref_ids': [int],
      'examples': [(int, int)],
      // pairs of theorem id and index of proof
    },
    'valid': {
      'ref_ids': [int],
      'examples': [(int, int)],
    },
    'test': {
      'ref_ids': [int],
      'examples': [(int, int)],
    },
  },
}

Statement: {
  'id': int,
  'type': string,
  'label': string,
  'categories': [string],
  'toplevel_categories': [string], // ProofWiki only
  'recursive_categories': [string], // ProofWiki only
  'title': string,
  'contents': [string],
  'refs': [string],
  'ref_ids': [int],
  'proofs': [Proof], // for theorems only
}

Proof: {
  'contents': [string],
  'refs': [string],
  'ref_ids': [int],
}

```

Figure 3: NATURALPROOFS dataset schema.

```

Content Categories
...
Definitions
...
Definitions by Topic
...
Definitions/Branch of Mathematics
Definitions/Abstract Algebra
Definitions/Algebra
Definitions/Analysis
...
Definitions/Topology

Proofs
...
Proofs by Topic
Abstract Algebra
Additive Functions
Examples of Additive Functions
Monotone Additive Function is Linear
Additive Groups
...
Zero Elements
Algebra
Analysis
...
Trigonometry

```

Figure 4: ProofWiki category graph. Nested structure represents sub-categories. Some nesting omitted here for simplicity.

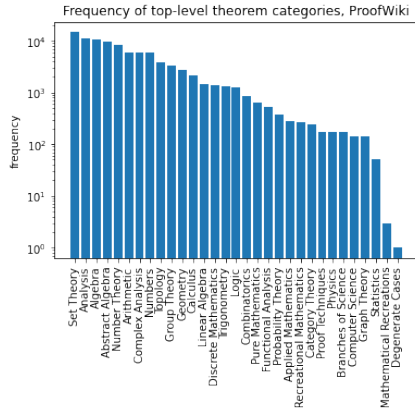


Figure 5: Frequency of top-level categories, ProofWiki.

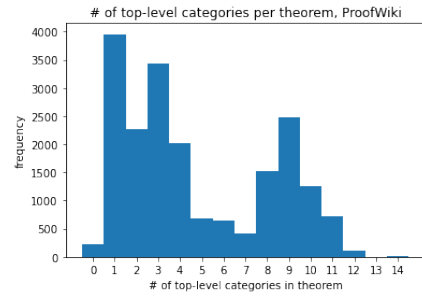


Figure 6: Number of top-level categories per theorem, ProofWiki.

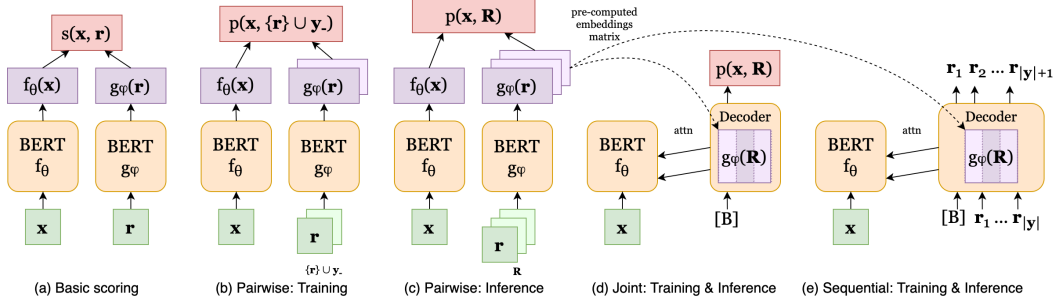


Figure 7: The pairwise, joint and sequential methods for mathematical reference retrieval.

B Implementation Details and Experimental Setup

Model input format. We format each statement (\mathbf{x} or \mathbf{r}) as, [CLS] title [SEP] content [SEP], and we truncate the statement when the sequence exceeds the model’s maximum length. Each sequence is tokenized using the bert-base-cased tokenizer.

B.1 Pairwise model

Models are implemented with transformers [46] and pytorch-lightning¹⁰. The theorem encoder $f_{\theta_1}^{\text{thm}}$ is parameterized using the bert-base-cased architecture and initialized with its parameters. The reference encoder $g_{\theta_2}^{\text{ref}}$ is also parameterized and initialized with (a separate instance of) bert-base-cased.

Training. Models are trained for 500,000 steps on one Quadro RTX 8000 GPU. Each batch contains a maximum of 16,384 (2^{14}) tokens. Validation is done every 5,000 steps. The model with the highest mAP computed on the validation set is selected for final evaluation.

Negatives. We use *in-batch negatives* as in [21], which computes a score matrix $\mathbf{S} = \mathbf{TR}^\top \in \mathbb{R}^{B \times B}$ on a batch of theorem embeddings $\mathbf{T} \in \mathbb{R}^{B \times d}$ and reference embeddings $\mathbf{R} \in \mathbb{R}^{B \times d}$, then defines the loss as $\sum_{i=1}^B \text{softmax}(\mathbf{S}[i, :])$, which treats elements on the diagonal of \mathbf{S} as positives and off-diagonal elements as negatives.

Evaluation. The full set of inputs \mathbf{x} and the full set of references \mathcal{R} are pre-encoded using their respective trained models (i.e. two instances of BERT). Then the encodings for each possible \mathbf{x}, \mathbf{r} pair are used to obtain scalar scores, inducing a ranked list of all $|\mathcal{R}|$ references for each input \mathbf{x} .

B.2 Autoregressive

We implement the autoregressive model as a sequence-to-sequence encoder-decoder model. Following Rothe et al. [34], we parameterize the encoder and decoder using BERT models. This allows for initializing with pairwise model components. Concretely, we implement the architecture using the transformers EncoderDecoderModel class with bert-base-cased encoder and decoder.

Let $f_{\theta_1}(\mathbf{x})$ denote the encoder and $h_{\theta_2}(\mathbf{r}_{<t}, f_{\theta_1}(\mathbf{x}))$ denote the decoder. The decoder has an embedding matrix $\mathbf{R} \in \mathbb{R}^{(|\mathcal{R}|+2) \times d}$, where each row represents a reference or special token $\langle \text{bos} \rangle, \langle \text{eos} \rangle$. At each step t , given a theorem and sequence of tokens $(\langle \text{bos} \rangle, \mathbf{r}_1, \dots, \mathbf{r}_{t-1})$, the decoder produces a next-token distribution $p_\theta(\cdot | \mathbf{x}, \mathbf{r}_{<t}) = \text{softmax}(\mathbf{R}h_t + \mathbf{b})$, where $h_t \in \mathbb{R}^d$ is the final hidden state obtained from the decoder $h_{\theta_2}(\mathbf{r}_{<t}, f_{\theta_1}(\mathbf{x}))$, and $\mathbf{b} \in \mathbb{R}^{(|\mathcal{R}|+2)}$ is a bias vector.

The model is trained using cross-entropy loss with the ground-truth (\mathbf{x}, \mathbf{y}) pairs, where $\mathbf{y} = (\langle \text{bos} \rangle, \mathbf{r}_1, \dots, \mathbf{r}_{|\mathbf{y}|}, \langle \text{eos} \rangle)$ is a reference sequence.

¹⁰<https://github.com/PyTorchLightning/pytorch-lightning>

Initialization. Let $f_{\tilde{\theta}_1}^{\text{thm}}$ and $g_{\tilde{\theta}_2}^{\text{ref}}$ be the theorem and reference encoder from a trained pairwise model (§B.1). The initialization settings listed in Table 9 are as follows. f^{thm} means initializing the encoder f_{θ_1} ’s parameters as $\theta_1 = \tilde{\theta}_1$, and then updating them during training. **R** means initializing and freezing the decoder’s embedding matrix as (omitting the $\langle \text{bos} \rangle$ and $\langle \text{eos} \rangle$ rows),

$$\mathbf{R} = \begin{bmatrix} - & g_{\tilde{\theta}_2}^{\text{ref}}(\mathbf{r}_1) & - \\ & \ddots & \\ - & g_{\tilde{\theta}_2}^{\text{ref}}(\mathbf{r}_{|\mathcal{R}|}) & - \end{bmatrix}.$$

Training. Models are trained for 50 epochs on one Quadro RTX 8000 GPU. Each batch contains a maximum of 16,384 (2^{14}) tokens. Validation is done every 5 epochs. The model with the highest mAP computed on the validation set is selected for final evaluation.

Generation evaluation. Let $\hat{\mathbf{y}} \sim \mathcal{F}(p_\theta, \mathbf{x})$ denote decoding a sequence $\hat{\mathbf{y}} = (\mathbf{r}_1, \dots, \mathbf{r}_{|\hat{\mathbf{y}}|}, \langle \text{eos} \rangle)$ given model p_θ and input \mathbf{x} , using decoding algorithm \mathcal{F} . For the reference generation task (§6.1), we use beam search with beam size 20, based on a preliminary search over beam size $\{1, 10, 20, 50\}$. For retrieval evaluation only, we use greedy decoding (beam size 1) with a 1-gram repetition mask since duplicates are not used during retrieval evaluation. For all decoding algorithms, we use the transformers implementations.

Retrieval evaluation. A retrieval model produces a ranked list $\mathbf{r}^{(1)}, \dots, \mathbf{r}^{(|\mathcal{R}|)}$ given an input \mathbf{x} . We evaluate our autoregressive model as a retrieval model by producing a ranked list $\mathbf{r}^{(1)}, \dots, \mathbf{r}^{(|\hat{\mathbf{y}}|)}, \dots, \mathbf{r}^{(|\mathcal{R}|)}$, where the first $|\hat{\mathbf{y}}|$ references come from the model’s generated sequence $\hat{\mathbf{y}} = (\mathbf{r}^{(1)}, \dots, \mathbf{r}^{(|\hat{\mathbf{y}}|)})$ after removing duplicates, and the remaining references are ordered according to the model’s first-step probabilities, $p_\theta(\mathbf{r}_1 | \mathbf{x}, \langle \text{bos} \rangle)$. In preliminary experiments we found the first step’s probabilities to perform slightly better than using the last step’s probabilities.

B.3 Joint retrieval

We implement the joint retrieval model as a one-step variant of the autoregressive retrieval model,

$$p_\theta(\cdot | \mathbf{x}) = \text{softmax}(\mathbf{R}h_t + \mathbf{b}), \quad (7)$$

where $h_t \in \mathbb{R}^d$ is the final hidden state obtained from $h_{\theta_2}(\langle \text{bos} \rangle, f_{\theta_1}(\mathbf{x}))$, and f_{θ_1} , h_{θ_2} are implemented using the same encoder-decoder architecture as the autoregressive model (§B.2). This was a design decision made to closely compare the effect of autoregressive vs. joint parameterizations; an alternative implementation could use an encoder-only model.

The model is trained using KL-divergence loss, using per-example reference-distributions

$$p_*(\mathbf{r} | \mathbf{x}, \mathbf{y}) = \begin{cases} \frac{1}{|\mathbf{y}|} & \mathbf{r} \in \mathbf{y} \\ 0 & \text{otherwise} \end{cases},$$

where $\mathbf{y} = \{\mathbf{r}_1, \dots, \mathbf{r}_{|\mathbf{y}|}\}$ is the ground-truth reference set.

We use the same training settings that were used with the autoregressive model (§B.2).

B.4 Retrieval Metrics

For the mathematical reference retrieval task, we evaluate with standard retrieval metrics – mean average prevision (mAP) and recall@ k ($\text{R@}k$) – and a Full@ k metric that measures ability to fully recover all true references within the top- k results. We use $k = 10$ and $k = 100$ for our evaluation.

mAP. Suppose for retrieval example (\mathbf{x}, \mathbf{y}) the model ranks all references as $\mathbf{r}^{(1)}, \dots, \mathbf{r}^{(|\mathcal{R}|)}$. The average precision is computed as

$$\text{AP} = \sum_{j=1}^{|\mathcal{R}|} \mathbb{I}[\mathbf{r}^{(j)} \in \mathbf{y}] \frac{\sum_{k=1}^j \mathbb{I}[\mathbf{r}^{(k)} \in \mathbf{y}]}{j}.$$

mAP is the mean of AP across all retrieval examples.

R@k. For each retrieval example, the recall@k is

$$R@k = \frac{\sum_{j=1}^k \mathbb{I}[\mathbf{r}^{(j)} \in \mathbf{y}]}{|\mathbf{y}|}.$$

We aggregate recall@k by micro-averaging across retrieval examples.

Full@k. For each retrieval example, the fully-recovering indicator is formally defined as

$$\text{Full@k} = \prod_{\mathbf{r} \in \mathbf{y}} \mathbb{I}[\mathbf{r} \in \{\mathbf{r}^{(j)} \mid 1 \leq j \leq k\}].$$

The overall Full@k metric is thus the mean of this fully-recovering indicator across all retrieval examples.

C Additional Results

	ProofWiki					Stacks				
	mAP	R@10	R@100	Full@10	Full@100	mAP	R@10	R@100	Full@10	Full@100
Random	0.04	0.00	0.33	0.00	0.00	0.08	0.10	0.43	0.00	0.13
Frequency	3.54	5.99	24.44	0.88	2.28	1.03	1.86	10.86	0.13	2.19
TF-IDF	6.33	10.31	21.82	4.74	8.69	13.45	24.95	48.24	19.61	36.77
BERT-pair (P+S)	13.84	19.31	56.99	8.60	31.96	17.29	33.29	74.14	23.61	63.23
+joint	33.85	37.15	72.25	17.12	48.46	25.12	36.00	74.24	27.35	64.13
BERT-pair	16.99	22.91	62.03	9.22	36.96	21.21	38.00	75.67	28.77	66.19
+joint	37.51	41.39	75.92	20.54	50.75	26.55	39.81	75.71	30.58	66.06

Table 16: *In-domain* performance on the mathematical reference retrieval task (validation set). **BERT** is finetuned on the part of dataset with the same source as the evaluation set, whereas **BERT (P+S)** is finetuned on the combined dataset from ProofWiki and Stacks sources. Recall is micro-averaged.

	ProofWiki				Stacks			
	All	Theorems	Definitions	Others	All	Theorems	Definitions	Others
Frequency	3.54	7.25	5.02	1.49	1.03	1.14	0.33	0.48
TF-IDF	6.33	10.07	2.33	2.19	13.45	12.11	15.51	13.94
BERT	16.99	14.71	13.39	11.06	21.21	19.31	24.39	17.10

Table 17: Retrieval performance (mAP) by reference type (validation set).

Performance by reference type. In Table 17 we break down the in-domain retrieval performance by reference type. BERT shows a consistent improvement over TF-IDF on all types of references. On ProofWiki, TF-IDF does much worse on definitions and other types than on theorems, whereas BERT gives a more balanced performance on different types of references.

D Supplementary Materials

Dataset documentation and intended uses. We use the Dataset Nutrition Labels framework [17] for dataset documentation. For the Statistics module, please refer to Table 3, Figure 5 and Figure 6.

The NATURALPROOFS dataset is intended to be used by researchers to build or evaluate machines on predicting references in proofs, generating proofs to mathematical theorems, or other related tasks. It should not be regarded as source of truth for defining particular mathematical concepts, proving particular mathematical theorems, or the existence of such proof(s). In that case the user is advised to consult authoritative mathematical resources.

Metadata	
Filename	proofwiki.json stacks.json ra-trench.json nt-stein.json
Format	json
Url	https://doi.org/10.5281/zenodo.4632538
Domain	natural language processing
Keywords	mathematics, theorems, proofs, language
Type	
Rows	80,795
Columns	9
Missing	none
License	CC BY-SA 4.0 (proofwiki.json) CC BY-NC-SA 4.0 (ra-trench.json) GFDL 1.2 (stacks.json) MIT License (ra-stein script)
Released	June 2021
Range	N/A
Description	This dataset is a collection of mathematical statements and proofs in natural language. It collects data from multiple sources, encompassing broad-coverage of all math topics, deep-dive with a selected topic, and low-resource scenarios. The dataset provides theorems, proof(s) to each theorem when applicable, and in-proof references to other mathematical statements.

Provenance	
Source	
ProofWiki	(https://proofwiki.org/)
Stacks	(https://stacks.math.columbia.edu/)
Textbook: Real Analysis	(https://digitalcommons.trinity.edu/mono/7/)
Textbook: Number Theory	(https://wstein.org/ent/)
Author	
Name	Sean Welleck et al.
Email	wellecks@uw.edu

Variables	
id	A unique ID for this statement.
type	The type of this statement; either <i>theorem</i> , <i>definition</i> , or <i>other</i> .
label	A string description of this statement.
categories	A list of topics that this statement pertains. For ProofWiki data only.
title	A descriptive title of this statement.
contents	The content of this statement or proof, written in \LaTeX .
refs	A list of labels of statements that this statement or proof refers to in its content.
ref_ids	IDs for items in refs.
proofs	A list of proofs for this theorem. May be empty.

Table 18: Dataset Nutrition Labels for NATURALPROOFS.

Dataset URL. The NATURALPROOFS dataset is hosted at <https://doi.org/10.5281/zenodo.4632538>. Additional instructions and resources are provided in the Github repo <https://github.com/wellecks/naturalproofs>.

Author statement and license. We bear all responsibility in case of violation of rights. We confirm that the data sources we use are licensed to permit redistribution with modification for non-commercial purposes.

Hosting, licensing, and maintenance plan. The dataset is hosted and maintained through Zenodo [10],¹¹ and the code is hosted by GitHub. The code is released under the MIT license. The dataset is released under per-file licenses: CC BY-SA 4.0 (proofwiki.json), CC BY-NC-SA 4.0 (ra-trench.json), GFDL 1.2 (stacks.json), MIT License (ra-stein script). Zenodo meta-data is openly available under the CC0 license, and all open content is openly accessible through open APIs.¹²

Links to access the dataset and its metadata. The NATURALPROOFS dataset is hosted at <https://doi.org/10.5281/zenodo.4632538>. Additional instructions and resources are provided in the Github repo <https://github.com/wellecks/naturalproofs>.

¹¹<https://zenodo.org/>

¹²<https://about.zenodo.org/>

Data format. We store the dataset as JSON files. The dataset can be read using common JSON libraries (e.g. the built-in `json` module in Python) and following the dataset schema in Figure 3.

Long-term preservation. We ensure this by uploading the dataset to the Zenodo dataset repository.

Explicit license. The code is released under the MIT license. The dataset is released under per-file licenses: CC BY-SA 4.0 (`proofwiki.json`), CC BY-NC-SA 4.0 (`ra-trench.json`), GFDL 1.2 (`stacks.json`), MIT License (`ra-stein script`). Zenodo meta-data is openly available under the CC0 license, and all open content is openly accessible through open APIs.

Structured metadata. We release the metadata along with the dataset on Zenodo.

Persistent dereferenceable identifier. <https://doi.org/10.5281/zenodo.4632538>.

Reproducibility. We ensure this by releasing our code on GitHub, which includes instructions to reproduce the evaluation numbers in the paper.