BERT is not The Count: Learning to Match Mathematical Statements with Proofs

Weixian Waylon Li[1] **Yftah Ziser**[1] **Maximin Coavoux**[2*] **Shay B. Cohen**[1]
[1] University of Edinburgh, School of Informatics, Edinburgh
[2] Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG,
W.Li-67@sms.ed.ac.uk yftah.ziser@ed.ac.uk

Abstract

We introduce a task consisting in matching a proof to a given mathematical statement. The task fits well within current research on Mathematical Information Retrieval and, more generally, mathematical article analysis (Mathematical Sciences, 2014). We present a dataset for the task (the MATCH dataset) consisting of over 180k statement-proof pairs extracted from modern mathematical research articles.[1] We find this dataset highly representative of our task, as it consists of relatively new findings useful to mathematicians. We propose a bilinear similarity model and two decoding methods to match statements to proofs effectively. While the first decoding method matches a proof to a statement without being aware of other statements or proofs, the second method treats the task as a global matching problem. Through a symbol replacement procedure, we analyze the "insights" that pre-trained language models have in such mathematical article analysis and show that while these models perform well on this task with the best performing mean reciprocal rank of 73.7, they follow a relatively shallow symbolic analysis and matching to achieve that performance.[2]

1 Introduction

Research-level mathematical discourse is a challenging domain for Natural Language Processing (NLP). Mathematical articles frequently switch between natural language and mathematical formulae, and a semantic analysis of mathematical text needs to solve relationships (e.g. coreference) between mathematical symbols and concepts. Moreover, mathematical writing follows many conventions, such as variable naming or typography that are implicit, and may differ between subfields.

Statement. When $m = 0$ we have $E^0_{rg} = \emptyset$, and when $m \neq 0$ we have $E^0_{rg} = E^0$.
Proof. When $m = 0$, the image of r is $\{1\}$. Hence $E^0_{rg} = \emptyset$. When $m \neq 0$, the map r is a surjective proper map. Hence $E^0_{rg} = E^0$.

Figure 1: Example of a statement-proof pair.

However, mathematical research can benefit from NLP (Mathematical Sciences, 2014), in particular as concerns bibliographical research: researchers need tools to find work relevant to their research. Indeed, prior NLP work on mathematical research articles focused on Mathematical Information Retrieval (MIR) and related tools or data (Zanibbi et al., 2016; Stathopoulos and Teufel, 2016, 2015).

We introduce a task aimed at improving the processing of research-level mathematical articles and make a step towards the modeling of mathematical reasoning. Given a collection of mathematical statements and a collection of mathematical proofs of the same size, the task consists in finding and assigning a proof to each mathematical statement. We construct and release a dataset for the task (MATCH), by collecting over 180k statement-proof pairs from mathematical research articles (an example is given in Figure 1).

Related datasets, such as LEANSTEP (Han et al., 2021) and the synthetic dataset of Polu and Sutskever (2020) do not include natural language. NaturalProofs (Welleck et al., 2021), another related dataset, only consists of 32k theorem-proof pairs from ProofWiki,[3] some sub-topics in algebraic geometry and two textbooks. Our dataset is over five times larger and contains pairs extracted

*Work mostly done at the University of Edinburgh.

[1]Our dataset and code are available at https://github.com/waylonli/MATcH.

[2]Like Bert, The Count (or Count von Count; ) is a character from the television show Sesame Street. The Count likes counting, and his main role in the show is to teach this skill to children.

[3]https://proofwiki.org/xmldump/latest.xml

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 3581–3593

from advanced academic mathematical papers.

There are multiple motivations for the design of the task and our dataset. We believe it may help MIR by serving as a proxy for the search for the existence of a mathematical result, or for theorems and proofs related to one another (e.g. using the same proof technique), an important search tool for any digital mathematical library (Mathematical Sciences, 2014). Learning to match statements and proofs would also benefit computer-assisted theorem proving, as it is akin to tasks such as premise selection, also recently addressed with NLP methods (Piotrowski and Urban, 2019).

We provide first results on our proposed task with an array of neural models, aimed at scoring the likelihood of relationship between a statement and its proof. An analysis through a **symbol replacement procedure** provides insight on what such neural models are capable of learning about mathematical equations and text.

We provide two methods for decoding, one is local decoding, matching a proof to a theorem in a greedy way, and one that provides a global bipartite matching based on a structured max-margin objective. Such an architecture may have applications to other NLP problems that can be cast as maximum bipartite matching problems (for a recent similar use in a different context, see Shao et al. 2023).

Our analysis shows that pre-trained language models do not obtain significant "mathematical insight" for performing this matching, but rather rely on shallow matching. However, this does not prevent them from performing the matching relatively well in several carefully crafted scenarios, reaching an MRR of 73.7.

2 Related Work

Most NLP work on mathematical discourse focuses on improving Mathematical Information Retrieval (Zanibbi et al., 2016, MIR) by establishing connections between mathematical formulae and natural language text in order to improve the representation of formulae.

The interpretation of variables is highly dependent on the context. For example, the symbol E could denote an expectation in a statistics article, or the energy in a physics article. Some studies use the surrounding context of a formula to assign a definition or a type to the whole formula, or to specific variables. Nghiem Quoc et al. (2010) focus on identifying coreferences between mathematical formulae and mathematical concepts in Wikipedia articles. Kristianto et al. (2012) extract definitions of mathematical expressions. Grigore et al. (2009), Wolska et al. (2011) and Schubotz et al. (2016) disambiguate mathematical identifiers, such as variables, using the surrounding textual context. Stathopoulos et al. (2018) infer the type of a variable in a formula from the textual context of the formula. Another line of work focused on identifying specialized terms or concepts to improve MIR (Stathopoulos and Teufel, 2015, 2016).

Some work adapts standard NLP tools to the specificity of mathematical discourse, e.g. POS taggers (Schöneberg and Sperber, 2014), with the objective of using linguistic features to improve the search for definitions of mathematical expressions (Pagel and Schubotz, 2014). More recent work focuses on typing variables in mathematical articles (Ferreira et al., 2022), modeling formulae (Mansouri et al., 2019; Dadure et al., 2021), and selecting premises (Ferreira and Freitas, 2020, 2021).

An earlier version of our work covers some of the material in this paper (Coavoux and Cohen, 2021). The main differences between that version and the current version are the introduction of the symbol replacement evaluation (§5) and the use of pre-trained language models rather than recurrent neural networks.

3 Task Description

Given a collection of mathematical statements $\{s^{(i)}\}_{i \leq N}$, and a separate equal-size collection of mathematical proofs $\{p^{(i)}\}_{i \leq N}$, we are interested in the problem of assigning a proof to each statement.

Evaluation We use two evaluation metrics. Assuming that a system predicts a ranking of proofs, instead of providing only a single proof, we evaluate its output with the Mean Reciprocal Rank (MRR) measure: $\text{MRR}(\{\hat{r}_i\}_{i \in \{1,\dots N\}}) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\hat{r}_i}$, where N is the number of examples and $\hat{r}_i$ is the rank of the gold proof for statement number i, as predicted by the system.

As a second evaluation metric, we use a simple accuracy, i.e. the proportion of statements whose first-ranked proof is correct.

By construction (see §4), it is possible though unlikely that the same mathematical statement occurs several times in the dataset. It is more unlikely that

Number of articles in the MREC corpus	439,423
Extracted articles with statement-proof pairs	27,841
Total number of statement-proof pairs	184,094
Number of (primary) categories	(120) 135
Average number of categories per article	1.7

Most represented primary categories	# articles	# pairs
math.AG Algebraic Geometry	2848	22029
math.DG Differential Geometry	2030	12440
math.CO Combinatorics	1705	10548
math.GT Geometric Topology	1539	9234
math.NT Number theory	1454	9521
math.PR Probability	1422	7660
math.AP Analysis of PDEs	1386	6981
math-ph Mathematical Physics	1249	6491
math.FA Functional Analysis	1143	8011
math.GR Group Theory	970	7806
math.DS Dynamical System	961	6424
math.QA Quantum Algebra	944	8074
math.OA Operator Algebras	923	8050

Table 1: Statistics about the dataset and categories of mathematical articles.

several occurrences have exactly the same formulation and use the same variable names. Therefore, we consider a match to be correct if and only if it is associated with its original proof.

4 Dataset Construction

This section describes the construction of the MATCH dataset of statement-proof pairs (see Figure 1 for an example).

Source Corpus We use the MREC corpus[4] (Líška et al., 2011) as a source. The MREC corpus contains around 450k articles from ArxMLiV (Stamerjohanns et al., 2010), an on-going project aiming at converting the arXiv[5] repository from LaTeX to XML, a format more suited to machine processing. In this collection, mathematical formulae are represented in the MathML[6] format, a markup language.

Statistics We extract statement-proof pairs as described in Appendix A. Our processing of MREC includes the identification of statement-proof pairs through meta tags and the linearization of the representation of mathematical equations.

We report in Table 1 some statistics about the dataset we collected. The extracted articles were from a diverse set of mathematical subdomains, and connected domains, such as computer science (746 articles from 30 subcategories) and mathematical physics (2562 from 31 subcategories). There are in average 6.6 statement-proof pairs per article. We report statistics about the size of statements and proofs in the number of tokens in Table 2. We report the number of tokens in formulae (math), in the text itself (text) and in both (text+math). On average, proofs are much longer than statements. Statements and proofs have approximately the same proportion of text and math. Overall, the variation in the number of tokens across statements and proofs is extremely high, as illustrated by the standard deviation (SD) of all presented metrics.

Statements	Min	Max	Mean±SD
Text+math	20	500	80±57
Text only	1	398	30±20
Math only	0	470	58±20
Math proportion	0%	99.5%	58%±20
Proofs			
Text+math	20	500	210± 127
Text only	1	467	81 ± 56
Math only	0	495	129 ± 96
Math proportion	0%	99.6%	56%± 21

Table 2: Number of tokens in the dataset. We report for statements and proofs the minimum, maximum and average number of tokens broken down by type ('math' for tokens extracted from formulae and 'text' for the others). A value of 0 for, e.g. the 'math only' row, means that the statement or proof does not contain mathematical symbols or formulae.

5 Symbol Replacements

With our current dataset setup, we implicitly make the assumption that both the theorem and the proof are authored by the same authors. This assumption is incongruent with the MIR-flavor of our task. First, it is not useful for researchers to match proofs they authored. Second, each person has a unique writing style expressed by unique mathematical jargon and notations. To relieve of this assumption, we introduce several symbol replacement levels for changing the names of the proof variables. Then, we train and test our models using these altered datasets. These replacement levels also provide insight on the ability of our models to semantically analyze the input statement-proof pairs.

Symbol Replacement Levels We propose different levels of symbol replacement, focusing on mathematical notation. More precisely, we aim to replace the proof variable names if they appear in

[4] https://mir.fi.muni.cz/MREC/, version 2011.4.439.

[5] https://arxiv.org/

[6] https://www.w3.org/Math/

the statement without damaging the proof semantics. To do that, we change symbols that appear both in the proof and the statement. We do not change constant symbols such as π, as they often carry semantic meaning outside of the proof scope.

We experiment with four levels of symbol replacement (examples in parenthesis):

- **Symbol conservation** - all symbols remain intact, so the theorem and the proof overlap. All previous work uses that. $(a_n = a_{n-1} + a_{n-2})$
- **Partial symbol replacement** - A fraction of α of all the symbols in the proof remain the same, and the rest are changed. In our experiments, we use $\alpha = 0.5$. $(x_n = x_{n-1} + x_{n-2})$
- **Full symbol replacement** - all symbol names are changed ($\alpha = 1.0$ as above). $(x_i = x_{i-1} + x_{i-2})$
- **Symbol transposition** - We permute the variables' names such that no symbol remains the same, thus changing their original functionality. $(n_a = n_{a-1} + n_{a-2})$

More details about this appear in Appendix C.

6 Bilinear Similarity Model

We propose a model based on an encoder (§8) that constructs fixed-size vector representations for statements and proofs, and a similarity function that scores the relatedness of a statement-proof pair.

Trainable Bilinear Similarity Function Given the encoded representations of a statement $\mathbf{s} = \text{enc}(s)$ and a proof $\mathbf{p} = \text{enc}(p)$, we compute an association score with the following bilinear form:

$$\text{score}(\mathbf{s}, \mathbf{p}) = \mathbf{s}^\top \cdot \mathbf{W} \cdot \mathbf{p} + b,$$

where $\mathbf{W}$ and b are parameters that are learned together with a self-attentive encoder parameters (§8).

Local Decoding For a collection of n statements and proofs, we first score all possible pairs (s, p), and construct a matrix $M = (m_{ij}) \in \mathbb{R}^{n \times n}$, with

$$m_{ij} = \text{score}(\mathbf{s}^{(i)}, \mathbf{p}^{(j)}),$$

where $\mathbf{s}^{(i)}$ and $\mathbf{p}^{(j)}$ are the encoded representations of, respectively, the i^{th} statement and the j^{th} proof. Then we can straightforwardly sort each row by decreasing order and assign the proof ranking to the corresponding statement. The best ranking proof $\hat{p}$ for statement i satisfies $\hat{p}^{(i)} = \arg\max_j \ m_{ij}$. We call this decoding method 'local', since it does not take into account dependencies between assignments. In particular, several statements may have the same highest-ranking proof.

Global Decoding The local decoding method overlooks a crucial piece of information: a proof should correspond to a single statement. In a worst-case situation, a small number of proofs may score high with most statements and be systematically assigned as highest-ranking proof by the local decoding method.

In preliminary experiments, we analyzed the output of our system with local decoding on the development set, focusing on the distribution of the single highest-ranking proof for each statement. We found that 23% of the proofs were assigned to at least two different statements, whereas more than 40% of proofs were assigned to no statement. See also Appendix B.[7]

We propose a second decoding method based on a global constraint on the output: a proof can be assigned only to a single statement. Intuitively, the constraint models the fact that if a proof is assigned by the system to a certain statement with high confidence, we can rule it out as a candidate for other statements. Under this constraint, the decoding problem reduces to a classical maximum weighted bipartite matching problem, or equivalently, a Linear Assignment Problem (LAP). In more realistic scenarios (e.g. if the input sets of statements and proofs do not have the same size), the method would require some adaptation.

Formally, we define an assignment A as a Boolean matrix $A = (a_{ij}) \in \{0, 1\}^{n \times n}$ with the following constraints:

$$\forall i \forall j, \sum_j a_{ij} = \sum_i a_{ij} = 1,$$

i.e. each row and each column of A contains a single non-zero coefficient. The score of an assignment A is the sum of scores of the chosen edges:

$$\text{score}(A, M) = \sum_i \sum_j a_{ij} m_{ij}.$$

Finally, global decoding consists in solving the

[7] We used a simple encoder for these experiments, which we describe in §8 (NPT).

following LAP:

$$\hat{A}(M) = \underset{\substack{A\in\{0,1\}^{n\times n} \\ \text{s.t. } \forall i \forall j, \sum_j a_{ij} = \sum_i a_{ij} = 1}}{\arg\max} \text{score}(A, M).$$

The LAP is solved in polynomial time by the Hungarian algorithm (Kuhn, 1955), the LAP-Jonker-Volgenant algorithm (LAP-JV; Jonker and Volgenant, 1987), or the push-relabel algorithm (Goldberg and Kennedy, 1995). These methods have a $\mathcal{O}(n^3)$ time complexity where n is the number of pairs, and $\mathcal{O}(n^2)$ memory complexity. This is too expensive in our case, due to our dataset size.

To remedy this limitation, when we perform decoding on a large set, we only consider the k best-scoring proofs (i.e. outgoing edges in the bipartite graph) for each statement, which makes the number of edges linear in the number of pairs n (considering k fixed). Moreover, we use a modification of the LAP-JV algorithm specifically designed for sparse matrices (LAP-MOD; Volgenant, 1996).

7 Local and Global Training

We propose two training methods for the similarity model above: a local training method that only considers statements in isolation (§7.1) and a global model trained to predict a bipartite matching (§7.2), with a hybrid global-local objective.

7.1 Local Training

We would like to train our model to assign a high similarity to the gold statement-proof pair, and a low similarity to all other statement-proof pairs. This corresponds to the following objective, for a single statement s and its gold proof p:

$$\begin{aligned}\mathcal{L}_{\text{LOC}}(s, p, P; \boldsymbol{\theta}) &= -\log \mathbb{P}(p|s; \boldsymbol{\theta}) \\ &= -\log\left(\frac{\exp(\text{score}(\mathbf{s}, \mathbf{p}))}{\sum_{p'\in P}\exp(\text{score}(\mathbf{s}, \mathbf{p}'))}\right),\end{aligned}$$

where P is the set of proofs, and $\boldsymbol{\theta}$ are the parameters of the model. Directly optimizing this loss function requires the computation of $\mathbf{p} = \text{enc}(p)$ for every proof in the dataset, for a single optimization step. This is not realistic considering memory limitations, the size of the train set, and the fact that our self-attentive encoder is the most computationally expensive part of the network.

Instead, we sample minibatches of b pairs and optimize the following proxy loss for the sequence $S' = (s_1, \ldots, s_b)$ of statements and the sequence $P' = (p_1, \ldots, p_b)$ of corresponding proofs:[8]

$$\mathcal{L}'_{\text{LOC}}(S', P'; \boldsymbol{\theta}) = \sum_{i=1}^{b} \mathcal{L}_{\text{LOC}}(s^{(i)}, p^{(i)}, P'; \boldsymbol{\theta}).$$

In practice, we sample uniformly and without replacement b pairs from the training set at each stochastic step.

7.2 Hybrid Local and Global Training

The local training method only considers statements in isolation. Even though we expect a locally trained model to perform better with global decoding, we hypothesize that a model that is trained to predict the full structure (a bipartite matching) will be even better.

For a collection of n proofs and n statements, the size of the search space (i.e. the number of bipartite matchings) is $n!$, since each matching corresponds to a permutation of proofs. As a result, the use of a globally normalized model is impractical. We turn to a max-margin model that does not require normalization over the full search space.

We use the following max-margin objective, for a set B of n pairs corresponding to matrix M:

$$\begin{aligned}\mathcal{L}_{\text{GLOB}}(B; \boldsymbol{\theta}) = \max(0, \Delta(\hat{A}, I) \\ + \text{score}(\hat{A}, M) - \text{score}(I, M)),\end{aligned}$$

where $\boldsymbol{\theta}$ is the set of all parameters $\hat{A}$ is the predicted assignment and I is the gold assignment, i.e. the identity matrix. The structured cost

$$\Delta(\hat{A}, I) = \sum_{ij} \max(0, (\hat{A} - I)_{ij})$$

aims at enforcing a margin for each individual assignment.

The computation of this loss requires exact decoding for each optimization step. Since exact decoding is only feasible for a small n, and since we need to keep track of all intermediary vectors to compute the backpropagation step,[9] we perform each stochastic optimization step on a minibatch of pairs of size b. Since this global objective had a slow convergence rate (§8), in practice, we use a hybrid local-global objective: $\mathcal{L}'_{\text{LOC}} + \mathcal{L}_{\text{GLOB}}$.

	Symbol Replacement Level							
	Conservation		Partial		Full		Transposition	
Encoder-Decoder	MRR	Acc	MRR	Acc	MRR	Acc	MRR	Acc
NPT-Local-Local	63.22	56.08	47.19	39.24	40.36	32.52	56.17	48.30
NPT-Local-Global	-	61.89	-	42.55	-	35.43	-	53.49
NPT-Global-Global	-	62.14	-	43.68	-	35.85	-	55.28
SCRATCHBERT-Local-Local	**73.73**	67.12	**64.79**	57.20	**60.67**	52.54	**73.17**	66.51
SCRATCHBERT-Local-Global	-	**74.68**	-	**62.80**	-	**57.69**	-	**74.03**
SCRATCHBERT-Global-Global	-	71.38	-	58.06	-	52.31	-	70.32
MATHBERT-Local-Local	54.51	46.45	44.31	36.10	38.91	30.62	52.57	44.52
MATHBERT-Local-Global	-	49.77	-	37.92	-	32.03	-	47.43
MATHBERT-Global-Global	-	45.38	-	33.64	-	28.47	-	43.41

Table 3: The MRR and accuracy scores for different combinations of encoders, decoders, and symbol replacement levels. All the models are trained and tested on the same replacement level. Best result in each column is in bold. Following the model name, we include its encoder and decoder type (both being either Local or Global). We do not include MRR scores for global inference, as there matching is done for all theorems together without ranking.

8 Experimental setup

Dataset We use the dataset whose construction is described in §4. We shuffle the collection of statement-proof pairs before performing a $80\%/10\%/10\%$ train-development-test split, corresponding to 147278 pairs for the training sets and 18408 pairs for the development and tests. We experiment with a default, Mixed, split and a harder Unmixed split (see §9.3).

Encoders We experiment with several encoders to obtain neural representations of the theorem and proof pairs. Our first encoder is a simple self-attentive encoder. We use $\ell = 2$ self-attentive layers with 4 heads to obtain contextualized embeddings of dimension $d = 300$. The query and key vectors have size $d_k = 128$. We construct a vector representation for the text with a max-pooling layer over the contextualized embeddings of the last self-attention layer. We do not use any form of pre-training for this encoder and hence name it "no pre-training encoder" (NPT). In addition, we experiment with a BERT model (Devlin et al., 2019) as an encoder. We do not use the pre-trained version provided by Devlin et al., but rather pre-train the base version from scratch (SCRATCHBERT), but we do compare our results against a math-tailored pre-trained version of BERT (Peng et al. 2021; see below). Both the NPT and SCRATCHBERT vocabularies are customized for our dataset, as preliminary experiments revealed the importance of the model-task vocabulary match.[10]

To further demonstrate how crucial this vocabulary match is, we experiment with math-BERT (MATHBERT; Shen et al. 2021), a state-of-the-art pre-trained model for mathematical formula understanding. This model is pre-trained on a large mathematical corpus ranging from pre-kindergarten, to high-school, to college graduate level mathematical content, including professional mathematical papers, using the BERT masked language modeling (MLM) task. We use the pre-trained version provided by the authors, *without vocabulary customization*. All of our encoders are fine-tuned on the matching task. In addition, we experiment with a naive token-matching system that computes cosine similarities between TF-IDF representations of statements and proofs. We discovered that their performance was very low, ranging from 11.4 to 29.8 (MRR), so we did not experiment with them further.

Hyperparameters For pretraining SCRATCHBERT, we first train a new word piece tokenizer[11]. Next, we train the SCRATCHBERT model on the MLM task for 60 epochs (around 3 days) using four NVIDIA V100 GPUs. We evaluate the language model every 500 steps, where one step stands for training on one example, and choose the one with

[8] We also experimented with a Noise-Contrastive Estimation approach (Gutmann and Hyvärinen, 2012). However, it exhibited a much slower convergence rate.

[9] In particular, the computation graph needs to conserve all encoding layers for the $2n$ texts involved.

[10] This supports the findings of Chalkidis et al. (2020), for example, in a different domain.

[11] https://huggingface.co/docs/transformers/tokenizer_summary#wordpiece

the best performance on the validation set.

We perform local and global training / finetuning respectively for the NPT model, MATHBERT, and SCRATCHBERT. NPT has 15M parameters while MATHBERT and SCRATCHBERT have 110M parameters. We observed in initial experiments that training only with the global objective required a long time to converge. Therefore, we used the following global-local objective: $\mathcal{L}'_{\text{LOC}} + \mathcal{L}_{\text{GLOB}}$, that we optimized by alternating one stochastic step for each loss.

We train the NPT model for 400 epochs (around 1 day with two GPUs) over the whole training set for local and global training. We use batches of size $b = 60$ and set learning rate $l = 5 \times 10^{-3}$ with the Averaged Stochastic Gradient Descent (ASGD; Polyak and Juditsky 1992) optimizer. We use an exponential learning rate scheduler (the learning rate multiplied by 0.996 after each epoch) to stabilize the optimizer in the latter training procedure (after 300 epochs). We evaluate the performance of the model on the validation set every 20 epochs during training and select the best one among these intermediate models.

We use four NVIDIA V100 GPUs to fine-tune MATHBERT and SCRATCHBERT on the training set for 60 epochs (around 2 days) with a learning rate of $l = 2 \times 10^{-3}$, an ASGD optimizer, batches of size $b = 16$, and a scheduler that multiplies the learning rate by 0.99 after each epoch. We choose the best model on the validation set, evaluating the models every five epochs.

Global Decoding Recall that exact global decoding is only feasible for a small subset of pairs. During global training, we chose a batch size small enough to perform exact decoding. However, it is not feasible to perform exact decoding on the whole development and test corpora. Therefore, we prune the search space by keeping only the 500-best candidate proofs for each statement, and use the LAP-MOD algorithm designed for sparse matrices. In practice, we used the implementations of the LAP-JV and LAP-MOD algorithms from the `lap` Python package,[12] for respectively exact decoding on mini-batches during global training and decoding on whole datasets during evaluation.

[12] https://github.com/gatagat/lap

9 Results

First, we assess the task difficulty under different replacement levels using different encoders and schemes (global or local training, global or local decoding). In particular, we are interested in assessing whether global decoding improves accuracy when training is only local, and how the more complex global training method fares with respect to local training. We then measure the informativeness of different types of input: text, mathematical formulae, or both. The comparison of these settings is meant to provide insight into which type of information is crucial to the task. Finally, we experiment with a cross replacement levels setup, i.e., when a model is tested on a different symbol replacement level from the one that was used during training. We hope this experiment will shed some light on the importance of training models on real-world datasets.

9.1 Main Results

Table 3 presents our results. We report MRR (if relevant) and accuracy scores across different levels of symbol replacement.

Encoders While MATHBERT is pre-trained on millions of examples curated from mathematical contents, it performs worse than the less complex NPT encoder, which is trained solely on the downstream task across all symbol replacement levels and decoders.[13] SCRATCHBERT, which shares MATHBERT architecture and NPT customized vocabulary, is outperforming both consistently. These results demonstrate the vocabulary importance for learning from mathematical texts.

Symbol Replacement Levels Difficulty Best performance is achieved when no symbol is replaced (Conservation), as the models can match identical symbols across theorem-proof pairs. The models achieve similar performance with Transposition replacement. These results suggest that the symbols' order, context, and function within the mathematical text do not play a significant role when the theorem and proof share the same symbols. In contrast, when the symbol names are changed (Partial and Full replacements), we observe a sharp decline in results.

[13] We observe similar trends when fine-tuning the out-of-the-box BERT model on the matching task.

	Source \ Target	Symbol Replacement							
		Conservation		Partial		Full		Transposition	
		MRR	Acc	MRR	Acc	MRR	Acc	MRR	Acc
Mixed	Conservation	73.73	67.12	43.87	36.36	29.74	25.36	69.56	62.23
	Partial	**74.21**	**67.96**	**64.79**	**57.20**	53.77	45.40	72.13	65.42
	Full	65.26	57.63	63.01	55.13	**60.67**	**52.54**	64.59	56.92
	Transposition	73.78	67.40	43.67	36.02	29.76	25.47	**73.17**	**66.51**
Unmixed	Conservation	**67.62**	57.54	21.26	13.83	7.09	3.68	59.54	48.61
	Partial	61.19	50.73	**55.26**	**44.45**	50.68	39.94	59.63	49.01
	Full	55.68	45.18	54.92	44.34	**54.62**	**44.22**	55.38	44.91
	Transposition	67.5	**57.76**	23.31	15.26	8.98	4.97	**66.25**	**59.29**

Table 4: Cross-replacement levels performance for the SCRATCHBERT-Local-Local model for both splits: Mixed and Unmixed.

Input	Symbol Replacement			
	Conservation		Full	
	MRR	Acc	MRR	Acc
	NPT			
Text	22.51	16.68	22.51	16.68
Math	65.08	58.47	34.55	27.30
Both	63.22	56.08	40.36	32.52
	SCRATCHBERT			
Text	36.85	29.18	36.85	29.18
Math	63.10	55.92	41.64	34.01
Both	**73.73**	**67.12**	**60.67**	**52.54**

Table 5: SCRATCHBERT-Local-Local and NPT-Local-Local performance for different input types. Both stand for the original and complete input.

Training and Decoding Effects In all settings, global *decoding* substantially improves accuracy. These improvements are more noticeable for the NPT and SCRATCHBERT encoders. For NPT, we observe better performance when using global *training*, but not for SCRATCHBERT and MATHBERT. Due to the lack of computational resources, we can not reach the global training full potential when using highly expressive encoders such as SCRATCHBERT and MATHBERT, which share BERT-base architecture.

9.2 Effect of Input Type Analysis

To better understand the importance of each input type, we examine SCRATCHBERT-Local-Local and NPT-Local-Local performance when fed with text, mathematical formulae, or both (Table 5). We test them on the Conservation and Full symbol replacement levels. The mathematical formulae input plays a more significant role for both models than the textual input. When trained and tested on the Conservation replacement level, NPT-Local-Local makes better use of the mathematical formulae input than the more expressive, pre-trained SCRATCHBERT-Local-Local. When trained and tested with Full replacement, where the models cannot rely on simple token-matching, NPT-Local-Local suffers from a sharper performance decline than SCRATCHBERT-Local-Local when fed with mathematical formulae input. These results suggest that when applied to the Conservation data, a less expressive model can get high results by harnessing simple token matching. SCRATCHBERT-Local-Local performs better for both replacement levels when fed with text and complete input.

9.3 Cross Replacement Setup

Table 4 shows the effect of testing a model on different symbol replacements than the one the model was trained on. We use the SCRATCHBERT-Local-Local model for all of our experiments. We observe a sharp decline in results when SCRATCHBERT-Local-Local is trained with Conservation and tested on Partial or Full. These drops in performance suggest the model developed a strong dependency on exact symbol name matching. In addition, the replacement shift from Conservation to Transposition and vice versa resulted in a minor performance drop. These results provide additional evidence for the lack of importance of mathematical functionality, order, and context of symbols' names shared across theorem and proof pairs. The model trained on the Partial symbol replacement level demonstrated significant resilience when tested with other symbol replacement levels. It outperforms the rest of the models when applied to out-of-domain re-

placement levels and the Conservation replacement level in-domain model.

In addition, we experimented with theorem-proof pairs split where pairs from the same paper could not appear in the same set: train, validation or test (Unmixed). All models exhibited a reduction in performance when trained and tested under these conditions. Particularly noteworthy was the decrease in performance observed in models that were trained using the Conservation and Transposition symbol replacement methods and evaluated on data using the Partial or Full replacement methods. These sharp declines highlight the dependence of the model on simple symbol matching rather than deeper inferential analysis.

9.4 Protected Symbols

Insofar, we overlooked that some symbols carry a default meaning over a whole mathematical domain (**protected** symbols, e.g. $P(x)$ for probability). Replacing them locally may result in a detrimental impact on semantic mathematical content. We test the impact of substituting protected symbols in a controlled setting by comparing models trained with symbol replacement methods that preserve protected symbols versus methods that treat all symbols equally. The test set preserves the protected symbols. We follow the Unmixed setup, where theorem-proof pairs from the same paper must appear in the same split.

We focus on the probability theory domain. Focusing on a single domain enables us to construct a list of protected symbols more precisely. Our list consists of the P (probability measure), E (expected value), V (variance), σ (standard deviation and covariance), and ρ (correlation) symbols.[14] Table 6 shows that training the SCRATCHBERT-Local-Local model using the Partial+P replacement method results in slightly better results. We present only a subset of our results for brevity; the pattern re-occurs with all symbol replacement methods.

9.5 Qualitative Analysis

To study which tokens affect our model predictions, we use LIME (Ribeiro et al., 2016), a method for calculating feature importance. We examine SCRATCHBERT-Local-Local trained with the Conservation setup and with Full replacement. Both are applied to original test examples. We observe that the Conservation SCRATCHBERT-Local-Local model heavily relies on the mathematical tokens and barely benefits from the text ones. In contrast, the SCRATCHBERT-Local-Local model that was trained in the full symbol replacement setup strongly relies on textual tokens with mathematical meaning, such as *module*, *supplement*, and *semistable*. We visualize that in Figure 2.

[14] We relied on Wikipedia, https://tinyurl.com/2c3kwsfx, for creating the protected symbols list.

Source \ Target	Symbol Replacement			
	Conservation		Partial+P	
	MRR	Acc	MRR	Acc
Conservation	**69.26**	**59.59**	27.9	18.29
Partial	61.36	51.72	54.06	42.67
Partial+P	62.1	51.92	**55.92**	**45.23**
Full	53.63	42.08	52.85	41.4
Full+P	56.27	45.13	**55.92**	44.84

Table 6: Controlled cross-replacement levels performance for the SCRATCHBERT-Local-Local model. Both train and test sets are curated from the probability theory domain. +P next to a symbol replacement method means that Protected symbols are not being replaced.

10 Conclusion

We developed a bilinear similarity model and a large dataset (MATCH) for a task focusing on the domain of mathematical research articles. The task consists in matching a proof to a mathematical statement. We proposed two ways to train and inference with our model and dataset: local matching and global matching. We assessed the difficulty of the task with several pre-trained encoders, demonstrating the importance of the vocabulary support for these models. Further assessment relies on using a symbol replacement procedure, which helps test the type of mathematical reasoning the encoders can perform. While our model performs well on this task, we observe through the symbol replacement procedure that the model makes a relatively shallow use of the text and formulae to obtain this performance.

Limitations

Our work has three main limitations. First, we aim to simulate a setup where the same author did not write both a theorem and its corresponding proof. We reduce the intersection size of symbols between the statement and the proof, which leads to more challenging setups. In practice, authors and mathematical communities within fields differ in their use of notation and their writing style (creating *mathematical language dialects*). Such overall dialect

cannot be altered using simple rule-based methods. We leave it for future work to explore a full MIR setup for our task that takes this into consideration.

Second, due to computational limitations, we could not explore the full potential of our global training method. Our GPUs cannot handle large batch sizes for large models such as MATHBERT and SCRATCHBERT. We use NVIDIA V100 GPUs that allow us to experiment with a batch size of 16 for MATHBERT and SCRATCHBERT, compared to 60 with NPT.

Third, while our symbol replacement method provides a coarse way to test the language model use of the symbols and text in mathematical articles, it presents cases in which the replacement is not precise. These cases arise because the use of symbols in mathematical language is rich and context-dependent (for example, while π often refers to the pie constant, it might also refer to a tuple-projection function or a permutation). We partially address that in §9.4.

Acknowledgments

We thank the reviewers for their helpful comments, and Marcio Fonseca for feedback on an earlier draft. We also thank Richard Zanibbi for a discussion about aspects of this work. The experiments in this paper were supported by a compute grant from the Edinburgh Parallel Computing Center (Cirrus) and another compute grant for the Baskerville service at the University of Birmingham.

References

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online. Association for Computational Linguistics.

Maximin Coavoux and Shay B Cohen. 2021. Learning to match mathematical statements with proofs. *arXiv preprint arXiv:2102.02110*.

Pankaj Dadure, Partha Pakray, and Sivaji Bandyopadhyay. 2021. Embedding and generalization of formula with context in the retrieval of mathematical information. *Journal of King Saud University - Computer and Information Sciences*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Deborah Ferreira and André Freitas. 2020. Premise selection in natural language mathematical texts. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7365–7374, Online. Association for Computational Linguistics.

Deborah Ferreira and André Freitas. 2021. STAR: Cross-modal [STA]tement [R]epresentation for selecting relevant mathematical premises. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3234–3243, Online. Association for Computational Linguistics.

Deborah Ferreira, Mokanarangan Thayaparan, Marco Valentino, Julia Rozanova, and Andre Freitas. 2022. To be or not to be an integer? encoding variables for mathematical text. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 938–948, Dublin, Ireland. Association for Computational Linguistics.

Andrew V. Goldberg and Robert Kennedy. 1995. An efficient cost scaling algorithm for the assignment problem. *Mathematical Programming*, 71(2):153–177.

Mihai Grigore, Magdalena Wolska, and Michael Kohlhase. 2009. Towards context-based disambiguation of mathematical expressions. In *The joint conference of ASCM 2009 and MACIS 2009. 9th international conference on Asian symposium on computer mathematics and 3rd international conference on mathematical aspects of computer and information sciences, Fukuoka, Japan, December 14–17, 2009. Selected papers.*, pages 262–271. Fukuoka: Kyushu University, Faculty of Mathematics.

Michael Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13:307–361.

Jesse Michael Han, Jason Rute, Yuhuai Wu, Edward W. Ayers, and Stanislas Polu. 2021. Proof artifact co-training for theorem proving with language models. *ArXiv preprint*, abs/2102.06203.

R. Jonker and A. Volgenant. 1987. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340.

Giovanni Yoko Kristianto, Minh quoc Nghiem, Yuichiroh Matsubayashi, and Akiko Aizawa. 2012. Extracting definitions of mathematical expressions in scientific papers. In *In JSAI*.

Harold W. Kuhn. 1955. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97.

Martin Líška, Petr Sojka, Michal Růžička, and Petr Mravec. 2011. Web Interface and Collection for Mathematical Retrieval: WebMIaS and MREC. In *Towards a Digital Mathematics Library.*, pages 77–84, Bertinoro, Italy. Masaryk University.

Behrooz Mansouri, Shaurya Rohatgi, Douglas Oard, Jian Wu, C.Lee Giles, and Richard Zanibbi. 2019. Tangent-cft: An embedding model for mathematical formulas. In *ICTIR '19: Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*.

C. o. P. a. L. o. t. Mathematical Sciences. 2014. Developing a 21st Century Global Library for Mathematics Research. *ArXiv e-prints*, abs/1404.1905.

Minh Nghiem Quoc, Keisuke Yokoi, Yuichiroh Matsubayashi, and Akiko Aizawa. 2010. Mining coreference relations between formulas and text using Wikipedia. In *Proceedings of the Second Workshop on NLP Challenges in the Information Explosion Era (NLPIX 2010)*, pages 69–74, Beijing, China. Coling 2010 Organizing Committee.

Robert Pagel and Moritz Schubotz. 2014. Mathematical language processing project. *CoRR*, abs/1407.0167.

Shuai Peng, Ke Yuan, Liangcai Gao, and Zhi Tang. 2021. Mathbert: A pre-trained model for mathematical formula understanding. *ArXiv preprint*, abs/2105.00377.

Bartosz Piotrowski and Josef Urban. 2019. Guiding theorem proving by recurrent neural networks. *ArXiv preprint*, abs/1905.07961.

Stanislas Polu and Ilya Sutskever. 2020. Generative language modeling for automated theorem proving. *ArXiv preprint*, abs/2009.03393.

B. T. Polyak and A. B. Juditsky. 1992. Acceleration of stochastic approximation by averaging. *SIAM J. Control Optim.*, 30(4):838–855.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144. ACM.

Ulf Schöneberg and Wolfram Sperber. 2014. POS tagging and its applications for mathematics. *CoRR*, abs/1406.2880.

Moritz Schubotz, Alexey Grigorev, Marcus Leich, Howard S. Cohl, Norman Meuschke, Bela Gipp, Abdou S. Youssef, and Volker Markl. 2016. Semantification of identifiers in mathematics for better math information retrieval. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*, pages 135–144. ACM.

Shun Shao, Yftah Ziser, and Shay B. Cohen. 2023. Erasure of unaligned attributes from neural representations. *arXiv preprint arXiv:2302.02997*.

Jia Tracy Shen, Michiharu Yamashita, Ethan Prihar, Neil T. Heffernan, Xintao Wu, and Dongwon Lee. 2021. Mathbert: A pre-trained language model for general NLP tasks in mathematics education. *ArXiv preprint*, abs/2106.07340.

Heinrich Stamerjohanns, Michael Kohlhase, Deyan Ginev, Catalin David, and Bruce Miller. 2010. Transforming large collections of scientific publications to xml. *Mathematics in Computer Science*, 3(3):299–307.

Yiannos Stathopoulos, Simon Baker, Marek Rei, and Simone Teufel. 2018. Variable typing: Assigning meaning to variables in mathematical text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 303–312, New Orleans, Louisiana. Association for Computational Linguistics.

Yiannos Stathopoulos and Simone Teufel. 2015. Retrieval of research-level mathematical information needs: A test collection and technical terminology experiment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 334–340, Beijing, China. Association for Computational Linguistics.

Yiannos Stathopoulos and Simone Teufel. 2016. Mathematical information retrieval based on type embeddings and query expansion. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2344–2355, Osaka, Japan. The COLING 2016 Organizing Committee.

A. Volgenant. 1996. Linear and semi-assignment problems: A core oriented approach. *Computers & Operations Research*, 23(10):917 – 932.

Sean Welleck, Jiacheng Liu, Ronan Le Bras, Hannaneh Hajishirzi, Yejin Choi, and Kyunghyun Cho. 2021. Naturalproofs: Mathematical theorem proving in natural language.

Magdalena Wolska, Mihai Grigore, and Michael Kohlhase. 2011. Using discourse context to interpret object-denoting mathematical expressions. In *DML 2011 - Towards a Digital Mathematics Library, Proceedings*.

Richard Zanibbi, Akiko Aizawa, Michael Kohlhase, Iadh Ounis, Goran Topic, and Kenny Davila. 2016. NTCIR-12 mathir task overview. In *Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies, National Center of Sciences, Tokyo, Japan, June 7-10, 2016*.

A Details on Dataset Construction

As mentioned in §4, we use the MREC corpus to extract statement-proof pairs.

Statement-proof Identification For each XML article (corresponding to a single arXiv article), we extract pairs of consecutive `<div>` tags such that: (i) the `class` attribute of the first `div` node contains the string `"theorem"`; (ii) the `class` attribute of the second `div` node is the string `"proof"`. Articles that do not contain such pairs of tags are discarded, as well as articles that are not written in English (representing 143 articles in French, 11 in Russian, 5 in German, 2 in Portuguese and 1 in Ukrainian), as identified by the `polyglot` Python package.[15]

In the remaining collection of pairs of statements and proofs, we filter out pairs for which either the statement or the proof is too short.[16] Indeed, the short texts were often empty (only consisting of a title, e.g. "5.26 Lemma."), which we attribute to the noise inherent to the conversion to XML, or not self-contained. In particular, we identified several prototypical cases:

- Omitted (or easy) proofs contain usually a single word ('omitted', 'straightforward', 'well-known', 'trivial', 'evident'), but are sometimes more verbose ('This is obvious and will be left to the readers').
- Proofs that consist of a single reference to
  - An appendix ('See Appendix A');
  - Another theorem ('This follows immediately from Proposition 4.4 (ii).');
  - The proof method of another theorem ('Similar to proof of Lemma 6.1')
  - Another article ('See [BK3, Theorem 4.8].');
  - Another part of the article ('The proof will appear elsewhere.', 'See above.', 'Will be given in section 5.').

Filtering on the number of tokens also excludes self-contained short proofs, such as 'Take $Q' = ph_i - p_i$.' However, such proofs were very infrequent on manual inspection of the discarded pairs (2 in a manually inspected random sample of 100 discarded proofs).

[15] www.github.com/aboSamoor/polyglot/

[16] We used a minimum length of 20 tokens for both statements and proofs, based on a manual inspection of the shortest examples. We also exclude proofs and statements longer than 500 tokens.

Preprocessing: Linearizing Equations Mathematical formulae in the XML articles are enclosed in a `<math>` markup tag, that materializes the switch to the MathML format, and whose internal structure represents the formula as an XML tree. As a preprocessing step, we linearize each formula to a raw sequence of strings.

In MathML, an equation can be encoded in a content-based (semantic) way or in a presentational way, using different sets of markup tags. We first convert all MathML trees to presentational MathML using the XSL stylesheet from the Content MathML Polyfill repository.[17] Then we perform a depth-first search on each tree rooted in a `<math>` tag to extract the text content of the whole tree.

During this preprocessing, we tested several processing choices:

- **Font information**. In mathematical discourses, fonts play an important role. Their semantics depend on conventions shared by researchers. If both x and $\mathbf{x}$ appear in the same article, they are most likely to represent different mathematical objects, e.g. a scalar and a vector. Therefore, we use distinct symbols for tokens that are in distinct fonts.
- **Math-English ambiguity**. Some symbols can be used both in natural language text and in formulae. For example, 'a' can be a determiner in English, or a variable name in a formula. To avoid increasing ambiguity when linearizing formula, we type each symbol (as math or text) to make the mathematical vocabulary completely disjoint from the text vocabulary.

Both these preprocessing steps had a beneficial effect on the baselines in preliminary experiments.

B Distribution of Proof-Statement Assignments

Table 7 depicts the cumulative distribution of proofs and the number of statements they are assigned to.

C Symbol Replacement Details

We follow the following rules when replacing symbols:

[17] https://github.com/fred-wang/webextension-content-mathml-polyfill

Lemma 3.2. Let M be a module and H a local submodule of M. Then H is a supplement of each proper submodule $K \leq M$ with $H + K = M$.
Proof. Since K is a proper submodule of M and $K + H = M$, we have $K \cap H$ is a proper submodule of H. Therefore $K \cap H \ll H$, since H is local. That is, H is a supplement of K in M.
(https://arxiv.org/pdf/0810.0041.pdf)

(a) Example statement/proof 1 - Symbol conservation

Lemma 3.2. Let M be a module and H a local submodule of M. Then H is a supplement of each proper submodule $K \leq M$ with $H + K = M$.
Proof. Since K is a proper submodule of M and $K+H = M$, we have $K \cap H$ is a proper submodule of H. Therefore $K \cap H \ll H$, since H is local. That is, H is a supplement of K in M.
(https://arxiv.org/pdf/0810.0041.pdf)

(b) Example statement/proof 1 - Full symbol replacement

Lemma 4.1.9. If $\mathscr{F}$ is a μ-semistable $\mathscr{X}$-twisted sheaf of rank r then dim Hom ($\mathscr{F}$, $\mathscr{F}$) $\leq r^2$.
Proof. Any endomorphism of $\mathscr{F}$ must preserve the socle (see Lemma 1.5.5ff of [4]); moreover, the quotient $\mathscr{F}$ / Soc($\mathscr{F}$) is also semistable. The result follows by induction from the polystable case, which itself follows immediately from the fact that stable sheaves are simple.
(https://arxiv.org/pdf/0803.3332.pdf)

(c) Example statement/proof 2 - Symbol conservation

Lemma 4 .1.9. If $\mathscr{F}$ is a μ-semistable $\mathscr{X}$-twisted sheaf of rank r then dim Hom ($\mathscr{F}$, $\mathscr{F}$) $\leq r^2$.
Proof. Any endomorphism of $\mathscr{F}$ must preserve the socle (see Lemma 1.5.5ff of [4]); moreover, the quotient $\mathscr{F}$ / Soc($\mathscr{F}$) is also semistable. The result follows by induction from the polystable case, which itself follows immediately from the fact that stable sheaves are simple.
(https://arxiv.org/pdf/0803.3332.pdf)

(d) Example statement/proof 2 - Full symbol replacement

Figure 2: LIME visualizations for the model that was trained in the symbol conservation setup (a and c) and their corresponding LIME visualizations for the model that was trained in the full symbol replacement setup (b and d). The LIME "match" class supporting features are colored in orange, and the "mismatch" is in blue. The darker the color, the higher (in absolute value) the feature importance.

Statements	Proofs	%
≥ 20	7	0.0
≥ 10	80	0.2
≥ 5	1027	1.9
≥ 2	11949	22.6
$= 1$	19531	37.0
< 1	21275	40.3

Table 7: Cumulative distribution of proofs in the development set, by number of statements to which they are assigned with the local decoding method.

- Only the proof symbols are being replaced as there is no need to replace both statement and proof symbols.
- We replace symbols only if they appear in both the statement and the proof.
- If the symbol a, for example, is mapped to b, we will map A to B, and vice versa.
- We do not replace the double-struck letters, e.g., $\mathbb{R}$, since they usually represent fields and constant. We do not replace standard constant symbols such as π.

D Prediction Visualization Examples

We provide a LIME visualization of several mathematical statements in Figure 2.