Constructing A Knowledge Graph of Advanced Mathematics

Anonymous ACL submission

Abstract

Knowledge graphs have been successfully applied in a variety of domains, including reasoning, question answering, and recommender systems. However, to date there are no large-scale knowledge graphs for all of advanced mathematics. In this paper, we propose constructing a system to automatically construct such a knowledge graph. Using a custom annotation software and a data-driven annotation scheme, we annotate *MathIEBench*, an annotated dataset for developing mathematical entity and relation extraction systems. We train strong baseline models for both tasks, and use these systems to construct *MathAtlas v1.0*, a knowledge graph constructed from 154 textbooks of advanced mathematics. Furthermore, we propose *MathAtlas* as a living knowledge graph, to be updated and curated as systems improve and new knowledge sources are acquired. Finally, we envision *MathAtlas* to be a "blueprint" (Massot, 2024) for all of mathematics (including at a research-level), a critical but missing tool for enabling development of autoformalization at an unprecedented scale.

1 Introduction

Knowledge graphs (KGs) have been applied in a variety of domains such as in reasoning (Liang et al., 2022; Lin et al., 2019; Zhang et al., 2017; Ding et al., 2019), question answering (Huang et al., 2019; Zheng et al., 2018; Dai et al., 2016; Mohammed et al., 2018), multimodal question-answering (Liang et al., 2021), and recommender systems (Zhang et al., 2016; Wang et al., 2018, 2019) among others (Ji et al., 2020). They provide structure to previously unstructured information, facilitating ease of search, improved automated reasoning, curation, and uniformity in representation among other benefits.

One domain which could benefit from such structure is modern mathematics, whose broad, deep, and highly interconnected information make knowledge graphs desirable. Along with previously mentioned applications, a knowledge graph would facilitate constructing a large and easily browsable "wiki" of mathematics, which would have applications in both education and research settings. Of particular note, such a knowledge graph would enable large-scale autoformalization of research level mathematics not only by improving model quality via retrieval augmentation (Liu et al., 2025), but importantly, by acting as a "blueprint" (i.e., dependency graph) for all of mathematics (Massot, 2024), allowing for a systematic approach to autoformalization at scale.

However, to be most broadly useful in these applications (especially for research mathematics), such a knowledge graph would need two attributes. First, it must be constructed from a wide variety of knowledge sources which cover all fields of mathematics, otherwise its usefulness is restricted to a particular setting. Secondly, when new math is published or sourced, the knowledge graph must be easily updatable *without additional human annotation*, which would otherwise prohibit the graph from staying current given the volume and complexity of modern math.

Previous mathematics knowledge graphs and databases lack one or both of these attributes. Online databases such as Wolfram MathWorld[1] or Wikipedia,[2] for example, are human constructed, making it difficult for them to stay current. On the other hand, KnowEdu (Chen et al., 2018) is automatically constructed, but its domain is geared towards education and is limited to grade-school topics, a largely simple and unchanging field.

In this paper, we propose an automatic construction of a large-scale mathematics knowledge graph consisting of extracted math entities and relations

[1] https://mathworld.wolfram.com/
[2] https://www.wikipedia.org/

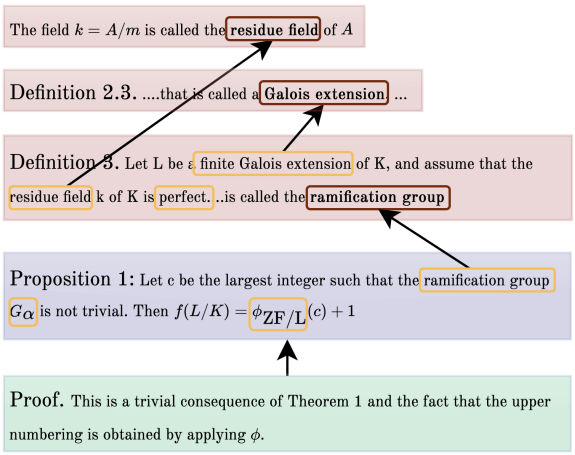

Figure 1: The structure of our annotated dataset *MathIEBench*. We annotate math items (definitions, theorems, proofs, and examples), together with identifiers (names in brown, references in yellow). Valid relations are shown with arrows (labels not shown).

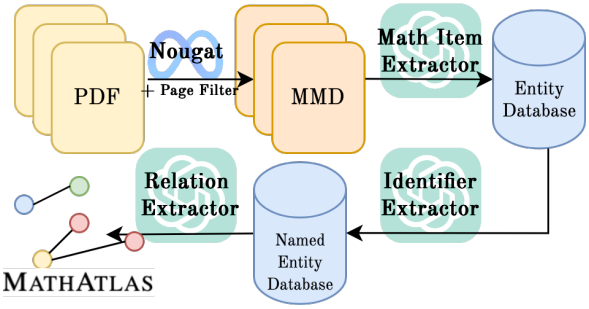

Figure 2: A demonstration of our KG construction pipeline. We collect mathematics texts (PDFs of textbooks or lecture notes) and process them with Nougat to obtain mathematical markdown (MMD), filtering out pages with only metadata (e.g., an index). Then, we extract math items (definitions, theorems, proofs, and examples) with our trained entity extraction model. Next, we pass the math items to an identifier extraction model to obtain names and references. Finally, our relation extraction system finds valid relations.

between them. To facilitate training such entity and relation extraction systems, we build a custom annotation tool, carefully develop an annotation scheme together with a team of expert annotators, and create an annotated training and benchmark dataset for each task sourced from textbooks spanning a diverse set of fields in mathematics. Together, we call these two annotated datasets ***MathIEBench***, which we release as a benchmark to enable developing mathematical entity and relation extraction systems.

Finally, we train baseline entity and relation extraction datasets, and run them on a collection of 154 diverse math texts to construct a knowledge graph which we call ***MathAtlas v1.0***. Currently, *MathAtlas v1.0* has over 169k extracted math items, and 800k relations. As it is constructed automatically, we propose *MathAtlas* to be a living knowledge graph, to be updated, improved, and curated continually as system improvements are made and new knowledge sources (e.g., papers or textbooks) are obtained. In this way, *MathAtlas* will remain up-to-date, diverse, and therefore useful for downstream tasks.

In summary, we make these contributions:

- We develop an annotation tool and a data-driven annotation scheme for extracting entities and relations from mathematical texts.
- We create an expert-annotated dataset (*MathIEBench*) of ∼1,400 extracted entities and ∼600 annotated relations from a variety of mathematical texts. We release all annotated data on GitHub as a training set and benchmark for future system development.
- We train baseline entity and relation extraction systems and use them to create *MathAtlas v1.0*, a first-of-its-kind automatically generated knowledge graph of advanced mathematics, containing over 169k math items, and 800k relations. We release *MathAtlas v1.0* on GitHub. Finally, we propose *MathAtlas* as a living knowledge graph, to be continually updated with system improvements and new sources.

The rest of the paper is organized as follows. First, we discuss related work (§ 2). Next, we discuss the structure and construction of *MathAtlas* (§ 3). We then present *MathIEBench* and discuss our annotation scheme and statistics (§ 4). Next, we discuss our baseline entity (§ 5) and relation extraction systems (§ 6), followed by our experimental setup and results (§ 7), and finally we conclude (§ 8).

2 Related Work

There is some previous work in knowledge graphs for mathematics (Zhao et al., 2019; de Paiva et al., 2023; Collard et al., 2022; Wang, 2022; Tomaszuk et al., 2023). Zhao et al. (2019) builds MathGraph, a knowledge graph specifically for solving high-school math problems together with an algorithm to solve the problem. However, the knowledge graph is small and manually created targeting only

their evaluation dataset.

Collard et al. (2022) perform entity extraction from sentences found in abstracts and on nLab[3] in the domain of category theory as a first step to build a knowledge graph. Similarly, de Paiva et al. (2023) apply LLMs to the task of term extraction in the same domain, finding they are effective but not at human levels. Both of these works are similar to one of our subtasks (identifier extraction) but is on a much smaller scale, more limited domain, and doesn't make a full knowledge graph. Collard et al. (2024) create a dataset of parsed mathematical sentences which can be used to improve performance in mathematical language processing tasks, including entity and relation extraction.

Wang (2022) build an automatic pipeline to generate a mathematics knowledge graph from Chinese web date. Our work differs in scale, domain, and purpose, as ours is much larger, has more advanced math, and is meant to be a living knowledge graph.

Tomaszuk et al. (2023) make a knowledge graph of advanced mathematics from the formal Mizar Mathematical Library (MML). However, MML manually curated and limited data source, making it ineffective for encompassing all of mathematics as it can't easily be updated from natural language mathematics. In contrast, our system operates on natural language mathematics (textbooks, notes, or research papers) and can therefore easily scale.

Finally, MaRDI (Schubotz et al., 2023; consortium, 2022) is an existing knowledge graph of advanced mathematics specifically tailored to multidisciplinary research. Our work differs in that it is fully automatically generated, which allows it to be scaled much larger.

3 *MathAtlas*

Here we present the structure of the knowledge graph *MathAtlas*, including entity types, relations, and some differences compared to traditional knowledge graphs. We discuss the KG construction pipeline and our plans for updating the graph.

Entities An **entity** (or **node**) refers to a vertex in the knowledge graph. We further subcategorize entities as either **math items** (definitions, theorems, examples, and proofs), or **identifiers** (names or references). Importantly, and different from traditional KGs, each entity in *MathAtlas* has an associated string and an associated span of text, which is the source text.

[3] https://ncatlab.org/nlab/show/HomePage

Relations A **relation** (or **link**) between entities is an edge in the knowledge graph. Entities can have zero or more incoming and outgoing edges. Additionally, relation types can be determined by the source and target entity types. We have the following relations: `proves` relates proofs to the theorem it proves. `nameof` relates names to the math item which it identifies. `refersto` relates references to the name(s) it refers to.

KG Construction To construct *MathAtlas*, we collect 154 PDFs of math texts from a variety of sources and all fields of mathematics including subfields of algebra, analysis, geometry, number theory, probability, quantum theory, category theory, topology, and others. Following our pipeline (Figure 2), we convert the PDFs to MMD files and filter out any metadata pages. Then, we run a math item extraction model (§ 5) on the MMDs, and feed the output math items to an identifier extractor model (§ 5). Finally, we run the rule-based candidate generator and relation extractor to generate relations (§ 6). See Appendix A for example extracted math items.

Updating *MathAtlas* We plan to update *MathAtlas* as better entity and relation extraction systems are developed in the future using our annotated dataset (*MathIEBench*, see below), by re-running our KG construction pipeline with the latest tools to obtain a new version of *MathAtlas*.

4 *MathIEBench*: Annotated Benchmark Dataset

We now discuss *MathIEBench*, our annotated entity and relation extraction datasets, including its formats, creation, and an overview of our annotation scheme. Our data is collected from a sample of mathematics textbooks and lecture notes, converted via Nougat (Blecher et al., 2023) from PDF to mathematical markdown (MMD).

4.1 Annotation Overview

Our goal for annotation is to acquire data to train and evaluate automatic entity and relation extraction systems. To ensure high quality annotations, all of our annotators are current undergraduate mathematics students or have mathematics degrees. We use a diverse sample of mathematics sources, including fields such as algebra, number theory, analysis, geometry, category theory, and probability, which ensures high coverage of mathematics.

We developed the annotation guidelines jointly with annotators over an iterative, data-driven process, to find annotation conventions that have a balance between being efficient to annotate, having maximum utility for downstream applications, and resulting in high inter-annotator agreement.

4.2 Entity Extraction Dataset

In this section, we present our entity extraction dataset and annotation scheme in detail.

Entity Extraction Statistics To make our extraction dataset, we annotate samples of 2-3 consecutive pages taken from 20 different texts spanning several areas of mathematics. In total, we annotate 731 extracted math items and 885 identifiers. A summary of statistics is shown in Table 1.

Entity Type	# Entities	Avg. # Tokens
definition	260	82
theorem	263	83
example	72	120
proof	136	299
name	625	4
reference	259	3

Table 1: Annotated dataset statistics by entity type. We use the GPT-4o-mini tokenizer to compute length.

4.2.1 Annotation Scheme

We identify and extract **math items**: definitions, theorems, proofs, and examples. Also, we annotate **identifiers**: names and references. In *MathIEBench*, all these are annotated as **spans** over the text.

Definitions A definition indicates a newly defined object and has an associated name (see Figure 4). Sometimes definitions will define multiple objects. For example, consider the following definition of open and closed sets: "Let A be a subset of $\mathbb{R}$. A is **closed** if . . ., and is **open** if . . .." The two definitions necessarily overlap (since "open" requires the context of A), in which case we use one definition annotation with multiple names.

Theorems and Proofs A theorem can be demarcated (e.g., in a "theorem" environment) or found in-line among other text. Lemmas, propositions, and conjectures are also considered theorems. They also often have an associated proof (see Figure 3. Note that a proof might be imprecise, such as "it follows that . . ." or ". . . by Lemma 3". Theorems may have associated names, but are not required to like definitions. Note that not every claim is considered a theorem, as some aren't rigorous enough. This can be difficult to discern, and we discuss ambiguous examples on a case-by-case basis.

Corollary 7.2._Let E be a finitely generated module and E' a submodule. Then E' is finitely generated._

Proof. We can represent E as a factor module of a free module F with a finite number of generators: If v_1,..., v_n are generators of E, we take a free module F with basis $\{x_1, \ldots, x_n\}$ and map x_i on v_i. The inverse image of E' in F is a submodule, which is free, and finitely generated, by the theorem. Hence E' is finitely generated.

Figure 3: Example of a theorem and associated proof taken from *MathAtlas*. In this case, the theorem has a name (Corollary 7.2) and several references to previously defined terms.

Examples We also annotation examples as they are occasionally referenced by future entities. Often times, an example is also technically a theorem. For example, the statement "$\mathbb{R}$ together with addition forms a group" is a common example of a group, but is also a provable theorem. In most cases, the context allows us to distinguish between the two, and we hold case-by-case discussion for ambiguous examples. Like theorems, examples can have names but are not required to.

Names A name is an identifier for a newly defined entity. Every definition has a name (Figure 4), and some theorems and examples will too (Figure 3). Occasionally, there can be multiple names for a single object, such as for alternative or shorthand notation (Figure 4b). In this case, we link the first name to the object, the second name to the first name, and so on. We only annotate names within other entities, extending the parent entity to encompass its name if necessary.

References While names identify new definitions or theorems, a reference is an identifier which references a previously defined object. For example, in the sentence "Let G be an abelian group", the phrase "abelian group" is a reference to the previously defined object. Note that we do not split up references by individual words. Instead, a single reference can contain multiple words, and can

By $S \subset \mathbb{N}$ having a _least element_, we mean that there exists an $x \in S$, such that for every $y \in S$, we have $x \leq y$.

(a) Example of a definition of "least element". We annotate "least element" as the name and add a relation between it and the definition (the whole span).

Definition 0.3.1.: A _set_ is a collection of objects called _elements_ or _members_.

(b) In this example, there are multiple objects defined, namely "set" and "elements". We add a relation from set and elements to the definition since theyware the names of the defined objects. In contrast, members is linked to elements since it is just an alternative name for the same object.

Figure 4: Example definition annotations.

Example 0.3.3.: The following are sets including the standard notations.

1. The set of _natural numbers_, $\mathbb{N} := \{1, 2, 3, \ldots\}$.
2. The set of _integers_, $\mathbb{Z} := \{0, -1, 1, -2, 2, \ldots\}$.
3. The set of _rational numbers_, $\mathbb{Q} := \left\{\frac{m}{n} : m, n \in \mathbb{Z} \text{ and } n \neq 0\right\}$.
4. The set of even natural numbers, $\{2m : m \in \mathbb{N}\}$.
5. The set of real numbers, $\mathbb{R}$.

Figure 5: An example of a clearly demarcated example annotation containing several examples of sets.

likewise have a relation to multiple names (e.g., "abelian" and "group"). As with names, we only annotate references within other entities. Additionally, we don't annotate references to objects defined within the same object. Finally, to save time during annotation, we don't annotate references to operators (e.g., $\oplus$) once it has been defined.

Minimal Spans Our goal is for each extracted entity to be self-contained, but carry all necessary information. Thus, our primary guiding principle for annotation is that we annotate the *smallest contiguous span* that communicates the math entity in question (see Figure 6). We include environment indicators: in the span "Theorem 1.2: Let g be a group. . ." for example, we include "Theorem 1.2:" as part of the theorem. Similarly, we include in-line indicators of definitions (e.g., "let", "define", "we call").

Nested Spans Furthermore, entities can be nested or partially overlapping. For example, consider the following theorem: "Let g be a group. Then, by Lemma 3.5, we have . . .." The theorem's proof is "by Lemma 3.5" which is nested in the theorem.

. . . that is, one vector cannot be a scalar multiple of the other. A **lattice** of $\mathbf{x}$ and $\mathbf{y}$ is the set of . . .

. . . that is, one vector cannot be a scalar multiple of the other. A **lattice** of $\mathbf{x}$ and $\mathbf{y}$ is the set of . . .

Figure 6: An example of an incorrect annotation (top) and correct annotation (bottom) of the name "lattice". Note that we exclude the "**" to ensure only the *smallest necessary span* is annotated.

4.2.2 Annotation Quality

We compute token-level F1 score compared against a reference annotator to measure annotation quality following previous work (Jiang et al., 2022; Brandsen et al., 2020), as Cohen's Kappa (Cohen, 1968) is not suitable for NER tasks as it requires the number of negative cases. We found high agreement, with a 93% token-level F1 computed against a reference annotator. At the end, one annotator normalized the annotations and fixed any errors.

4.3 Relation Extraction Dataset

We also annotate a dataset for relation extraction. To acquire entities to annotate, we use our math item extraction and identifier extraction systems trained on the above annotated extraction dataset. We run our math item and identifier extraction models over 154 textbooks to create a database of named entities and references, then use our rule-based candidate generator (see § 6) to generate candidate relations, and finally we randomly sample 618 candidate relations.

We annotate whether each relation is valid or invalid. For a relation to be valid, its source must be mathematically equivalent to the target. For example, a relations from "...normal distribution" to "...normal subgroup" would be invalid despite having the same identifier since they are referring to different objects. In contrast, a relations from "...abelian group" to "...a group is abelian" would be valid despite only being partially represented. Of the 618 relations in this dataset, 464 (75%) are invalid and 154 (25%) are valid.

4.4 Annotation Tool

We develop a custom annotation tool designed for our purpose. In particular, our tool needed to render PDFs and text side-by-side, handle long (500+ page) documents, thousands of annotations per document, overlapping and nested annotations, and long-distance and cross-document relations. Our tool will be released on GitHub, and can be easily modified for a number of tasks. See Figure 13 for an example screenshot.

5 Automatic Entity Extraction

We now present our math entity extraction system. At a high level, we first convert PDFs into mathematical markdown (MMD), a superset of Markdown compatible with math written in LaTeX. Next, we filter pages containing metadata (e.g., table of contents or index pages). Finally, we run a fine-tuned math item extraction model to create a database, and our fine-tuned identifier extraction model on the extracted math items to add names and references.

PDF to MMD To convert PDFs to MMDs, we use Nougat (Blecher et al., 2023) (see Figure 8).[4] Nougat can rarely misprocess a page resulting in it being omitted, which we simply ignore and filter from our dataset (in total, 0.1% of pages fail to process). Additionally, we train a binary page filter by annotating 130 randomly sampled pages on whether they should be processed or skipped (see Appendix B). Pages consisting of exclusively metadata (table of contents, index, etc.) are marked invalid, while all others are marked valid. We then filter out metadata pages to prevent our entity extractor from finding false positives.

Math Item Extraction We developed a variety of math item extraction models and tested their performance (§ 7 and Table 5). Our best math item extraction model is a fine-tuned GPT-4o-mini trained on our annotated data and tested on a held-out set. The input format is the instruction followed by the input MMD text, and the model output a single JSON with entity types as keys and lists of extracted entities as values (Figure 7). To ensure valid JSON output, we use structured decoding[5] (see Figure 11).

[4] One limitation of Nougat is its inability to handle figures, which we leave to future work.

[5] https://platform.openai.com/docs/guides/structured-outputs

Identifier Extraction We developed two fine-tuned identifier extraction models and tested their performance (§ 7 and Table 6). Our identifier extraction model's input is a math item's text and it outputs the names and references contained in that item. As before, we fine-tune GPT-4o-mini on annotated identifier data and test on the remaining held-out set. However, since identifiers are short and often appear multiple times in a definition or theorem, the JSON output format we use previously has ambiguity, since we can't differentiate between one instancewof a word and another. Instead, we use XML as the expected output format (see Figure 12). Finally, if a definition has no associated name, we filter it out since it's impossible to reference.

6 Automatic Relation Extraction

For relation extraction, we follow the high-level approach from Bansal et al. (2019), altering details to better fit our problem. We use two steps. First, we generate candidates using a rule-based system to find matching names $N = \{n_1, \ldots, n_k\}$ given a particular reference $r \in R$. Then, we use a binary classifier $L : N \times R \to \{0, 1\}$ to determine which n_i are valid targets for R. The resulting set $N' = \{n_i : L(n_i, r) = 1\}$ are the final relations. This approach sidesteps the need for finding andwannotating valid relations, which is a highly non-trivial task since there can be multiple valid relations across multiple textbooks.

We now describe our relation extraction system in detail.

Candidate Generator The candidate generator is a rule-based system which, for each reference r, generates a list of possible candidate targets (Algorithm 1). First, it checks for an exact match ignoring pluralization ("s" or "es" endings), adding all such matches to the list of candidates. Secondly, we also attempt to match individual tokens of the reference. This step improves recall for the cases where a reference consists of several modifiers followed by a noun (e.g., "transcendental entire function") but the definitions are separate ("a function is **entire** when. . ."). This case may require multiple necessary relations to be considered "complete."

Relation Extractor The relation extraction model is a binary classifier trained to identify valid relations given the reference, the math item its found in, the candidate name, and its math item.

```
In the following LaTeX, extract entities of the following types and return them in JSON output.
...
We give the first truly striking result about cardinality. To do so$\ldots$
**Definition 0.3.33**.: The \_power set\_ of a set \(A\), ...
For example, if \(A\coloneqq\{1,2\}\), then ...
**Theorem 0.3.34** (Cantor*).: \_Let \(A\) be a set. Then ...
```

```
{ "definition": ["**Definition 0.3.33**.: The _power set_..."],
  "theorem": ["**Theorem 0.3.34** (Cantor*).: _Let \(A\) be a set ..."],
  "proof": [],
  "example": ["if \(A\coloneqq\{1,2\}\), then "]}
```

Figure 7: An example input/output of our math item extraction model. We supply an instruction to output JSON (which is constrained by structured decoding) together with the input snippet. Our output is a JSON whose values are lists of strings corresponding to the predicted annotations. See Figure 10 for the full prompt.

Algorithm 1 Rule-Based Candidate Generator

```
procedure FindCandidates(r, N)
    candidates ← {}
    for n ← N do
        tokens ← tokenize(r)
        if matches(n, r) then
            append(n, candidates)
        end if
        for t ← tokens do
            if matches(n, t) then
                append(n, candidates)
            end if
        end for
    end for
    return candidates
end procedure
```

Specifically, we fine-tune GPT-4o-mini on our annotated relation extraction dataset and test on a held-out set.

7 Experiments and Results

Here we present our entity and relation extraction experiments and results, as well as a human evaluation of *MathAtlas*.

Math Item and Identifier Extraction For extraction, we train two systems: our math item extraction model which identifies definitions, theorems, proofs, and examples, and our identifier extraction model, which identifies names and references present within entities. We test few-shot prompting of GPT-3.5, GPT-4, and GPT-4o-mini, fine-tuning BERT-CRF (Devlin et al., 2019), and fine-tuning GPT-4o-mini. For hyperparameters, see Appendix B. GPT models are asked to output JSON as shown in Figure 7 (see Figure 11 for the structured output schema). The BERT-CRF model is trained as a BIO-tagger, that is, on token-level multi-class classification, with hidden states passed to a neural CRF. Each fine-tuned GPT-4o-mini model costs approximately $6 to train. All training and inference code will be released on GitHub.

We train our baseline math item extraction models on 48 annotation sets (629 math items), validate on 3 sets (46 math items), and test on 3 sets (56 math items). In each case, we find that fine-tuning GPT-4o-mini is by far the most effective system, with our best model achieving a token-level F1 score of 76.2% (Table 5). In particular, we find that few-shot prompting can identify obvious examples (demarcated theorems and definitions) but miss in-line entities and overlapping entities. The BERT-based models performed better, but suffered from overfitting. In contrast, fine-tuning GPT-4o-mini showed high performance even for more challenging examples.

For our baseline identifier model, we train a BERT-based tagger and a fine-tuned GPT-4o-mini trained to output XML. Our annotated dataset consists of 900 names and 200 references. We first train the model on 725 of name annotations, and fine-tune on the remaining name and reference annotations, which we found improves performance. BERT again performed worse than GPT-4o-mini, which achieved an F1 scorewof 88.4 (Table 6).

Relation Extraction Given the lower performance of BERT from the previous experiments, we use GPT-4 again for entity linking. We fine-tune GPT-4o-mini on 463 annotated links, and test on

Entity Type	Count	#Named	Avg. Refs
definition	34k	34k	3.9
theorem	79k	45k	4.6
example	19k	8k	5.9
proof	37k	-	-

Table 2: Number of entities of each type, how many are named, and how many references are found in each on average. Recall we don't annotate within proofs.

Class	Precision	Recall	F1
Definition	93.2	64.0	75.8
Theorem	69.0	76.6	72.3
Example	79.6	21.7	34.1
Proof	94.6	77.6	85.2

Table 3: Class-wise results on the test set for our entity extraction model. Theorems and Examples are often mislabeled for each other, likely due to class imbalance.

the remaining 155 (Table 4). Our baseline model achieves an F1 score of 83.1 with a precision of 86.2 and recall of 80.0.

Human Evaluation of *MathAtlas* We perform a manual evaluation of 100 randomly selected entities (25 of each math item type) from *MathAtlas v1.0* together with their identifiers. In total, 84% of definitions, 96% of theorems, 68% of examples, and 80% of proofs were found to be correctly extracted, which indicates the baseline models generalize well to real-world data. Note that theorems and examples easily conflated. Furthermore, definitions and theorems had a median of 4 references each, and 90% of the extracted relations were valid. 35% of references had at least one valid relation. This indicates the relation extraction system has high precision but low recall. Improvements for the future can include both improving the system performance with better models and by adding more knowledge sources as potential relation targets. Finally, we measure how well the baseline model generalizes to fields outside its training data. The 82 correctly extracted entities spanned 26 broadly categorized fields of math, of which only 12 appeared in the training data. Of the 18 misextracted entities which spanned 12 fields, 5 of them appeared in the training data. These results suggest that the training data is sufficiently diverse to support robust model development.

8 Conclusion

In conclusion, we propose developing systems for creating a large-scale, automatically generated knowledge graph for advanced mathematics. To support this development, we build a custom annotation tool, carefully develop an annotation scheme together with a team of expert annotators, and annotate entity and relation extraction dataset *MathIEBench* from a diverse set of mathematics textbooks. We use this dataset to train baseline entity and relation extraction systems. We then run these systems on a collection of 154 math textbooks to generate *MathAtlas v1.0*, the first-of-its-kind knowledge graph of advanced mathematics. We propose *MathAtlas* as a living knowledge graph, to be continually updated and improved as system improvements are made and new knowledge sources become available. We believe *MathAtlas* can be an effective tool for a variety of domains such as mathematical reasoning, education, and providing a necessary launching point for large-scale autoformalization. Along with continual updates, we leave to future work the creation of search and visualization tools for *MathAtlas*.

Label	Precision	Recall	F1
Valid	74.1	51.2	84.0
Invalid	80.7	97.4	88.3

Table 4: Performance of our relation extractor on the test set.

Model	Precision	Recall	F1
Few-shot			
GPT-3.5	77.5	19.0	30.5
GPT-4	61.3	48.4	54.1
GPT-4o-mini	73.4	49.7	59.5
Fine-tuned			
BERT-CRF	67.0	**75.39**	70.96
GPT-4o-mini	**84.4**	69.5	**76.2**

Table 5: Test set performance for the entity extraction models. We find fine-tuning GPT-4o-mini has the best overall performance.

Model	Precision	Recall	F1
BERT-CRF	74.6	96.0	84.0
GPT-4o-mini	**81.0**	**97.3**	**88.4**

Table 6: Test set performance for identifier extraction models, all fine-tuned. We find GPT-4o-mini outperforms BERT.

Limitations

Our dataset is limited by the quality of the mathematics textbooks and lecture notes we extract our

knowledge from. Any errors will be propagated to our dataset. Also, as our systems are all reliant on proprietary models (GPT-4o-mini) with an undetermined end-of-life date, we may need to retrain them once the current ones are deprecated. Reproducing this work with an open-source and open-data model to avoid this problem is a future goal. Finally, we are also limited by the capabilities of Nougat, which makes mistakes and can lead to nonsense in the output MMD. It is also unable to handle figures, which can be important to some areas of mathematics.

References

Trapit Bansal, Pat Verga, Neha Choudhary, and A. McCallum. 2019. Simultaneously linking entities and extracting relations from biomedical text without mention-level supervision. *AAAI Conference on Artificial Intelligence*.

Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. 2023. Nougat: Neural optical understanding for academic documents. *Preprint*, arXiv:2308.13418.

Alex Brandsen, Suzan Verberne, Milco Wansleeben, and Karsten Lambers. 2020. Creating a dataset for named entity recognition in the archaeology domain. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4573–4577, Marseille, France. European Language Resources Association.

Penghe Chen, Yu Lu, Vincent W. Zheng, Xiyang Chen, and Boda Yang. 2018. Knowedu: A system to construct knowledge graph for education. *IEEE Access*, 6:31553–31563.

Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological Bulletin*, 70(4):213–220.

Jacob Collard, Valeria de Paiva, Brendan Fong, and Eswaran Subrahmanian. 2022. Extracting mathematical concepts from text. *arXiv preprint arXiv: 2208.13830*.

Jacob Collard, Valeria de Paiva, and Eswaran Subrahmanian. 2024. Mathematical entities: Corpora and benchmarks. *International Conference on Language Resources and Evaluation*.

The MaRDI consortium. 2022. Mardi: Mathematical research data initiative proposal.

Zihang Dai, Lei Li, and Wei Xu. 2016. CFO: Conditional focused neural question answering with large-scale knowledge bases. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 800–810, Berlin, Germany. Association for Computational Linguistics.

Valeria de Paiva, Qiyue Gao, Pavel Kovalev, and Lawrence S. Moss. 2023. Extracting mathematical concepts with large language models. *arXiv preprint arXiv: 2309.00642*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *North American Chapter of the Association for Computational Linguistics*.

Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. 2019. Cognitive graph for multi-hop reading comprehension at scale. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2694–2703, Florence, Italy. Association for Computational Linguistics.

Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. 2019. Knowledge graph embedding based question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, page 105–113, New York, NY, USA. Association for Computing Machinery.

Shaoxiong Ji, Shirui Pan, E. Cambria, P. Marttinen, and Philip S. Yu. 2020. A survey on knowledge graphs: Representation, acquisition and applications. *IEEE Transactions on Neural Networks and Learning Systems*.

Hang Jiang, Yining Hua, Doug Beeferman, and Deb Roy. 2022. Annotating the Tweebank corpus on named entity recognition and building NLP models for social media analysis. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 7199–7208, Marseille, France. European Language Resources Association.

K. Liang, Lingyuan Meng, Meng Liu, Yue Liu, Wenxuan Tu, Siwei Wang, Sihang Zhou, Xinwang Liu, and Fu Sun. 2022. A survey of knowledge graph reasoning on graph types: Static, dynamic, and multimodal. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Weixin Liang, Yanhao Jiang, and Zixuan Liu. 2021. Graghvqa: Language-guided graph neural networks for graph-based visual question answering. *MAI-WORKSHOP*.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. Kagnet: Knowledge-aware graph networks for commonsense reasoning. *Conference on Empirical Methods in Natural Language Processing*.

Qi Liu, Xinhao Zheng, Xudong Lu, Qinxiang Cao, and Junchi Yan. 2025. Rethinking and improving autoformalization: towards a faithful metric and a dependency retrieval-based approach. In *The Thirteenth International Conference on Learning Representations*.

Patrick Massot. 2024. Lean blueprint. `https://github.com/PatrickMassot/leanblueprint`. Accessed: 2024-12-14.

Salman Mohammed, Peng Shi, and Jimmy Lin. 2018. Strong baselines for simple question answering over knowledge graphs with and without neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 291–296, New Orleans, Louisiana. Association for Computational Linguistics.

Moritz Schubotz, Eloi Ferrer, Johannes Stegmüller, Daniel Mietchen, Olaf Teschke, Larissa Pusch, and Tim OF Conrad. 2023. Bravo mardi: A wikibase powered knowledge graph on mathematics. *arXiv preprint arXiv: 2309.11484*.

Dominik Tomaszuk, Łukasz Szeremeta, and Artur Korniłowicz. 2023. MMLKG: Knowledge Graph for Mathematical Definitions, Statements and Proofs. *Scientific Data*, 10(1):791.

Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. Dkn: Deep knowledge-aware network for news recommendation. *arXiv preprint arXiv: 1801.08284*.

Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-task feature learning for knowledge graph enhanced recommendation. *arXiv preprint arXiv: 1901.08907*.

Jianing Wang. 2022. Math-kg: Construction and applications of mathematical knowledge graph. *arXiv preprint arXiv: 2205.03772*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv: 1910.03771*.

Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 353–362, New York, NY, USA. Association for Computing Machinery.

Yuyu Zhang, H. Dai, Zornitsa Kozareva, Alex Smola, and Le Song. 2017. Variational reasoning for question answering with knowledge graph. *AAAI Conference on Artificial Intelligence*.

Tianyu Zhao, Yan Huang, Songfan Yang, Yuyu Luo, Jianhua Feng, Yong Wang, Haitao Yuan, Kang Pan, Kaiyu Li, Haoda Li, and Fu Zhu. 2019. Mathgraph: A knowledge graph for automatically solving mathematical exercises. In *International Conference on Database Systems for Advanced Applications*.

Weiguo Zheng, Jeffrey Xu Yu, Lei Zou, and Hong Cheng. 2018. Question answering over knowledge graphs: question understanding via template decomposition. *Proc. VLDB Endow.*, 11(11):1373–1386.

A Examples from *MathAtlas*

Example definitions with highlighted names taken from *MathAtlas*:

11.3.1 Definition.: Let (M, d) and (N, ρ) be metric spaces. A function $f\colon M \to N$ is uniformly continuous if for every $\epsilon > 0$ there exists $\delta > 0$ such that $\rho(f(x), f(y)) < \epsilon$ whenever $x, y \in M$ and $d(x, y) < \delta$.uniformly

The number of bits (0's or 1's) in the string is the **length** of the string

For each graded module M and each integer $m \in \mathbb{Z}$ there is graded module $M(m)$ associated to a graded module M. The shift do not alter the module structure of M, not even the sets of homogeneous elements is affected, but they are given new degrees. The new degrees are defined by setting

$$M(m)_d = M_{m+d}.$$

In other words, one declares the degree of the elements in M_m to be equal to $d - m$

Example theorems taken from *MathAtlas*:

If R is an ordered ring which is an Ore domain, show that the ordering can be extended in a unique way to the field of fractions of R.

Corollary (8.10). – Let R, R' be rings, N a bimodule. Then the functor $\bullet \otimes_R N$ preserves direct limits, or equivalently, direct sums and cokernels.

the symmetric algebra $S(M)$ has as basis the set of monomials ε^a with $\alpha \in \mathbf{N}^n$ (SS 6, no. 6, Theorem 1) and $S(M*)$ the set of monomials ε^{a*} with $\alpha \in \mathbf{N}^n$; recall (no. 5) that u_a for $|\alpha| = k$, denotes the element of the basis of $(S^k(M))^*$, dual to the basis $(\varepsilon^a)_{|\alpha|=k}$ of $S^k(M)$; the u_a, for $\alpha \in \mathbf{N}^n$, therefore form a basis of $S(M)^*$.

Example example from *MathAtlas*

Example 2.3.14.: **The category of tangles.** Let $S_{m,n}$ be the disjoint union of m circles $\mathbb{R}/\mathbb{Z}$ and n intervals $[0, 1]$. A _tangle_ is a smooth embedding $f : S_{m,n} \to \mathbb{R}^2 \times [0, 1]$ such that the boundary maps to the boundary and the interior to the interior. We will abuse the terminology by also using the term "tangle" for the image of f.
Let x, y, z be the Cartesian coordinates on $\mathbb{R}^2 \times [0, 1]$. Any tangle has inputs (points of the image of f with $z = 0$) and outputs (points of the image of f with $z = 1$). For any integers $p, q \geq 0$, let $\widetilde{T}_{p,q}$ be the set of all tangles which have p inputs and q outputs, all having a vanishing y-coordinate. Let $T_{p,q}$ be the set of isotopy classes of elements of $\widetilde{T}_{p,q}$; thus, during an isotopy, the inputs and outputs are allowed to move (preserving the condition $y = 0$), but cannot meet each other. We can define a canonical composition map $T_{p,q} \times T_{q,r} \to T_{p,r}$, induced by the concatenation of tangles. Namely, if $s \in T_{p,q}$ and $t \in T_{q,r}$, we pick representatives $\widetilde{s} \in \widetilde{T}_{p,q}, \widetilde{t} \in \widetilde{T}_{q,r}$ such that the inputs of $\widetilde{t}$ coincide with the outputs of $\widetilde{s}$, concatenate them, perform an appropriate reparametrization, and rescale $z \to z/2$. The obtained tangle represents the desired composition ts.

B Experimental Details

When training our BERT models, we found they performed better when treating overlapping and nested entities as a new class rather than as multiple labels. For example, an overlapping theorem and definition would be considered a "theorem-definition." This also allows for simple addition of a CRF.

All GPT models were fine-tuned with default parameters and were used with temperature 0.0. We ran 100 trials of hyperparameter search for our BERT models, and the best hyperparameters found were a learning rate of 0.000144, batch size of 8, label smoothing of 0.1, and 6% warmup with an inverse square root decay over 3000 updates. We used huggingface's transformers library for all local training and inference (Wolf et al., 2019).

Our page filter is also based on GPT-4o-mini and is trained similarly. We randomly sample 130 pages and annotate them as either "missing" for pages Nougat failed to process, "metadata" for pages consisting of only metadata such as an index, and "math" for pages we want to annotate. We keep only "math" pages. This model achieves 95% accuracy on a heldout test set.

Definition 0.3.21. Let $\mathcal{R}$ be a relation on a set A. Then $\mathcal{R}$ is said to be

(i) *Reflexive* if $a \mathcal{R} a$ for all $a \in A$.

(ii) *Symmetric* if $a \mathcal{R} b$ implies $b \mathcal{R} a$.

(iii) *Transitive* if $a \mathcal{R} b$ and $b \mathcal{R} c$ implies $a \mathcal{R} c$.

If $\mathcal{R}$ is reflexive, symmetric, and transitive, then it is said to be an *equivalence relation*.

```
**Definition 0.3.21**.: Let \(\mathcal{R}\) be a relation on a set \(A\).
Then \(\mathcal{R}\) is said to be

1. _Reflexive_ if \(a\mathcal{R}\)\(a\) for all \(a\in A\).
2. _Symmetric_ if \(a\mathcal{R}\)\(b\) implies \(b\mathcal{R}\)\(a\).
3. _Transitive_ if \(a\mathcal{R}\)\(b\) and \(b\mathcal{R}\)\(c\) implies
\(a\mathcal{R}\)\(c\).

If \(\mathcal{R}\) is reflexive, symmetric, and transitive, then it is said
to be an _equivalence relation_.
```

Figure 8: A snippet of the PDF of the free online textbook *Basic Analysis I* by Jiří Lebl, together with Nougat's generated output.

30.3.16 Proposition.: In a C^*-algebra involution is an isometry. That is, $\|a^*\| = \|a\|$ for every element a of the algebra. w

For $k = 1, 2$ let V_k be a normed linear space, ... Then f is an isometry (or an isometric map) if ....

A function $f\colon S \to S$ from a set to itself is an involution if it is its own inverse; that is if $f(f(s)) = s$ for all $s \in S$.

Figure 9: A randomly sampled entity from *MathAtlas* with two references, "C^*-algebra involution" and "isometry", together with the named definitions they are linked to.

C Relation Extraction Candidate Generator

In Algorithm 1, we describe our candidate generation algorithm. In our implementation, `matches` is defined as follows:

```
def matches(name: str, ref: str):
    # 1. Check exact match
    # 2. Check plurals
    if ref == name \
      or ref == name + "s" \
      or ref == name + "es":
        return True

    # Do same with name/ref reversed
    if name == ref \
      or name == ref + "s" \
      or name == ref + "es":
        return True

    # Finally, check ies -> y conversion
    if ref.endswith("ies"):
        if ref[:-3] + "y" == name:
            return True

    # Otherwise no match
    return False
```

```
In the following LaTeX, extract entities of the following types and return them in JSON output.
1. definition
2. theorem
3. proof
4. example
Your output should be a single JSON with 4 keys corresponding to the 4 entity types
above. Spans may be part of multiple entities. Do not hallucinate text.
```

Figure 10: Full entity extraction prompt for GPT models.

```
{
    "type": "json_schema",
    "json_schema": {
        "name": "entity_extraction_response",
        "schema": {
            "type": "object",
            "properties": {
                "definition": {
                    "type": "array",
                    "items": {
                        "type": "string"
                    }
                },
                "theorem": {
                    "type": "array",
                    "items": {
                        "type": "string"
                    }
                },
                "example": {
                    "type": "array",
                    "items": {
                        "type": "string"
                    }
                },
                "proof": {
                    "type": "array",
                    "items": {
                        "type": "string"
k                    }                }
            },
            "required": ["definition", "theorem", "example", "proof"],
            "additionalProperties": false
        },
        "strict": true
    }
}
```

Figure 11: The JSON schema used for structured output with the OpenAI GPT models.

```
In the following LaTeX snippet, extract every newly defined named entity (name) and reference to
previously defined named entity (reference). Your output should be in XML.

**Definition 7.6.1**.: Let \((X,d_{X})\) and \((Y,d_{Y})\) be metric spaces. A map
\(\varphi\colon X\to Y\) is a _contraction_ (or a contractive map) if it is a \(k\)-Lipschitz map for
some \(k<1\)...
```

```
**<name>Definition 7.6.1</name>**.: Let \((X,d_{X})\) and \((Y,d_{Y})\) be <ref>metric spaces</ref>.
A map \(\varphi\colon X\to Y\) is a _<name>contraction </name>_ (or a <name>contractive map</name>)
if it is a <ref>\(k\)-Lipschitz map </ref> for some \(k<1\)...
```

Figure 12: An example input/output of our identifier model. We supply an instruction along with the target snippet of MMD text, and expect XML output containing name and reference tags.

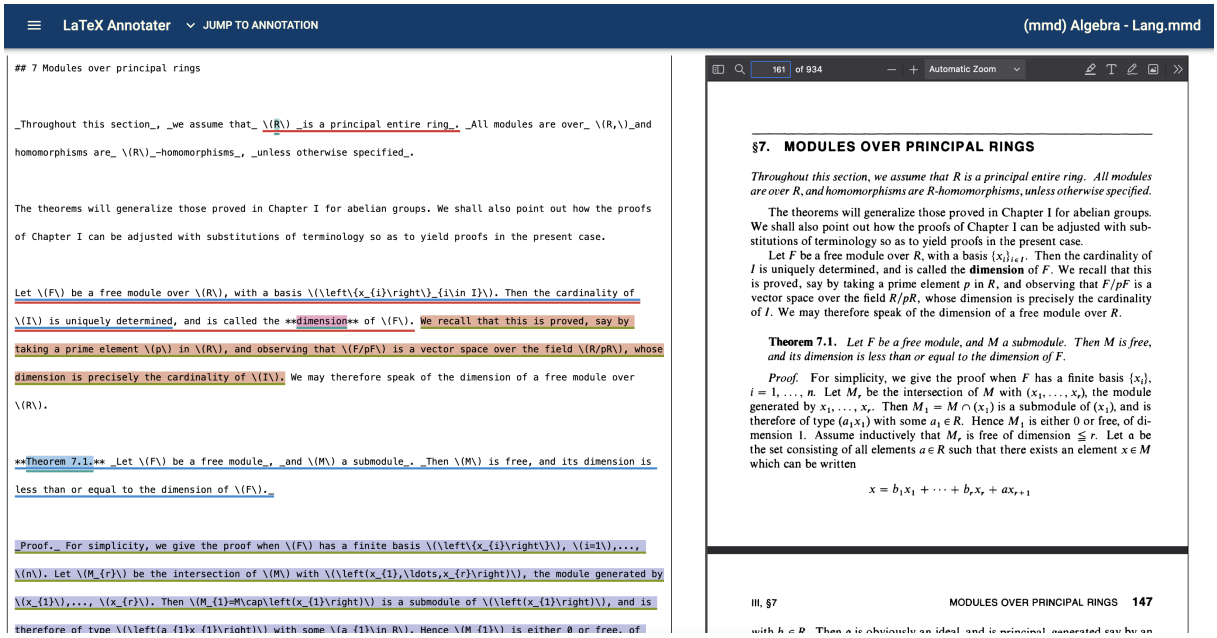

Figure 13: A screenshot taken of our custom annotation tool. On the left, we have the annotation window and the MMD being annotated, and on the right we have the corresponding PDF for ease of reading. The underline color of an annotation corresponds to its entity type, while the highlight color indicates an annotated relation to another entity (which is highlighted when hovered).