ProofLang: the Language of arXiv Proofs*

Henry Hammer[0009−0009−3942−8753], Nanako Noda[0009−0001−2664−2330], and Christopher A. Stone[0009−0006−5720−3433]

Harvey Mudd College, Claremont CA 91711 USA
{stone,hhammer,nnoda}@cs.hmc.edu

Abstract. The ProofLang Corpus includes 3.7M proofs (558 million words) mechanically extracted from papers that were posted on arXiv.org between 1992 and 2020. The focus of this corpus is proofs, rather than the explanatory text that surrounds them, and more specifically on the *language* used in such proofs. Specific mathematical content is filtered out, resulting in sentences such as Let MATH be the restriction of MATH to MATH. This dataset reflects how people prefer to write (informal) proofs, and is also amenable to statistical analyses and experiments with Natural Language Processing (NLP) techniques.

Keywords: Dataset · Informal proofs · Natural Language Processing.

1 Introduction

How do people use language in proofs written for other humans? Is "Assume..." more or less common than "Suppose..."? How closely do the English-language subsets accepted by tools like Mizar [1], Naproche-SAD [2], or Isar [3] correspond to the way proofs are written in the literature? How well do off-the-shelf NLP tools like NLTK [4] work on the stylized language of English-language proofs? Is Ganesalingam [5] correct that "Mathematical language is exceptionally well-suited" to analysis "in the way that generative syntacticians and semanticists analyze natural languages"?

The ProofLang Corpus[1] is a resource for addressing such questions. It consists of the text of English-language proofs mechanically extracted from the LaTeX sources of papers posted on arXiv.org from 1992 through April 2020. In order to focus on the language used in proofs (and make repeated word patterns more apparent), LaTeX mathematical content is replaced by the token MATH. Further cleanup steps include replacing references like \ref{...} or Theorem 2(a) by REF; replacing citations like \cite{...} or Smith [42] with CITE; and replacing proper names with NAME.

The resulting dataset (a few examples appear in Figure 1) is freely available for download or can be directly accessed from Python scripts.

* This research was supported in part by the Jean Perkins Foundation and by the NSF under Grant No. 1950885. Any opinions, findings, or conclusions are those of the authors alone, and do not necessarily reflect the views of the NSF or JPF.

[1] https://huggingface.co/datasets/proofcheck/prooflang

```
Existence is a special case of CITE. Uniqueness follows from the proof of CITE.
This is exactly REF. (Actually, that reference works with the abelian variety MATH ...
We have MATH if and only if MATH, which is equivalent to MATH.
REF stays valid when considered locally at a prime number MATH, that is, replacing ...
What we need to prove is equivalent to the statement that the subgroup MATH acts ...
This follows by taking MATH in the following lemma.
Interpret MATH as a MATH real matrix by identifying MATH with MATH. Let MATH be ...
Let us recall the situation of REF : we have a symplectic basis MATH of MATH with ...
We already know MATH by REF , which is the first statement in REF . Next, we have ...
REF is a special case of REF as follows. Pick MATH. Then MATH since multiplication by ...
Note that REF already implies that MATH acts trivially on MATH, so that it remains to ...
Assume that MATH is an extension of degree MATH, so MATH, MATH and MATH as in ...
This follows with some algebraic manipulation from REF (see also CITE). Indeed, ...
Let MATH be a quotient of theta constants as in REF . We start by proving MATH. Note ...
The NAME coefficients are finite sums of factors MATH from the definition of MATH. ...
We have already proven the result for MATH. Any element MATH can be written as MATH ...
Given MATH, let MATH be a generator of MATH as in the definition of MATH. Let MATH ...
This is an elementary calculation built on the observation that each element MATH in ...
Substituting MATH into REF gives the result.
See REF.
```

Fig. 1. The Beginnings of 20 Extracted Proofs

2 Constructing the ProofLang Corpus

We started with LaTeX sources for 1.6M papers submitted to arXiv.org between 1992 and mid-2020, retrieved using arXiv's Bulk Data Access. We identified 352K candidate papers where some source file contained the line `\begin{proof}`. After skipping papers where no proof could be extracted and running language detection on extracted proofs (to remove non-English output), we ended up with 328K papers with at least one extracted proof. These papers hail from a variety of different fields but the bulk of relevant papers are in mathematics or computer science: looking at arXiv subject tags, 281K papers have a math tag, 68K CS, 20K physics, 5.1K systems, 3.4K economics, and 2.4K biology.

The proofs were extracted using a Python script simulating parts of LaTeX (including defining/expanding macros). It does no real typesetting, throws away text not in `\begin{proof}`...`\end{proof}`, and compresses math content to `MATH`. Surprisingly, trying to implement LaTeX more faithfully gave worse results; the more the extractor knew about idiosyncrasies of TeX, the more it ran into other imperfectly simulated constructs, with unrecoverable errors in complex macros. It worked better to hard-code common cases (e.g., common environments to skip because they represent figures or diagrams; common macros to ignore because they only affect spacing or color) and to ignore unrecognized macros.

During extraction, math-mode formulas (signalled by `$`, `\[`, `\begin{equation}`, etc.) become `MATH`; `\ref{...}` and its variants (`\autoref`, `\subref`, etc.) become `\REF`; `\cite{...}` and its variants (`\Citet`, `\shortciteNP`, etc.) become `CITE`; words that appear to be proper names become `NAME`; and `\item` becomes `CASE:`.

We then run a cleanup pass on the extracted proofs that includes fixing common extraction errors (e.g., due to uninterpreted macros); replacing textual

references by REF (e.g., Theorem A.2 or Postulate (*)); replacing textual citations with CITE (e.g., Page 47 of [3]); and replacing more proof-case markers with CASE: (e.g., Case 3)).

The resulting dataset (3.7M proofs containing 558M words in 38.9M sentences) is publicly available under a Creative Commons Attribution 4.0 license.[2] Data can be downloaded as TSV files; accessing the data programmatically from Python is also possible using the Hugging Face `Datasets` library [6], e.g.,

```
from datasets import load_dataset
dataset = load_dataset('proofcheck/prooflang', 'proofs', split='train')
for d in dataset:
   print(d['paper'], d['proof'])
```

To loop over individual sentences from the proofs, replace `'proofs'` and `d['proof']` by `'sentences'` and `d['sentence']`.

Also available is a mapping from paper IDs to a comma-separated list of arXiv subject tags (which allows filtering proofs by subject area), and a "raw" version of the proofs that shows the extracted proofs before the cleanup pass. Further metadata for proofs can be found on arXiv, since each sentence or proof tagged with paper `<id>` is from the paper at `https://arxiv.org/abs/<id>` .

3 Experimenting with the Corpus

So far, we have focused on collecting the data, and have only run relatively simple experiments. For example, it's easy to count occurrences of words. In this corpus, the word assume appears more often than suppose (1119K vs. 784K case-insensitive occurrences), but at the start of a sentence Suppose... occurs much more often than Assume... (436K vs. 263K occurrences).

3.1 Identifying collocations

We can also count occurrences of repeated sentences. Of course short ones like Let MATH. or Then MATH. are most likely to repeat exactly (315K and 185K times), but we can also compare counts for variants of longer sentences, e.g.,

Without loss of generality we may assume that MATH.	4.0K
Without loss of generality assume that MATH.	2.3K
Without loss of generality we assume that MATH.	2.2K
Without loss of generality we can assume that MATH.	2.2K
We may assume without loss of generality that MATH.	0.8K

Longer sentences rarely have a large number of exact repeats, so one can also look for collocations (idiomatic phrases) occurring inside sentences.

[2] Papers on arXiv.org are not in the public domain, but all relevant legal factors (the purpose and character of the use, the nature of the original works, the amount and substantiality of the copied text, and the effect upon the potential market for the original) favor a conclusion of Fair Use under US copyright law.

ad absurdum	equally likely	regularly varying
almost surely	finitely many	roughly speaking
arithmetic progressions	greater than	slowly varying
barycentric subdivision	MATH-differentially private	social welfare
blow ups	maximally entangled	spherical harmonics
brownian motion	mutatis mutandis	sufficently large
brute force	one-point compactification	supplementary material
compactly supported	pigeon hole	tamely ramified
complementary slackness	principally polarized	train track
dominated convergence	purely inseparable	vice versa
dynamic programming	references therein	without loss

Fig. 2. Selected 2-Word Collocations from the Corpus

We started from bigrams (2-word sequences) in our corpus. For example, the sentence Let MATH be even contains the three bigrams (let, MATH), (MATH, be), and (be, even). For bigrams occurring more than 500 times we calculated statistics such as Pointwise Mutual Information and χ^2 to identify pairs of words that appeared together more often than could be explained by chance. This automatically finds a large number of conventional phrases; a few are shown in Figure 2. By treating the discovered collocations as single-word units and iterating the bigram process, we can find longer collocations like induces an isomorphism between and it suffices to check that.

3.2 Testing the NLTK Part-of-Speech Tagger

NLTK, the Natural Language Toolkit [4], is a Python package for traditional NLP statistical algorithms. We were curious how well it would do at POS (Part-of-Speech) tagging on our corpus, i.e., identifying a part of speech for each word in a sentence. As a simple example, the three words in We like proofs should be tagged as personal-pronoun, third-person-present-verb, and plural-noun.

NLTK's default POS tagger (a perceptron trained on text from the Wall Street Journal) is reasonably accurate for many purposes, but experiments with the corpus showed it performs more poorly on mathematical language.

The first aspect is the vocabulary, not only because mathematical jargon is rare in the training corpus, but also because the usage of more common words differs. For example, integral is generally used as an adjective in newspapers, so the default tagger always tags integral as an adjective even though within proofs integral is often a noun. Worse, integral is in the top 300 most-frequent words (occurring 187K times), and whenever the tagger mis-tags integral, it is likely to mis-tag neighboring words as well. Other problems include literal, metric, and variable (where the default tagger confuses noun vs. adjective) and partition, factor, and embedding (where the default tagger confuses verb vs. noun).

The most consistent problem is the mis-tagging of verbs in imperative sentences. Trained on news text, the default tagger seems unaware that sentences can start with verbs and tags these unexpected verbs as proper nouns, apparently because the start of a sentence is always capitalized. For example, given the sentence Suppose MATH is even , NLTK mis-tags Suppose MATH as proper

nouns as if this were a declarative sentence like "John Smith is tall" (except that it further mis-tags even as an adverb instead of an adjective).

Similarly, Consider for all MATH the subtree MATH of MATH spanned by the root and the first MATH leaves MATH with MATH is mis-tagged with Consider as a proper noun, subtree as an adjective, and leaves as a verb.

Preliminary experiments with re-training the NLTK POS tagger showed that adding manually tagged sentences from our corpus to the training set—5 imperative sentences for each of 50 common verbs—significantly improved its performance on imperative sentences with these and other verbs. Interestingly, adding more sentences for each verb made performance worse overall; we hypothesize it was over-fitting and memorizing just the specific imperative verbs we provided.

4 Conclusion and Future Work

The ProofLang Corpus is a new dataset showing how people naturally phrase their claims, structure mathematical arguments, and use language in proofs written for humans. We hope it can aid the development of language-based proof assistants and proof checkers for professional and educational purposes.

As the corpus is heuristically extracted from LaTeX files, it does contain errors; one can probably spend arbitrary amounts of time polishing the corpus by handling more corner cases. More general cleanup strategies might be helpful, e.g., training *Named Entity Recognition* to recognize references to mathematical facts (e.g., Theorem 2a.) replacing our ad hoc regular expressions.

It would also be interesting to distinguish MATH-as-noun (e.g., Then $n > 1$ is prime) from MATH-as-clause (e.g., If $n > 1$ then the theorem holds). We conjecture that generalized POS tagging might be helpful here.

While we plan further analyses ourselves (e.g., identifying more general collocations and applying language-model techniques), we also hope that others can find interesting applications for this dataset.

References

1. Grabowski, A., Kornilowicz, A., Naumowicz, A.: *Mizar in a Nutshell*. Journal of Formalized Reasoning **3**(2), 153–245 (2010). https://doi.org/10.6092/issn.1972-5787/1980
2. De Lon, A., Koepke, P., Lorenzen, A.: *Interpreting Mathematical Texts in Naproche-SAD*. In: Intelligent Computer Mathematics (CICM 2020), LNCS, vol. 12236, pp. 284–289. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-53518-6_19
3. Wenzel, M.: *Isar — A Generic Interpretative Approach to Readable Formal Proof Documents*. In: Bertot, Y., Dowek, G., Théry, L., Hirschowitz, A., Paulin, C. (eds) Theorem Proving in Higher Order Logics (TPHOLs 1999), LNCS, vol. 1690, pp. 167–183 (1999). https://doi.org/10.1007/3-540-48256-3_12
4. Bird, S., Loper, E., Klein, E.: *Natural Language Processing with Python*. O'Reilly Media Inc. (2009)
5. Ganesalingam, M.: *The Language of Mathematics*. Springer-Verlag (2013). https://doi.org/10.1007/978-3-642-37012-0
6. Hugging Face: *Datasets*, `https://huggingface.co/docs/datasets`