DISTILLING WIKIPEDIA MATHEMATICAL KNOWLEDGE INTO NEURAL NETWORK MODELS

Joanne T. Kim[*]
Lawrence Livermore National Laboratory
Livermore, CA 94550, USA
kim102@llnl.gov

Mikel Landajuela Larma[*]
Lawrence Livermore National Laboratory
Livermore, CA 94550, USA
landajuelala1@llnl.gov

Brenden K. Petersen[*†]
Lawrence Livermore National Laboratory
Livermore, CA 94550, USA
bp@llnl.gov

ABSTRACT

Machine learning applications to symbolic mathematics are becoming increasingly popular, yet there lacks a centralized source of real-world symbolic expressions to be used as training data. In contrast, the field of natural language processing leverages resources like Wikipedia that provide enormous amounts of real-world textual data. Adopting the philosophy of "mathematics as language," we bridge this gap by introducing a pipeline for distilling mathematical expressions embedded in Wikipedia into symbolic encodings to be used in downstream machine learning tasks. We demonstrate that a *mathematical language model* trained on this "corpus" of expressions can be used as a prior to improve the performance of neural-guided search for the task of symbolic regression.

1 INTRODUCTION

"The basis of all human culture is language, and mathematics is a special kind of linguistic activity."

— Arnold & Manin (2000)

A growing number of machine learning works leverage datasets of mathematical expressions to perform various symbolic reasoning tasks. These tasks include symbolic regression (Koza, 1992; Petersen et al., 2021), symbolic integration (Lample & Charton, 2019), solving differential equations, (Lample & Charton, 2019), and freeform mathematical question/answering (Saxton et al., 2019). The expression datasets used to complete these tasks are typically generated procedurally using various rules or heuristics (Lample & Charton, 2019; Saxton et al., 2019), crafted by hand (Uy et al., 2011), or manually extracted from existing texts (Udrescu & Tegmark, 2020). Yet, to the best of our knowledge, there exists no large-scale, customizable, and/or widely used dataset of mathematical expressions.

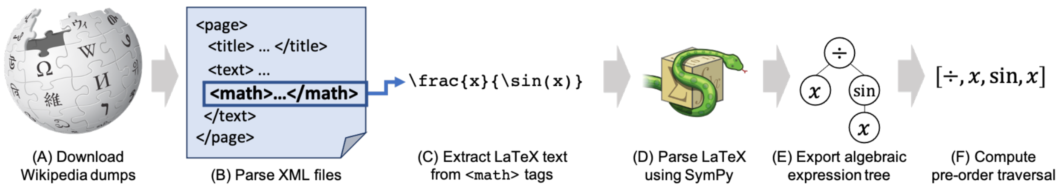

(A) Download Wikipedia dumps
<page>
<title> ... </title>
<text> ...
<math>...</math>
</text>
</page>
(B) Parse XML files
\frac{x}{\sin(x)}
(C) Extract LaTeX text from <math> tags
(D) Parse LaTeX using SymPy
÷
x
sin
x
(E) Export algebraic expression tree
[÷, x, sin, x]
(F) Compute pre-order traversal

Figure 1: Pipeline for generating the "corpus" of mathematical expressions.

[*]All authors contributed equally. [†]Corresponding author.

We introduce a simple pipeline for extracting mathematical expression encodings directly from raw Wikipedia text. Wikipedia is an excellent source for extracting mathematical knowledge for several reasons. First, Wikipedia is ever-growing and spans virtually all scientific domains. Second, expressions embedded in Wikipedia are conveniently annotated via XML tags and follow standard markup encodings (i.e. LaTeX), facilitating extraction and parsing. Third, Wikipedia is structured into hierarchical categories, allowing users to extract customized expression corpora, e.g. based on a particular branch of science. Finally, raw Wikipedia data dumps are frequently updated (without requiring web-crawling), easy to access, and freely available.

We demonstrate the practical use of our expression dataset by using it to train a *mathematical language model*, then using the trained model as a prior in a downstream neural-guided search task. Specifically, we consider symbolic regression, the task of searching the space of tractable mathematical expressions to fit a dataset. Symbolic regression is an excellent testbed problem for symbolic search because it poses a large combinatorial search space, is computationally inexpensive, and has well-established suites of benchmark problems (White et al., 2013). We demonstrate that leveraging a pre-trained mathematical language model as a prior to guide the search improves the ability to recover symbolic expressions from data, and provides other advantages such as reduced semantically invalid expressions.

2 Related Work

Datasets of expressions. Various expression datasets have been proposed for symbolic tasks. Lample & Charton (2019) procedurally generate expressions to perform symbolic mathematics, specifically to perform integration and solve ordinary differential equations. Expressions are randomly generated with hand-crafted rules, followed by a cleaning process to simplify and remove invalid expressions. Saxton et al. (2019) released a dataset for mathematical reasoning, which is also generated through rules that sample answer and generates matching questions. Udrescu & Tegmark (2020) introduce a dataset of 120 expressions to be used as benchmarks for symbolic regression, manually pulled from the Feynman lectures on physics (Feynman et al., 1965). To the best of our knowledge, there exists no large-scale dataset (or dataset-generating pipeline) of real-world symbolic expressions.

Mathematics as language. The perspective of viewing mathematics as a form of language dates as far back as the history of mathematics itself (Arnold & Manin, 2000). Within machine learning, Lample & Charton (2019) recently addressed symbolic mathematics as a machine translation problem, representing mathematical expressions as sequences and solving mathematical problems with a seq2seq model. We also consider mathematics as language to create a *mathematical language model* based on human-created, widely-used mathematical expressions in Wikipedia.

3 Methods

Extracting an expression corpus from Wikipedia. Our pipeline for extracting expressions from Wikipedia is illustrated in Fig. 1. Wikipedia provides its raw text data as XML (Extensible Markup Language) "dump" files.[1] Since it is written in a markup language, users can easily parse this dataset of rich information. Specifically, we extract embedded mathematical expressions, which are annotated via `<math>` tags (Fig. 1b-c). Raw expressions are represented by LaTeX text, which we convert into a tree-based representation—called an algebraic expression tree—via the computer algebra system SymPy (Meurer et al., 2017)[2]. Finally, expressions are encoded as sequences of tokens corresponding to the pre-order traversal of the algebraic expression tree. This data can then be readily used for downstream machine learning pipelines.

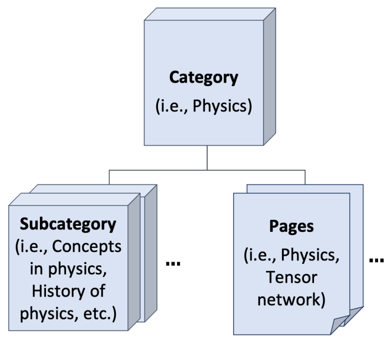

Category
(i.e., Physics)
Subcategory
(i.e., Concepts in physics, History of physics, etc.)
...
Pages
(i.e., Physics, Tensor network)
...

Figure 2: Wikipedia hierarchy structure.

[1]https://dumps.wikimedia.org
[2]https://sympy.org/

We also leverage Wikipedia's hierarchical page categorizations. As illustrated in Fig. 2, each category comprises subcategories and pages, forming a tree structure. Pages are the user interface to access content in Wikipedia, while categories are metadata used to point users to related content. To build a category tree, we need the edges connecting page and categories. Edge information is only partially available in the XML dump files. Therefore, we resort to the database provided by MediaWiki.[3] In particular, this database includes three relevant tables: Categorylinks, Category, and Page (Fig. 4 in the Appendix). Using database management system such as MySQL, we use this to generate a category tree from a given root category. Since categories may not be in strict hierarchy (i.e. they can be recursive), we specify a maximum search depth. In this manner, users can customize the expression dataset to specific categories (e.g. scientific domain) of interest.

Training a mathematical language model. By considering each symbolic token as a "word" from a dictionary of tokens, and each sequence of tokens (i.e. expression) as a "sentence," we propose building a mathematical language model (MLM) to estimate the probability for a mathematical expression represented by a sequence of tokens.

Before training the MLM, we first filter and augment the dataset based on the operators and operands used in the downstream learning task. For instance, if the task does not use integrals, each expression containing an integral symbol could be simply removed. Instead, we choose two approaches to augment this data: replacing and splitting. When an integral appears while traversing the expressional tree, the whole subtree with a root of integral can be replaced with a terminal token. Another approach is splitting the tree to use the expression in the integrand for data augmentation.

Finally, we train a recurrent neural network (RNN) with our preprocessed and augmented expression dataset to create a MLM using the set of given symbols as the vocabulary. Specifically, we train the MLM to predict the next token in a sequence by minimizing the cross-entropy loss between the output of the RNN and the given label. This strategy is standard in natural language processing (NLP) for training language models that predict the next word or character given a partial sequence (Mikolov et al., 2010).

Integrating with deep symbolic regression. Deep symbolic regression (Petersen et al., 2021) uses an autoregressive recurrent neural network, or *Controller*, parameterized by θ, that emits a categorical distribution on which token to select next, conditioned on all previously selected tokens.

Here, we propose using the MLM to inform the Controller by directly adjusting the logits that are emitted by it, as in Fig. 3. Specifically, at time step i, given input x (e.g. the previously selected token), the Controller emits logits $\ell^{(i)}_{\text{DSR}}$ and updates its internal state $c^{(i)}_{\text{DSR}}$:

$$(\ell^{(i)}_{\text{DSR}}, c^{(i)}_{\text{DSR}}) = \text{Controller}(x, c^{(i-1)}_{\text{DSR}}; \theta)$$

Further, DSR allows incorporating in situ constraints to the search space (e.g. constraining nested trigonometric operators) via logits $\ell^{(i)}_{\oslash}$, whose values are either zero or negative infinity (see Petersen et al. (2021) for details). To incorporate the MLM into this generative process, we introduce a third vector of logits computed by the MLM:

$$(\ell^{(i)}_{\text{MLM}}, c^{(i)}_{\text{MLM}}) = \text{MLM}(x, c^{(i-1)}_{\text{MLM}}; \phi)$$

Finally, the logit vectors are summed, a softmax operator is applied, and the resulting probability vector defines a categorical distribution used

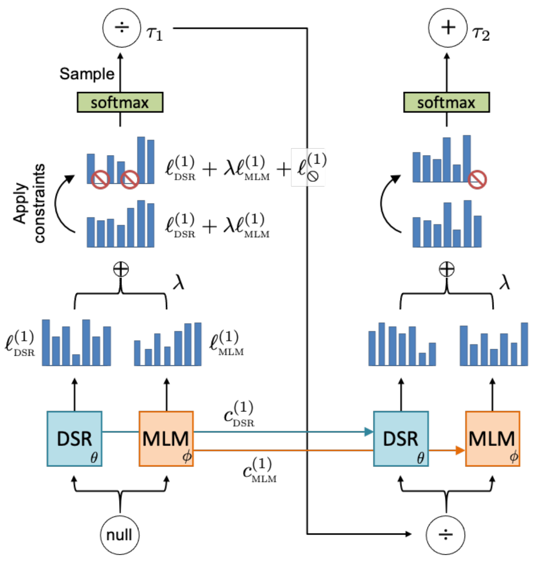

÷
τ_1
Sample
softmax
$\ell^{(1)}_{\text{DSR}} + \lambda\ell^{(1)}_{\text{MLM}} + \ell^{(1)}_{\oslash}$
Apply constraints
$\ell^{(1)}_{\text{DSR}} + \lambda\ell^{(1)}_{\text{MLM}}$
$\oplus$
λ
$\ell^{(1)}_{\text{DSR}}$
$\ell^{(1)}_{\text{MLM}}$
DSR θ
MLM ϕ
$c^{(1)}_{\text{DSR}}$
$c^{(1)}_{\text{MLM}}$
null
+
τ_2
softmax
$\oplus$
λ
DSR θ
MLM ϕ
÷

Figure 3: Integration between DSR and the MLM prior. For each token, each architecture (DSR, MLM) outputs a vector of logits. Logits are combined, constraints are applied, and a final softmax determines the categorical distribution from which a token is sampled.

[3]https://mediawiki.org

Table 1: Comparison of recovery rate, mean steps to solve, and mean rate of invalid expressions in the symbolic regression task with and without the MLM prior.

	DSR without MLM			DSR with MLM		
Benchmark	Recovery	Steps	Invalid	Recovery	Steps	Invalid
$x^3 + x^2 + x$	100.%	165.2	47.82	100.%	99.47	56.75
$x^4 + x^3 + x^2 + x$	100.%	264.6	35.86	100.%	235.6	36.46
$x^5 + x^4 + x^3 + x^2 + x$	100.%	349.2	28.35	100.%	329.1	26.93
$x^6 + x^5 + x^4 + x^3 + x^2 + x$	100.%	672.2	17.65	100.%	525.5	20.75
$\sin(x^2)\cos(x) - 1$	76.%	847.8	23.67	94.%	672.8	22.81
$\sin(x) + \sin(x + x^2)$	100.%	189.3	45.26	100.%	120.3	51.63
$\log(x + 1) + \log(x^2 + 1)$	35.%	1513.	11.02	27.%	1620.	10.91
$\sqrt{x}$	95.%	601.	31.81	99.%	365.1	32.62
$\sin(x) + \sin(y^2)$	100.%	117.8	34.35	100.%	95.52	27.34
$2\sin(x)\cos(y)$	100.%	368.3	17.55	100.%	364.3	13.93
x^y	100.%	22.36	56.37	100.%	12.58	41.13
$x^4 - x^3 + \frac{1}{2}y^2 - y$	0.%	2000.	6.373	0.%	2000.	5.71
Average:	83.8%	592.6	29.7%	**85.0%**	**536.7**	**28.9%**

to sample the next token τ_i:

$$\tau_i \sim \text{Categorical}(\text{Softmax}(\ell^{(i)}_{\text{DSR}} + \lambda \ell^{(i)}_{\text{MLM}} + \ell^{(i)}_{\oslash})).$$

The hyperparameter $\lambda \in \mathbb{R}^+$ controls the strength of the MLM prior. We provide an interpretation of λ based on the temperature of the softmax function. Note the softmax function with temperature T and logits ℓ is defined as: $\text{softmax}_T(\ell) \triangleq \text{softmax}(\ell/T)$. Multiplying logits by λ yields:

$$\lambda\ell = \text{inverse-softmax}(\text{softmax}(\lambda\ell)) = \text{inverse-softmax}(\text{softmax}_{\lambda^{-1}}(\ell)),$$

where inverse-softmax is defined up to an arbitrary constant. Thus, λ can be viewed as the *inverse temperature* of the MLM prior's contribution to the final softmax. Higher temperatures (lower λ) result in a lower influence of the MLM prior on the final softmax. At the extreme, $\lambda \to 0$ corresponds to infinite temperature, in which case the MLM prior is ignored.

Pseudocode for DSR integrated with the MLM prior is provided in the Appendix. Note that the recurrent architecture used in the MLM can be completely different than the one used in DSR; this allows the MLM to be pre-trained offline.

4 Results and Discussion

Dataset statistics. From a raw Wikipedia dump file of ∼70 GB, we extracted 798,998 mathematical expressions across 41,763 different pages. As an example of using the category hierarchies, when we search pages under the category *Physics* with depth 3, we collect 67,404 expressions from 2,265 pages in 879 categories, while *Physics* itself has 25 pages with 1,374 expressions.

Application to symbolic regression. We demonstrate the value of the MLM by using it to inform the task of symbolic regression. For simplicity, we replicate the experimental setup and hyperparameters detailed in Petersen et al. (2021), leveraging the accompanying open-source package "Deep Symbolic Regression." The only change is the introduction of the MLM prior as previously described, sweeping over inverse temperature hyperparameter $\lambda \in \{0.1, 0.2, \ldots, 1.0\}$. The MLM is trained for 200 epochs using a recurrent neural network with a hidden layer size of 256, showing the cross-entropy loss going down to 0.367.

In Table 1, we report the recovery rate (fraction of runs in which the exact symbolic expression is found), average number of steps required to find the solution, and the fraction of invalid expressions (those that produce floating-point errors, e.g. overflows) produced during training, over 100 independent runs. The MLM prior show an improvement in recovery rate, requiring fewer steps and generating fewer invalid expressions. In particular, we note the dramatic reduction in steps required to find expressions $\sqrt{x}$, $\sin(x) + \sin(y^2)$, and x^y. Overall, these results show that the use of the MLM provides a better informed search in symbolic regression tasks.

5 Conclusion and Future Direction

We introduce a pipeline for generating a largescale "corpus" of mathematical expressions directly from Wikipedia, and demonstrate that a language model trained on this dataset can improve neural-guided search for the task of symbolic regression. Possible alternative uses of a MLM include auto-completion of writing expressions in LaTeX, or improving recognition of hand-written expressions.

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC. LLNL-CONF-820039.

References

V Arnold and Yu Manin. Mathematics as profession and vocation. In *Mathematics: Frontiers and Perspectives*, pp. 153–159. American Mathematical Society, 2000.

Richard P Feynman, Robert B Leighton, Matthew Sands, and EM Hafner. The feynman lectures on physics; vol. i. *American Journal of Physics*, 33(9):750–752, 1965.

John R Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, volume 1. MIT press, 1992.

Guillaume Lample and François Charton. Deep learning for symbolic mathematics. *arXiv preprint arXiv:1912.01412*, 2019.

Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, et al. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

Brenden K Petersen, Mikel Landajuela, T Nathan Mundhenk, Claudio P Santiago, Soo K Kim, and Joanne T Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *Proc. of the International Conference on Learning Representations*, 2021.

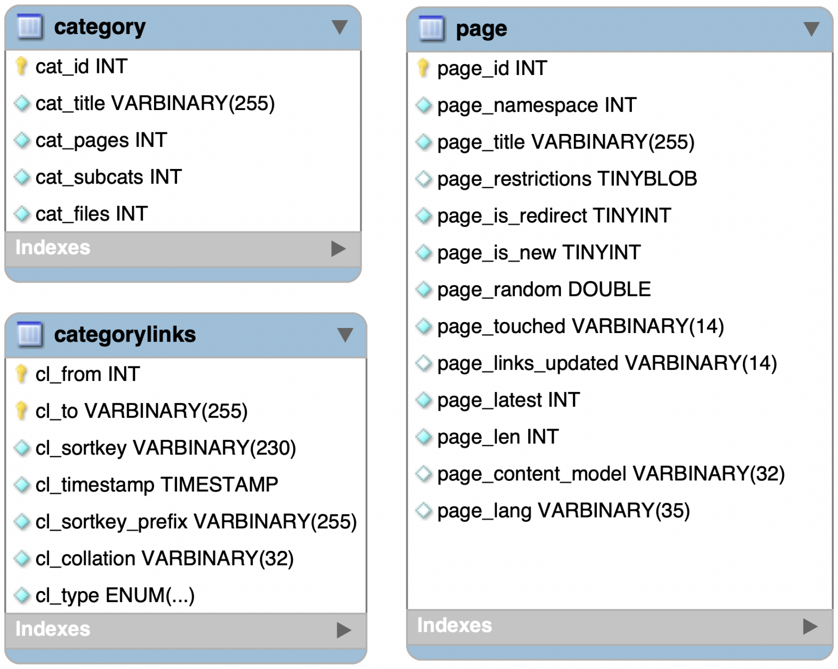
David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557*, 2019.

Silviu-Marian Udrescu and Max Tegmark. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.

Nguyen Quang Uy, Nguyen Xuan Hoai, Michael O'Neill, Robert I McKay, and Edgar Galván-López. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 12(2):91–119, 2011.

David R White, James Mcdermott, Mauro Castelli, Luca Manzoni, Brian W Goldman, Gabriel Kronberger, Wojciech Jaśkowski, Una-May O'Reilly, and Sean Luke. Better gp benchmarks: community survey results and proposals. *Genetic Programming and Evolvable Machines*, 14(1): 3–29, 2013.

A Appendix



category
cat_id INT
cat_title VARBINARY(255)
cat_pages INT
cat_subcats INT
cat_files INT
Indexes

categorylinks
cl_from INT
cl_to VARBINARY(255)
cl_sortkey VARBINARY(230)
cl_timestamp TIMESTAMP
cl_sortkey_prefix VARBINARY(255)
cl_collation VARBINARY(32)
cl_type ENUM(...)
Indexes

page
page_id INT
page_namespace INT
page_title VARBINARY(255)
page_restrictions TINYBLOB
page_is_redirect TINYINT
page_is_new TINYINT
page_random DOUBLE
page_touched VARBINARY(14)
page_links_updated VARBINARY(14)
page_latest INT
page_len INT
page_content_model VARBINARY(32)
page_lang VARBINARY(35)
Indexes

Figure 4: Schema of Wikipedia database tables used to extract hierarchy tree of categories and pages. The Categorylinks table holds the subcategory information by setting `cl_type` as *subcat*. Note that `categorylinks.cl_from` stores the id of the article which matches `page.page_id`, a primary key, while `categorylinks.cl_to` stores the name of the category as text.

Algorithm 1 Generating an expression from DSR under the MLM

```
input  DSR controller with parameters $\theta$; MLM with parameters $\phi$; library of tokens $\mathcal{L}$
output  Pre-order traversal $\tau$ of an expression sampled from the RNN
1: $\tau \leftarrow []$                                                  ▷ Initialize empty traversal
2: $x \leftarrow$ empty||empty                                           ▷ Initial RNN input is empty parent and sibling
3: $c_0 \leftarrow \vec{0}$                                              ▷ Initialize RNN cell state to zero
4: for $i = 1, 2, \ldots$ do
5:     $(\ell^{(i)}_{\text{DSR}}, c^{(i)}_{\text{DSR}}) \leftarrow \text{Controller}(x, c^{(i-1)}_{\text{DSR}}; \theta)$      ▷ Emit DSR logits; update DSR state
6:     $(\ell^{(i)}_{\text{MLM}}, c^{(i)}_{\text{MLM}}) \leftarrow \text{MLM}(x, c^{(i-1)}_{\text{MLM}}; \phi)$      ▷ Emit MLM logits; update MLM state
7:     $\ell^{(i)}_{\oslash} \leftarrow \text{Constraints}(\mathcal{L}, \tau)$      ▷ Compute constraints
8:     $p^{(i)} \leftarrow \text{Softmax}(\ell^{(i)}_{\text{DSR}} + \lambda \ell^{(i)}_{\text{MLM}} + \ell^{(i)}_{\oslash})$      ▷ Compute probability vector
9:     $\tau_i \leftarrow \text{Categorical}(p^{(i)})$      ▷ Sample next token
10:    $\tau \leftarrow \tau \| \tau_i$      ▷ Append token to traversal
11:    if expression complete then return $\tau$      ▷ If expression is complete, return it
12:    $x \leftarrow \text{ParentSibling}(\tau)$      ▷ Compute next parent and sibling
13: end for
```