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 $f : X \to S$ be locally of finite type with X quasi-compact. Then size$(X) \leq$ size(S).
Proof	We can find a finite affine open covering $X = \bigcup_{i=1,\ldots n} U_i$ 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 $\lvert A[x_1, \ldots, x_n]\rvert \leq \max\{\aleph_0, \lvert A\rvert\}$. 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 unique solution.
Proof	From Solution of Linear Congruence: Existence: the problem of finding all integers satisfying the linear congruence $ax \equiv b \pmod{n}$ is the same problem as: the problem of finding all the x values in the linear Diophantine equation $ax - ny = b$. Let: $\gcd\{a, n\} = 1$ Let $x = x_0, y = y_0$ be one solution to the linear Diophantine equation: $ax - ny = b$ From Solution of Linear Diophantine Equation, 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	1231.0	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
LaTeX env	**Type**
theorem	theorem
lemma	theorem
proposition	theorem
definition	definition
remark	other
remarks	other
proof	proof

Source	Textbook: RA
LaTeX env	**Type**
theorem	theorem
lemma	theorem
corollary	theorem
definition	definition
proof	proof

Source	Textbook: NT
LaTeX env	**Type**
theorem	theorem
lemma	theorem
corollary	theorem
proposition	theorem
definition	definition
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]] $\\left({S, \\circ}\\right)$.",
        "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]].",
        "{{qed}}",
        "[[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 $(\mathbf{x}, \mathbf{y})$ example is formed where $\mathbf{x}$ is the concatenation of `'title'` and `'contents'`, and $\mathbf{y}$ 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.

[8] It 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)

[9] `https://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.

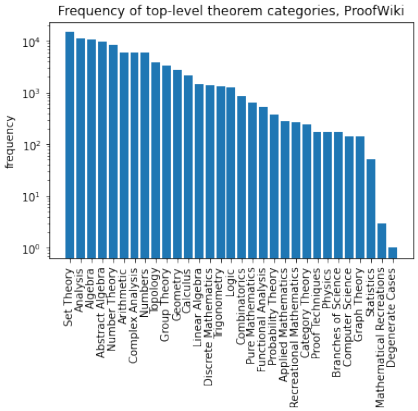

Figure 5: Frequency of top-level categories, ProofWiki.

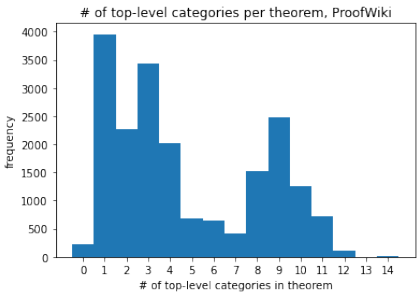

Figure 6: Number of top-level categories per theorem, ProofWiki.

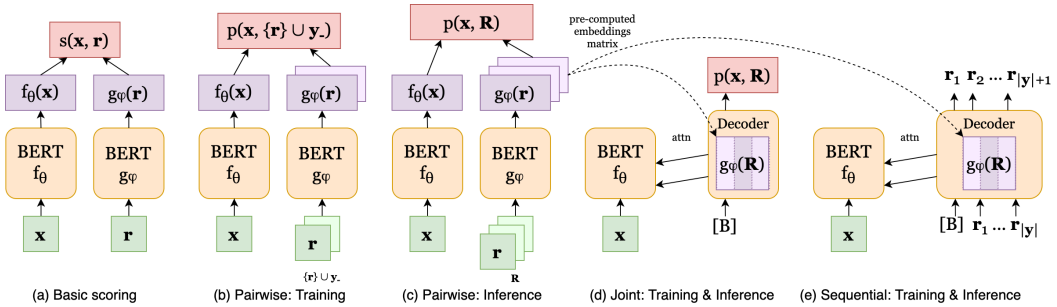

Figure 7: The pairwise, joint and sequential methods for mathematical reference retrieval.

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-lightning`[10]. The theorem encoder $f^{\text{thm}}_{\theta_1}$ is parameterized using the `bert-base-cased` architecture and initialized with its parameters. The reference encoder $g^{\text{ref}}_{\theta_2}$ 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 $\mathbf{S} = \mathbf{T}\mathbf{R}^\top \in \mathbb{R}^{B\times B}$ 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 $\langle bos\rangle$, $\langle eos\rangle$. At each step t, given a theorem and sequence of tokens $(\langle bos\rangle, \mathbf{r}_1, \ldots, \mathbf{r}_{t-1})$, 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 $\mathbf{y} = (\langle bos\rangle, \mathbf{r}_1, \ldots, \mathbf{r}_{|\mathbf{y}|}, \langle eos\rangle)$ is a reference sequence.

[10] https://github.com/PyTorchLightning/pytorch-lightning

Initialization. Let $f^{\text{thm}}_{\tilde{\theta}_1}$ and $g^{\text{ref}}_{\tilde{\theta}_2}$ 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. $\mathbf{R}$ means initializing and freezing the decoder's embedding matrix as (omitting the $\langle bos \rangle$ and $\langle eos \rangle$ rows),

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

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 $\hat{\mathbf{y}} = (\mathbf{r}_1, \ldots, \mathbf{r}_{|\hat{\mathbf{y}}|}, \langle eos \rangle)$ 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 $\mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(|\mathcal{R}|)}$ given an input $\mathbf{x}$. We evaluate our autoregressive model as a retrieval model by producing a ranked list $\mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(|\hat{\mathbf{y}}|)}, \ldots, \mathbf{r}^{(|\mathcal{R}|)}$, where the first $|\hat{\mathbf{y}}|$ references come from the model's generated sequence $\hat{\mathbf{y}} = (\mathbf{r}^{(1)}, \ldots, \mathbf{r}^{|\hat{\mathbf{y}}|})$ after removing duplicates, and the remaining references are ordered according to the model's first-step probabilities, $p_\theta(\mathbf{r}_1 | \mathbf{x}, \langle bos \rangle)$. 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,

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

where $h_t \in \mathbb{R}^d$ is the final hidden state obtained from $h_{\theta_2}(\langle bos \rangle, f_{\theta_1}(\mathbf{x}))$, 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 $\mathbf{y} = \{\mathbf{r}_1, \ldots, \mathbf{r}_{|\mathbf{y}|}\}$ 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 (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 $\mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(|\mathcal{R}|)}$. 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

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

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

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

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

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 *theorem*, *definition*, or *other*.
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 statament 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 metadata 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.