To be or not to be an Integer? Encoding Variables for Mathematical Text

Deborah Ferreira[1], **Mokanarangan Thayaparan**[1,2], **Marco Valentino**[1,2], **Julia Rozanova**[1], **Andre Freitas**[1,2]
Department of Computer Science, University of Manchester, United Kingdom [1]
Idiap Research Institute, Switzerland [2]
deborah.ferreira@manchester.ac.uk

Abstract

The application of Natural Language Inference (NLI) methods over large textual corpora can facilitate scientific discovery, reducing the gap between current research and the available large-scale scientific knowledge. However, contemporary NLI models are still limited in interpreting mathematical knowledge written in Natural Language, even though mathematics is an integral part of scientific argumentation for many disciplines. One of the fundamental requirements towards mathematical language understanding is the creation of models able to meaningfully represent variables. This problem is particularly challenging since the meaning of a variable should be assigned exclusively from its defining type, i.e., the representation of a variable should come from its context. Recent research has formalised the variable typing task, a benchmark for the understanding of abstract mathematical types and variables in a sentence. In this work, we propose VarSlot, a Variable Slot-based approach, which not only delivers state-of-the-art results in the task of variable typing, but is also able to create context-based representations for variables.

1 Introduction

The articulation of mathematical arguments is a fundamental part of scientific reasoning and communication. Across many scientific disciplines, expressing relations and inter-dependencies between quantities is at the centre of its argumentation. One of the particular linguistic elements used for such argumentation is *variables*: they are essential for expressing complex mathematical ideas, allowing scientists to refer to a set of values compactly and rigorously. Given the essential nature of variables, models that perform inference over scientific and mathematical text should be able to meaningfully represent and leverage such elements to understand mathematical language and improve downstream inference performance.

While there is still debate on a universally accepted definition for variables, their functional aspects within mathematical text are well established. A variable is usually defined as a symbol standing as a referent for a set consisting of at least two elements (Philipp, 1992). Given their nature of abstract and dynamic referents, two aspects make the representation of variables particularly challenging in the field of NLP: (i) The meaning of a variable is exclusively determined by its context (Schoenfeld and Arcavi, 1988), a variable carries no meaning when considered in isolation, behaving unlike any word in English; (ii) The same variable symbol (e.g., the variable x) can be reused in an unlimited number of sentences and expressions, possibly assuming different meanings and referring to different sets of values while keeping the same name. While a similar problem can also be found for the representation of words (i.e., word sense disambiguation), the scale of ambiguity is more marked, due to the fact that variable names, unlike words, are not grounded a priori to any particular set of concepts.

In order to understand mathematical text, the meaning of variables (i.e., their type) needs to be implicitly or explicitly inferred at some point from the text. Identifying and qualifying the binding between variables and their types, therefore, is crucial for reasoning with mathematical text, given that any form of inference on a variable is fundamentally constrained by the possible values it can take, and those values are uniquely determined by its type. For example, we can only correctly predict the entailment between two sentences containing variables if we infer their types beforehand.

As a benchmark for variable comprehension in mathematical text, the variable typing task (Stathopoulos et al., 2018) requires finding the connections between variables and their respective types in a sentence. Despite its importance for scientific inference, the task is still widely unexplored. Most of the existing work focusing on the representation of mathematical elements, in fact, has been carried out in a non-textual setting, where the mathematical elements are represented independently from any textual content (Aizawa and Kohlhase, 2021). While some expressions and formulas are universal enough to be encoded without context (e.g., Pythagoras theorem), variables have no meaning in isolation and different symbols can be arbitrarily chosen to indicate variables with the same meaning. Therefore, an important principle for designing and evaluating a variable representation is that such a representation should be agnostic to the specific symbols adopted as referents in the text – i.e., the variable names. This is because the variable names contained in a generic passage can be opportunely renamed without altering the sentences' meaning. For instance, a robust variable representation should be able to encode the sentences "Let x be an integer" and "Let y be an integer" in exactly the same way, if they are inserted in the same context.

However, while this characteristic seems to be intrinsic in the nature of variables, it has been largely ignored by current evaluation frameworks, where there is no agnostic way to verify the quality of the generated variable representation independently of their surface form. To move a step forward towards more robust and generalisable representations, this work proposes a new testing and modelling framework based on the property of *generalisation to variable renaming*. Specifically, we define a model to be generalisable to variable renaming if the model's performance does not decrease when renaming variables in the test set with variable names never seen during training. Through testing such a property, since the renaming of variables does not alter the context or meaning of the sentences, we are verifying whether the context is correctly moulding the variable encoding and, at the same time, whether the performance is not due to overfitting to the surface form of the text.

To address and study generalisation to variable renaming, this work proposes **VarSlot** (Variable Slot), a model for variable typing that represents variables in a surface agnostic way. To achieve a generalisable representation, VarSlot initialises variables as blank slots and employs a multi-slot mechanism, extending from slot attention (Locatello et al., 2020), to iteratively specialise the representation. Specifically, VarSlot leverages self-attention to conform the representation of each variable to its context (i.e., its surrounding words and other variables), where each surrounding element has a different influence on the final representation. Experiments adopting VarSlot to extend sentence embeddings based on Transformers (Devlin et al., 2019) demonstrate not only that the proposed framework is able to achieve state-of-the-art results on the variable typing task, but also, in contrast with previous work, that VarSlot allows for better generalisation to variable renaming. To the best of our knowledge, this is the first work that focuses on generalisation and robustness for variable representation in mathematical text, providing, at the same time, a critical analysis of large language models (LLMs) targeting the scientific domain. To summarise, the contributions of this paper are as follows:

- We propose a new evaluation framework for testing the property of generalisation to variable renaming;
- We systematically analyse the understanding of variables in large language models, demonstrating their limitations when handling variable renaming;
- We propose a state-of-the-art model for the variable typing task, demonstrating, at the same time, that it significantly outperforms large language models when analysing the property of generalisation to variable renaming;

2 Variable Typing problem

This work follows the same definition of a variable used in (Stathopoulos et al., 2018), considering as variables *simple expressions* composed of a single, possibly scripted, base identifier. The *variable typing task* requires the assignment between variables in the sentence and its respective types. We use the same setting as (Stathopoulos et al., 2018) for this task: given a sentence s with a pre-identified set of variables V and types T, the task is defined as a binary classification of all edges $V \times T$, where a positive edge means that the variable is assigned that type and negative otherwise. Figure 1 introduces an example of a mathematical statement, containing one variable b with type *persistence length*. The edge linking to this type is a positive one, while the one linking to type *chains* is a negative edge.

While the task carries some similarity to two other well-known tasks in NLP, *Coreference Res-*

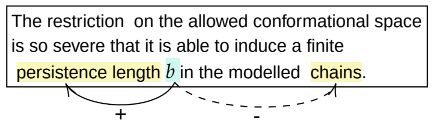

The restriction on the allowed conformational space is so severe that it is able to induce a finite persistence length b in the modelled chains.
+
-

Figure 1: Example of a mathematical statement containing one variable. This example shows two different edges (variable→type), with a positive and a negative binding.

olution and *Relation Extraction*, there are some fundamental differences. Variables have a particular behaviour, where there is a disconnection between the variable name and its meaning. Also, the variable typing task mainly involves intra-sentence dependencies and does not rely on a predicate-argument relation.

3 Our Approach: Variable Slots (VarSlot)

Following the recent literature (Stathopoulos et al., 2018), this work frames the variable typing problem as binary classification, where each variable is tested against all possible types. Given a sentence s from a mathematical text containing known variables $V = \{v_1, v_2, \ldots, v_{n_v}\}$ and types $T = \{t_1, t_2, \ldots, t_{n_t}\}$, VarSlot aims at finding a function such that $f(v_i, t_j) = 1$ if v_i has type t_j and $f(v_i, t_j) = 0$ otherwise. In this section we detail our approach, **VarSlot**, illustrated in Figure 2.

3.1 From types and expressions to hypotheses

A novel framing for the variable typing problem is proposed in this work. Instead of treating sentences, types and variables as different features of the problem (Stathopoulos et al., 2018), each sentence s_i is converted to a set of hypotheses $H_{s_i} = \{h_{(v_1,t_1)}, h_{(v_1,t_2)}, \ldots, h_{(v_{n_v},t_{n_t})}\}$ of size $n_v \times n_t$ by adding to the end of the sentence, the following phrase: "Then [variable] is of type [type]" where *variable* and *type* are replaced by the ones being evaluated at that instance. For example, if one wants to test if the variable x is of type *integer* in the sentence "Let x be an integer and y be a real", it can be converted to a hypothesis by adding "Then x is of type *integer*" after the initial sentence. This modification will allow our approach to leverage pre-trained sentence encoders, obtaining enriched representations.

3.2 Encoding the sentences

Previous research has shown that LLMs struggle to understand concepts such as numeracy (Spithourakis and Riedel, 2018) and solving math word problems (Piękos et al., 2021), but there is still no research on the representation of variables inside the mathematical text for such models, despite their higher performance for different inference tasks (Rogers et al., 2020).

We hypothesise that it is possible to leverage pre-trained sentence representations to obtain an initial encoding for each hypothesis. By using an encoder, each hypothesis h of size N is mapped to a representation $E \in \mathbb{R}^{N \times D}$, where each token is assigned a vector of size D, regardless of being a variable or a word. At this point, the representation is unable to distinguish between both modalities of elements.

3.3 Representing Variables with Multi-Slots

Variables represent a set of possible values, acting as a place-holder element for any possible value inside that set (Schoenfeld and Arcavi, 1988). This set of values is attached to the type corresponding to that variable; for example, if a variable is typed as an integer, we expect it to take values from that set only. A variable with the symbol x can refer to completely different sets depending on its context. Therefore, when representing a variable, its *surface form* (or symbol) should not interfere with its representation; the semantics of the variable should be guided exclusively by the defined type. We aim to approximate this behaviour by representing each variable using slots (Locatello et al., 2020). Slots use a common representational format, where each slot can store any object from the input, rendering it a suitable candidate for obtaining a latent representation of variables since the same variable name can take many different contexts and possible types. We extend previous work by designing an approach that combines the representations obtained from different slots (multi-slot).

Given a hypothesis $h_{(v_i,t_j)}$ containing the variables $V = \{v_1, \ldots, v_{n_v}\}$, a pre-trained sentence embedding is used to obtain a representation E for this hypothesis. In order to obtain a representation that can better distinguish from symbol and abstraction of the variables, we generate a new representation $E_{(v_i-)} \in \mathbb{R}^{N \times D}$ for each variable in the sentence, where the vectors representing the variables are all replaced by zeroes. For each variable, the representation obtained from the encoder is dropped, preserving only the other tokens. This step allows our model to learn the representation of

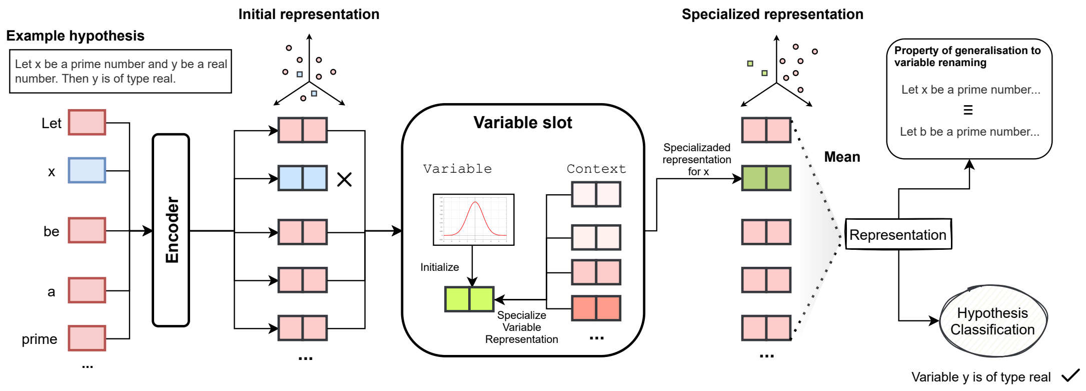

Example hypothesis
Let x be a prime number and y be a real number. Then y is of type real.
Initial representation
Let
x
be
a
prime
...
Encoder
Variable slot
Variable
Context
Initialize
Specialize Variable Representation
Specializaded representation for x
Specialized representation
Mean
Representation
Property of generalisation to variable renaming
Let x be a prime number...
≡
Let b be a prime number...
Hypothesis Classification
Variable y is of type real ✓

Figure 2: Our model takes as input a statement containing words and variables. These tokens are encoded using a pre-trained language model. Then, the variables' representation is enriched using slot attention. The final representation is obtained as the mean of all tokens. In the end, we obtain the classification of the hypothesis.

the variables based on their context and abstraction, diminishing the weight from its surface features.

A representation $e_{(v_i)} \in \mathbb{R}^D$ is obtained for each variable through iterative Scaled Dot-Product Attention. First, in order to obtain an initial representation, we initialise $e_{(v_i)}$ by sampling from a Gaussian distribution $\mathcal{N}(\mu, \sigma)$, with learnable parameters $\mu \in \mathbb{R}^D$ and $\sigma \in \mathbb{R}^D$. This will generate an empty slot, which will be iteratively conformed into a representation for the variable v_i.

Previous work has shown that having a separated representation for mathematical elements and words can be beneficial for performing inference over mathematical text (Ferreira and Freitas, 2021). We hypothesise that allowing our model to learn a new representation for variables will naturally separate it from other elements.

We apply the linear transformation k and v over $E_{(v_i-)}$ and q over $e_{(v_i)}$ to compute the Scaled Dot-Product Attention (Vaswani et al., 2017):

$$\texttt{Att} = \texttt{softmax}(\frac{q(e_{(v_i)})k(E_{(v_i-)})^T}{\sqrt{D}})v(E_{(v_i-)}) \tag{1}$$

The obtained attention is applied to a Gated Recurrent Unit (Cho et al., 2014) with hidden size D and transformed with a multi-layer perceptron (MLP) with ReLU activation in order to obtain the new value for $e_{(v_i)}$. This process is repeated for T iterations, for each variable in the sentence.

After obtaining the representation for each variable $e_{(v_i)}$, we need to match it again with the original representation. This is achieved by mapping the obtained variable representations to their original positions in $E_{(v_i-)}$. The final matrix is encoded by a BiLSTM layer, obtaining a final enriched representation $\mathcal{E}$ for the sentence. The algorithm describing this process can be found in Appendix C.

3.4 Classifying hypotheses

Finally, we obtain a representation for the hypothesis as a single vector of dimension D by computing the mean over the rows of $\mathcal{E}$. In order to obtain the classification for each hypothesis, we use a final linear layer, with as training objective function the *Binary Cross-Entropy Loss*.

4 Experiments

This section presents the experiments performed to evaluate the performance of VarSlot for the variable typing task. The models and datasets used are made available in our repository[1]. As the encoder for the model, we use the Sentence Transformers (Reimers and Gurevych, 2019) version of SciBERT (Beltagy et al., 2019) pre-trained for NLI tasks (SciBERT-NLI)[2]. We compare our approach with previous research and standard pre-trained language models. The evaluation is conducted in two different settings:

1. **Classic Variable Typing**: This setting represents the canonical evaluation of variable typing task, as it was initially described in (Stathopoulos et al., 2018).
2. **Variable Typing with Renaming**: In order to evaluate the model's ability to abstract from the surface form of the variables and generalise to variable renaming, we replace all the

[1]Anonymous repository.

[2]The model `gsarti/scibert-nli` from Hugging Face is used.

variables in the test set with new symbols, unseen in the Train and Dev set without modifying the sentence's meaning.

4.1 Dataset

The dataset used in this work to evaluate the performance of the proposed model is the Variable Typing Dataset (Stathopoulos et al., 2018). This dataset was manually curated and annotated from mathematical statements contained in scientific papers.

For the **Renaming** setting, the Train and Dev set is the same as the previous setting, but the Test set contains only variable names in the format $x_1, x_2, x_3, ..., x_n$. The Test set in both settings has the same hypotheses with identical semantic meaning but different variable names. For example, the fragment "Let b be ..." becomes "Let x_1 be ..." in the Renaming setting. For reproducibility purposes, we make this expanded dataset available in our repository.

4.2 Baselines

We compare our approach to the following models:

- **(Stathopoulos et al., 2018)**: This architecture is based on a Bidirectional LSTM. The model uses one string feature, which is referred to as *supertype*. If the token is a type, then this feature is the string key of the embedding vector of its supertype or *NONE* otherwise. These features are mapped to a separate embedding space and then concatenated with the word embedding to form a single task-specific word representation.
- **BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), SciBERT (Beltagy et al., 2019) and MathBERT (Peng et al., 2021)**: To assess the understanding of variables in pre-trained language models, we use the mentioned models as baselines for the Variable Typing task. For all models we use the `base` and `uncased` version. The models are fine-tuned as a classic NLI model, using a `[SEP]` token to separate the original sentence with our new typing sentence. All models are fine-tuned with batch size of 32, for 10 epochs. The full list of used hyper-parameters can be found in the Appendix A.

4.3 Quantitative results

We present the results for the Variable Typing task in both the classic setting (Table 1) and the renaming setting (Table 2). We include here our approach with $T = 3$ and $T = 2$. We will start our discussion with the results from the classic setting and then move to the renaming setting.

4.3.1 Classic Setting

Considering first the classic setting (Table 1), we can observe that both BERT and RoBERTa achieve good results, outperforming the approach proposed by (Stathopoulos et al., 2018), which was designed explicitly for variable typing and uses specific typing embedding. Such performance does not come as a surprise since, as discussed previously, variable typing carries some similarity to tasks that these models have excelled in the past. However, such performance does not imply that these models have a good understanding of the meaning of variables; most likely, they are leveraging the syntactic knowledge they possess to connect types and variables.

Model	**Test (Classic)**		
	P	**R**	**F1**
Stathopoulos et al. (2018)	83.10	74.70	78.90
BERT	77.8	82.98	80.31
RoBERTa	82.32	80.13	81.21
MathBERT	81.85	73.76	77.59
SciBERT	77.26	87.54	82.08
VarSlot (T=3)	83.18	81.36	**82.26**
VarSlot (T=2)	83.95	79.08	81.44

Table 1: Comparison of our approach with different baselines for Variable Typing for classic setting. We present the values for precision (P), recall (R) and F1-score.

MathBERT and SciBERT are models specialised in scientific and mathematical knowledge. While SciBERT is pre-trained using corpora from several scientific disciplines, MathBERT was exclusively trained on mathematical text. We initially expected both models to excel on this task since they have been previously exposed to mathematical notation. However, as seen from our results, MathBERT was the worst-performing model from our baselines, while SciBERT was the best performing one. While we cannot establish the reasons for MathBERT's lower performance, since this is outside the scope of this work, we hypothesise that training a model with a large amount of mathematical notation without explicitly designing an encoding which reflects its abstract semantics can be damaging to a model's performance. For example, many variable names are reused across different mathematical disciplines, and as we will see in the renaming setting, most models cannot abstract variable meaning from their surface form. SciBERT has been exposed to a smaller mathematical corpus, and usually inside non exclusively mathematical contexts (e.g., computer science papers). Therefore, it is able to achieve higher performance. Given the results obtained from SciBERT, we decided to use the Sentence-Transformers version of this model as our encoder.

We can see that for $T = 3$, VarSlot can outperform all of the models for the classic setting. Even though VarSlot uses SciBERT as part of its model, we can still see an improvement when combining it with multi-slots. For $T = 2$ we still obtain competitive results, being outperformed only by SciBERT.

4.3.2 Renaming setting

In this setting (Table 2), we have the same sentences as in the previous setting, but the variables in the Test split have now different names. Considering the semantic of variables, the previous results should not change, considering the sentence's meaning remains untouched; only the surface of the variables have been altered. However, the obtained results prove otherwise. For all the approaches, there is a decrease relative to the results obtained in the classic setting. Such results hint at the fact that the models still do not encode the expected variable behaviour.

Model	Test (Renaming)				Average (C+R)
	P	**R**	**F1**	**Decrease**	**F1**
BERT	54.00	78.23	63.89	20.44%	72.1
RoBERTa	55.43	68.34	61.21	24.62%	71.21
MathBERT	54.37	44.29	48.82	37.07%	63.21
SciBERT	30.25	93.15	45.67	44.35%	63.88
VarSlot (T=3)	60.86	73.47	66.58	19.06%	74.42
VarSlot (T=2)	69.80	79.04	**71.86**	**11.76%**	**76.65**

Table 2: Results for VarSlot and baselines for Variable Typing for renaming setting. We include here also the decrease in score relative to the classic setting and the average score for both classic and renaming setting.

Looking at the obtained results, we can notice that the results obtained using SciBERT suffers from a more remarkable decrease, with a 44.35% (82.08→45.67) decrease in performance for this new setting, even though it was the best performing model for the previous setting. These results hint that SciBERT is overfitting to the names of the variables instead of abstracting from its symbols and adapting from context. A significant decrease can also be seen for MathBERT. The best performing baseline for this setting is BERT, which is likely, as previously mentioned, using syntactical cues to obtain the correct types.

We can see that VarSlot with $T = 2$ is more robust to this transition, having a less prominent decrease when compared to the other results (81.44→71.86), while we see a slight larger decrease for $T = 3$. The results suggest that our model better understands the difference in behaviour between variables and words and the surface agnostic aspect of variables, not over-fitting as much as the baselines for names of the variables. The results also hints that our model can correctly obtain the meaning of the variables from the context.

4.4 Robustness to substitution

In a more practical scenario, a model should harmonise a combination of the classic and renaming settings. During inference time, a mixture of seen and unseen variables is likely to be present. In this section, we evaluate the robustness of the model for the duplication of hypotheses with substituted variable names.

We add n repetitions of the same hypothesis in the test set for this experiment but with different variable names. For $n = 0$, the Test set is the same as the classic setting. For $n > 0$, we follow the same procedure as the renaming setting; however, we change the letter being used when adding more repetitions. For instance, for $n = 1$ the variables have format $x_{[variable\_index]}$, while for $n = 2$ we use the format $y_{[variable\_index]}$. For $n = 10$, we will have the same hypothesis appear ten times, but all with different variable names.

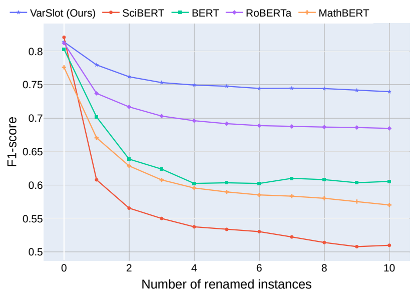


Figure 3: F1 score obtained for different models when adding the same hypotheses with different variable name substitutions.

This test allows us to compare the robustness of each model to the addition of different variables names. Figure 3 presents the performance of the different models in this task, considering the F1 score and number of added substitutions n. For $n = 0$ the results are the same as the ones present in Table 1.

We can observe that all the models suffer degradation in the performance as we add more duplicated modified hypotheses. While the models start at a close point, we can notice that as we increase the value of n, the gap between the proposed approach and the others models increases. Again, we note a big decrease in SciBERT's performance. The performance obtained for VarSlot shows how using multi-slots for representation of variables can increase the robustness of the model for variable name substitution through the test set.

4.5 Ablation Studies

In order to establish what components are impacting on VarSlot comprehension of variables, we perform different ablation studies. Table 3 presents the F1 score for different tests. The first row shows the results for considering only the encoder of our approach (SciBERT-NLI), removing the enriched variable representations and fine-tuning for the classification task. We can notice that our approach improves on top of the sentence representation, with significantly better results on the Renaming setting, showing that multi-slots plays a crucial role on that task.

Ablation	**Classic**	**Renaming**
1. Encoder only (SciBERT-NLI)	80.92	58.88
2. Encoder only (BERT-NLI)	80.67	66.63
3. Using BERT-NLI as encoder	81.24	69.19
4. VarSlot ($T = 1$)	79.52	68.03
5. VarSlot ($T = 2$)	81.44	**71.86**
6. VarSlot ($T = 3$)	**82.26**	66.58
7. VarSlot ($T = 4$)	80.42	65.88

Table 3: Comparison of our approach for the different settings.

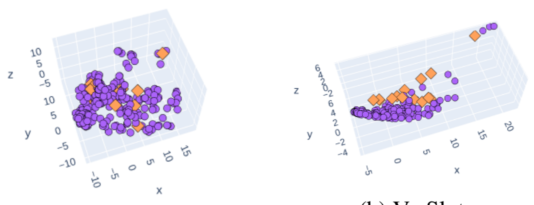
We also tested the generalisability of our approach to different encoders. Row 2 presents the results for the hypothesis classification only using BERT-NLI[3], and Row 3 presents our approach combined with BERT-NLI. We found that by using BERT-NLI instead of SciBERT-NLI for encoding our hypothesis, we could still observe an increase in performance for both classic and renaming settings. The proposed model consistently increases the understanding of variables for different sentence representation models.

The number of iterations T can also have an impact on the generalisation abilities of VarSlot. Rows 4-7 presents how its performance changes as the number of iterations increases. We can notice that we can obtain best results for the renaming setting with $T = 2$, while the best performance for the classic setting comes for $T = 3$. Our model requires a few iterations to conform the variable representation into the correct shape. When adding more iterations, VarSlot is more likely to overfit, leading to a degradation in performance with increasing number of iterations.

4.6 Qualitative Analysis

Previous work (Ferreira and Freitas, 2021) has found that having separate embedding spaces for mathematical elements and natural language when representing mathematical text can be beneficial for performance in other mathematical language processing tasks. We initially hypothesised that by allowing our model to learn the representation of variables, such separation would naturally happen when starting from a Gaussian initialised slot.

In order to verify if the model can discriminate between variables and natural language, we compare the embedding of words obtained from SciBERT-NLI (fine-tuned for the Variable Typing task) and our approach. We selected ten random sentences from the test set, obtained their embedding using both and reduced them to three dimensions using Principal component analysis[4]. The results are presented in Figure 4.



(a) SciBERT-NLI

(b) VarSlot

Figure 4: Embedding of the words and variables obtained from the test set for SciBERT-NLI and VarSlot.

The embeddings for the variables are represented here using the yellow diamond. From the figure,

[3]Model obtained from Hugging Face: `bert-base-nli-mean-tokens`

[4]We use the PCA implementation from `Scikit-learn` library with the default parameters.

we can immediately observe that our model easily distinguishes between words and variables, creating a clear separation. The same is not observed for SciBERT-NLI: there is no sharp separation between words and variables, suggesting that the model might not recognise the distinct semantic behaviour.

We also designed a probing test to quantify which model generates a more separable representation for variables. We obtained the representations for every token present in the test set, generating a representation for each word and variable using VarSlot and SciBERT-NLI. Then, given each representation, we trained a Linear Model to classify these representations into variables or non-variables. For the Linear Model, we use the standard configurations from the Probe-Ably[5] (Ferreira et al., 2021) probing framework. The results can be observed in Figure 5. In terms of accuracy, we can notice that the representation generated by our model allows for an easier disentanglement between variables and words, with a Linear model achieving maximum accuracy for linear models with different complexity (evaluated in terms of the norm). Following the classic probing protocol (Hewitt and Liang, 2019; Rozanova et al., 2021), we also provide the results for the Probing selectivity in Appendix B.

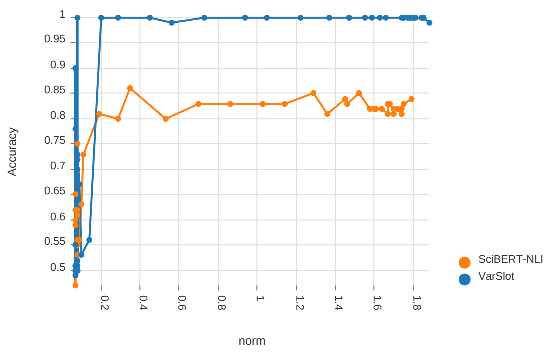


Figure 5: Results for the probing task, where we compare the representations generated from SciBERT-NLI (yellow) and VarSlot (blue).

Such finding reinforces the idea that variables requires a specialised representation. The slots for variables could have potentially relearned to replicate the original representations obtained from the encoder, however, a natural separation has been obtained between variables and words without explicitly being trained for that. This hints to the fact that such separation is beneficial for a better performance on tasks involving mathematical knowledge and models dealing with mathematical knowledge could benefit from explicitly defining such property.

5 Related Work

Even though mathematical text is often a crucial element of scientific discourse, the area of mathematical language processing is still widely unexplored (Ferreira and Freitas, 2020b,a). While initial efforts have focused on rule-based and bag of words-based approaches (Pagael and Schubotz, 2014; Schubotz et al., 2016; Alexeeva et al., 2020; Kristianto et al., 2012), recent work transitioned to supervised-learning methods. (Stathopoulos et al., 2018) presents three different models for addressing this problem, framing it as a binary classification between a variable and a type. The models proposed are an SVM model using features that are type and variable-centric, a ConvNet using pretrained embeddings to represent the input tokens and a Bidirectional LSTM model that takes as input the full sentence along with the pairs for classification. (Stathopoulos et al., 2018) shows that neural models vastly outperform models with manual feature engineering. The authors also introduce a dataset for the task.

6 Conclusion

In this paper, we introduced VarSlot, a model for solving the task of variable typing generating a more generalisable representation of variables. In order to evaluate the proposed encoding, with a particular emphasis on variable semantics, we propose a new setting for the variable typing task, where during inference type, the model is only exposed to new variable names. We observed that in this setting, there is a decrease in performance for all tested models; however, VarSlot was the most robust model. As future work, we leave the application of these specialised embeddings for different downstream tasks, similar to the work done in (Stathopoulos et al., 2018). We expect that generating better representations for variables will enable an increase in performance across different mathematical language inference tasks.

[5]The parameters used for the Linear Model and splits are found in the Probe-Ably repository `https://github.com/ai-systems/Probe-Ably`

References

Akiko Aizawa and Michael Kohlhase. 2021. Mathematical information retrieval. In *Evaluating Information Retrieval and Access Tasks*, pages 169–185. Springer, Singapore.

Maria Alexeeva, Rebecca Sharp, Marco A Valenzuela-Escárcega, Jennifer Kadowaki, Adarsh Pyarelal, and Clayton Morrison. 2020. Mathalign: Linking formula identifiers to their contextual natural language descriptions. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 2204–2212.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*.

Deborah Ferreira and André Freitas. 2020a. Natural language premise selection: Finding supporting statements for mathematical text. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2175–2182.

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

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, Julia Rozanova, Mokanarangan Thayaparan, Marco Valentino, and André Freitas. 2021. Does my representation capture X? probe-ably. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 194–201, Online. Association for Computational Linguistics.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

Giovanni Yoko Kristianto, Minh-Quoc Nghiem, Yuichiroh Matsubayashi, and Akiko Aizawa. 2012. Extracting definitions of mathematical expressions in scientific papers. In *Proc. of the 26th Annual Conference of JSAI*, pages 1–7.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. 2020. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems*, volume 33, pages 11525–11538. Curran Associates, Inc.

Robert Pagael and Moritz Schubotz. 2014. Mathematical language processing project. *arXiv preprint arXiv:1407.0167*.

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

Randolph A Philipp. 1992. The many uses of algebraic variables. *The Mathematics Teacher*, 85(7):557–561.

Piotr Piękos, Henryk Michalewski, and Mateusz Malinowski. 2021. Measuring and improving bert's mathematical abilities by predicting the order of reasoning. *arXiv preprint arXiv:2106.03921*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Julia Rozanova, Deborah Ferreira, Marco Valentino, Mokanrarangan Thayaparan, and Andre Freitas. 2021. Decomposing natural logic inferences in neural nli. *arXiv preprint arXiv:2112.08289*.

Alan H Schoenfeld and Abraham Arcavi. 1988. On the meaning of variable. *The mathematics teacher*, 81(6):420–427.

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*, pages 135–144.

Georgios Spithourakis and Sebastian Riedel. 2018. Numeracy for language models: Evaluating and improving their ability to predict numbers. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2104–2115.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

A Hyperparameters

The list of used hyper-parameters, which are shared across baselines and our approach are as follows:

- Seed: 42
- Batch size: 32
- Train Epochs: 10
- Learning Rate: 1e-5 (5e-5 for MathBERT)
- Gradient Accumulation Steps: 1
- Weight Decay: 0.0
- Adam Epsilon: 1e-8
- Warmup Steps: 0
- Max Grad Norm: 1.0

For fine-tuning all the models, we used 4 Tesla 16GB V100 GPUs.

B Probing Selectivity

The selectivity score, namely the difference in accuracy between the representational probe and a control probing task with randomised labels, serves as an indicator that the probe architectures used are not expressive enough to "memorise" unstructured labels. Ensuring that there is no drop-off in selectivity increases the confidence that we are not falsely attributing strong accuracy scores to the representational structure where they could have been explained by over-parameterized probes.

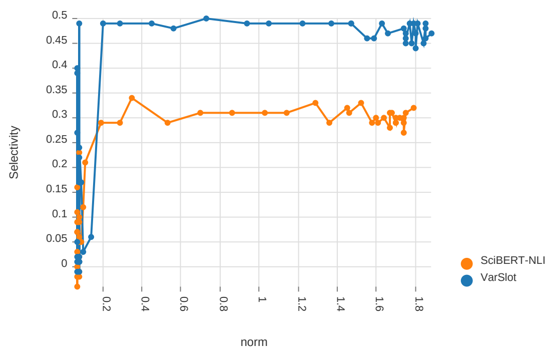


Figure 6: Results for the probing task, where we compare the representations generated from SciBERT-NLI (yellow) and VarSlot (blue), evaluating in terms of selectivity

Figure 6 presents the selectivity results for our probing task. We can notice that the selectivity remains stable for more complex probes, indicating that these are in fact, probes that are more reliable.

C VarSlot approach algorithm

This section presents Algorithm 1 used for obtaining the encoded representations of each hypothesis.

Algorithm 1: Learning the representation of mathematical statements with variables.

```
Input: Encoded representation of the hypothesis $E$,
       variables in the hypothesis $v_1, v_2, ..., v_{n_v}$, and
       positions of each variable in the hypothesis
       $P_{v_1}, P_{v_2}, ...P_{v_{n_v}}$.
Output: New hypothesis representation $\mathcal{E}$.
$E_{(v_i-)} \leftarrow E$
for $j \leftarrow P_{v_0}$ to $P_{v_n}$ /* Discard variable
    representations */
 do
    $E_{(v_i-)_j} \leftarrow [0\ 0\ \dots\ 0\ 0]$
end
for $i \leftarrow 0$ to $n_v$ /* For all variables */
 do
    $e_{(v_i)} \sim \mathcal{N}(\mu, \sigma)$; /* Initialise new
        representation */
    for $t \leftarrow 0$ to $T$ /* Iterate over
        representation $T$ times */
     do
        $e_{(v_i)_{t-1}} \leftarrow e_{(v_i)}$
        $e_{(v_i)} \leftarrow \mathtt{LayerNorm}(e_{(v_i)})$
        $K \leftarrow k(E_{(v_i-)})$
        $Q \leftarrow q(e_{(v_i)})$
        $V \leftarrow v(E_{(v_i-)})$
        $A \leftarrow \mathtt{Att}(Q, K, V)$
        $e_{(v_i)} \leftarrow \mathtt{GRU}(e_{(v_i)_{t-1}}, A)$
        $e_{(v_i)} \leftarrow e_{(v_i)} + \mathtt{MLP}(\mathtt{LayerNorm}(e_{(v_i)}))$
    end
    for $j \leftarrow P_{v_{i0}}$ to $P_{v_{in}}$ /* Replace with
        new representation */
     do
        $E_{(v_i-)} \leftarrow e_{(v_i)}$
    end
end
$\mathcal{E} \leftarrow \mathtt{BiLSTM}(E_{(v_i-)})$
return $\mathcal{E}$
```