Language Models, Mathematics, Embeddings

Zsolt Zombori*[1,3], Pál Zsámboki*[1,3], Ádám Fraknói[3], Máté Gedeon[2], and András Kornai[2]

[1] Alfréd Rényi Institute of Mathematics, Budapest
[2] Dept. of Algebra, Budapest University of Technology and Economics
[3] Eötvös Loránd University, Budapest, Hungary

Problem statement We are interested in improving the performance of transformer language models (LMs) in solving mathematical tasks. Today, even large language models (LLMs) with billions of parameters are notoriously bad for performing simple arithmetic tasks [2], a situation we attribute to a fundamental difference between the role individual tokens play in natural language processing (NLP) and in arithmetic. NLP relies on short sequences of tokens (word pieces) to recur in morphemes and words (meaningful units) predictably, and with a strong conditioning between such units and the units that make up their immediate context. For example, in disambiguation tasks (such as between bank_1 'side of the river' and bank_2 'financial institution'), a context window of three words on either side is generally sufficient, and LLMs now match human performance on the task [4, 1]. In contrast, in arithmetic expressions the tokens are unpredictable based on local context, and proper parenthesization, a task infamously hard for humans [5] (consider 'the milk the rat the cat the dog chased caught sought curled') is very relevant.

Approach taken An LM can be seen as a sequence of transformations, each of which produces a vector representation of the input data. The focus of our project is understanding what kind of representations emerge during learning and how they interact with the model's ability to solve various downstream tasks.

We consider learned representations both as a window that allows us to analyse the inner workings of the LM as well as a funnel through which one can inject background knowledge. Proper data represenation plays a crucial role in LM performance and in mathematical theories we often have some a priori expectations about what a good representation should look like. For example, given a set of arithmetic expressions, we would expect those that evaluate to a similar value to be represented close to each other. We can leverage such background knowledge by adding an explicit auxiliary learning task that encourages a certain structure on the representation. Such auxiliary tasks are frequent in machine learning and are referred to as *probing* [1] in NLP.

We created a flexible dataset generator and a decoder-only transformer implementation that allows for co-training on several different tasks and datasets, with facilities for inspecting and visualizing the models both during and after training.

The tasks that we consider come from two categories. First, we implemented classical language modelling tasks such as masked language modelling (predict the tokens missing from the input) and generative modelling (given a partial input string, predict the next token), which we refer to as the *language task*. Second, we provide auxiliary tasks related to the structure of the representation, referred to as the *embedding task*.

*First two authors contributed equally.

Embedding Task A transformer produces context dependent vector representations at the output of each transformer block for each input token. It also provides a special token (the `[CLS]` at the beginning of each input string) whose representation can be treated as the embedding of the entire input sequence. Furthermore, representations internal to a transformer block also emerge that are factorised into three components (key, query and value vectors). At this phase of our work, we focus on the sentence embedding associated with the `[CLS]` token. We have the choice to focus on any particular block of the model and optimise its sequence embedding.

In order to actually induce the learning procedure to store sequence specific useful information in this vector, a suitable learning task has to be selected. It is highly nontrivial to decide what kind of embedding structure should be enforced for a given a mathematical theory, and it requires experimentation. Here, we just give some illustrative intuitions:

- Two logical formulae that are equisatisfiable (e.g. $\neg(p \wedge q)$ and $\neg p \vee \neg q$) should be mapped close to each other.
- The greater the distance between the values of two arithmetic expressions, the greater the distance in the embedding space (e.g. $23 + 12$ should be closer to 40 than to 1000).

Current Status We are experimenting with arithmetic problems of varying complexity:

- **Solve formula**: Compute the value of an arithmetic formula, e.g. $(4 - 3) + 23 \rightarrow 24$
- **Derive formula**: Compute the value of an arithmetic formula with intermediary results, e.g. $(4 - 3) + 23 \rightarrow 1 + 23 \rightarrow 24$
- **Solve equations**: Compute the solution of a set of linear equations, e.g. $2X + Y = 7, X + 2Y = 8 \rightarrow X = 2, Y = 3$
- **Derive equations**: Compute the solution of a set of linear equations with intermediary results of gaussian elimination, e.g. $2X + Y = 7, X + 2Y = 8 \rightarrow \ldots \rightarrow X = 2, Y = 3$

Our embedding task follows [3]: we take sets of expressions, two of which have equal value and we train the model to embed the expressions with equal value closer to each other than other expressions.

We have conducted initial probing experiments and the results are not yet conclusive. We hope to see patterns in the embeddings emerging in models trained to very high precision, as well as improvement on the language tasks for some datasets when trained with an auxiliary embedding task. We expect to report results by the time of the meeting.

Conclusion We built training and analysis infrastructure for a language model to be trained on mathematical problems provided in textual format. The system can be trained with auxiliary learning objectives related to the structure of latent representations of mathematical expressions. We hope to uncover the interplay between the model's learned representations and its performance on the final task.

References

[1] Judit Acs, Endre Hamerlik, Roy Schwartz, Noah A. Smith, and Andras Kornai. Morphosyntactic probing of multilingual bert models. *Natural Language Engineering*, page 1–40, 2023.

[2] Shima Imani, Liang Du, and Harsh Shrivastava. MathPrompter: Mathematical reasoning using large language models. In Sunayana Sitaram, Beata Beigman Klebanov, and Jason D Williams, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*

(Volume 5: Industry Track), pages 37–42, Toronto, Canada, July 2023. Association for Computational Linguistics.

[3] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online, November 2020. Association for Computational Linguistics.

[4] Vatsal Sharan, Sham Kakade, Percy Liang, and Gregory Valiant. Prediction with a short memory. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, page 1074–1087, New York, NY, USA, 2018. Association for Computing Machinery.

[5] Victor H. Yngve. The depth hypothesis. In R. Jakobson, editor, *Structure of Language and its Mathematical Aspects*, pages 130–138. American Mathematical Society, Providence, RI, 1961.