PRE-TOKENIZATION OF NUMBERS FOR LARGE LANGUAGE MODELS

Zhenglong Wu, Qi Qi[*], Zirui Zhuang, Haifeng Sun, Jingyu Wang
School of Computer Science
Beijing University of Posts and Telecommunications
Beijing, 100876, China
{wuzhenglong,qiqi8266,zhuangzirui,hfsun,wangjingyu}@bupt.edu.cn

ABSTRACT

There have been many recent advances in LLM for zero-shot tasks. While these models have shown great promise, pre-tokenization process methods for numbers are mostly empirical and lack experimental justification. In this paper, we analyze the tokenization methods of numbers through time series forecasting. We conducted experiments to evaluate the impact of different factors on Byte-Pair Encoding (BPE) tokenizers and proposed a novel pre-tokenization algorithm that is justified to maintain the balance between details and memory cost. Our analysis highlights the importance of a systematic understanding of the pre-tokenization process and provides a baseline for further exploration.[1]

1 INTRODUCTION & BACKGROUND

Large Language Models (LLMs) have proven their generalization capabilities across diverse tasks, demonstrating exceptional performance in zero-shot settings (Touvron et al., 2023a;b; Brown et al., 2020; Workshop et al., 2022). With extensive training and learning from human feedback, these models have learned to process and generate language with unprecedented accuracy even beyond English (Ouyang et al., 2022; Du et al., 2022; Zeng et al., 2022). The success of LLM has inspired exploration of their potential in adjacent areas, such as time series analysis. Remarkably, these models continue to excel in zero-shot tasks when processing tokenized input (Gruver et al., 2023), highlighting their remarkable flexibility and adaptability.

While directly feeding LLM with the text form of numbers has been shown to be effective (Gruver et al., 2023), existing research largely conducts the pre-tokenization process on an empirical basis. To the best of our knowledge, systematical pre-tokenization approach

By revisiting the pre-tokenization process of textual sequences, we propose several pre-tokenization strategies for numbers. For both open and closed LLMs, the current state-of-the-art (SOTA) models are based on the Byte-Pair Encoding (BPE) tokenization algorithm (Sennrich et al., 2015).

2 METHODOLOGY & EXPERIMENTS

In this section, we propose process of pre-tokenization and validate with experiments as in Figure 1. We hire the Informer dataset (Zhou et al., 2021), which is a collection of 6 time series forecasting datasets.

2.1 OBSERVATIONS

Rounding: The Precision and Redundancy Trade-off. In NLP tasks, preserving more tokens enables LLMs to access more context, but overly detailed context also raised the Out-Of-Vocabulary issue (Sennrich et al., 2015). A double-precision FP64 floating-point number occupies 64 bits (8

[*]Corresponding author.

[1]Implementation of our method can be found at: *github.com/itdevwu/Pre-Tokenization-of-Numbers*

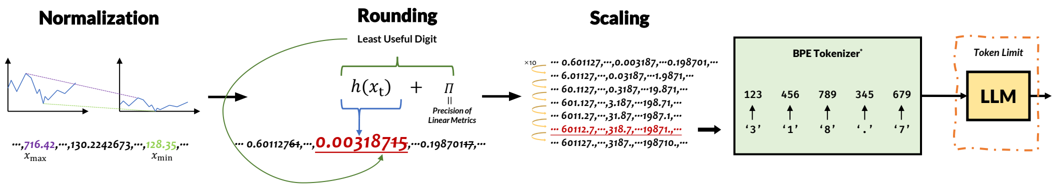

Figure 1: Algorithm flow. **Normalization**: Scale the entire sequence to a similar range; **Rounding**: rounding off some bits without affecting precision; **Scaling**: Scaling the sequence to make it the shortest possible textual form. Some LLMs tokenize digits separately (Touvron et al., 2023a;b)

Bytes), yet with modified BPE tokenizer from models like LLaMA (Touvron et al., 2023a;b) or ChatGLM (Du et al., 2022; Zeng et al., 2022), FP64 is represented as a maximum 18-token sequence and stored as a $[1 \times 18]$ tensor, costing $18 \times w_{bit}$ bits, where w_{bits} denotes bit width (usually 16 bits, could be 8 bits or 4 bits after quantization). Longer textual representations leads to more memory consumption and shorter context window, and trimming trailing digits appropriately during pre-processing can save memory and denoise the series

Normalization & Scaling: Controllable Model Performance Defeating Repeat. Normalization scales the series into a range, which standardized the input for LLM. While normalization is commonly used to ensure consistency and comparability, another key usage of normalization in LLM zero-shot subject is, to ensure the un-tuned LLM behave as expected. Scaling is a counter-intuitive that prevents repeat, which undermine zero-shot numerical ability of LLM. We discovered that keeping the input sequence in a generally large and diverse range can reduce repeat. This is possibly because of larger value range increase appearance of a larger group of token. As Figure 2 has illustrated, scaling series into a larger range improves zero-shot performance LLM.

2.2 Our Proposal

Inspired by the above observations, we propose a pre-tokenization algorithm described in Figure 1 that reduces overall token length while maintaining required precision. Let Π as the precision required for linear metrics as mean absolute error (MAE). Our approach starts by normalizing the series $X^T = \{x_1, x_2, \cdots, x_t, \cdots x_T\}$ using min-max normalization into the range $[0, 1]$. Next, we identify the smallest number $x_{min} \in X^T$. Denote the highest non-zero digit of x as $h(x) = \lceil(-\lg x)\rceil$, and then calculate $h(x_{min}) + \Pi$ as the lowest useful digit for x_{min}, as digits below this threshold are smaller than $\frac{1}{10^{\Pi}}$. As long as this precision ensures that x_{min} satisfies the precision requirement, the entire sequence will not lose precision due to rounding. After rounding, we multiply each element of X^T by 10 repeatedly until all numbers are integers. Throughout this process, we record the smallest token length for the entire X^T which saves the most memory.

The maximum number of digits for a double-precision floating-point number is 18, which leads to an upper bound on the time complexity of the scaling stage of $O(18T)$. Since both the input normalization and finding x_{min} have a time complexity of $\Theta(T)$, the upper bound of our proposed algorithm should be $O(20T)$, resulting in a linear time complexity of $\Theta(T)$. We tested our method, and it performed similarly while reducing token lengths.

3 Conclusion

In this paper, we reviewed factors affecting performance of zero-shot numerical tasks with LLM. By summarizing observations, we proposed a linear algorithm that saves memory without losing precision. We hope that to invoke more research that unleash potential of LLM in its era.

URM Statement

The authors acknowledge that at least one key author of this work meets the URM criteria of ICLR 2024 Tiny Papers Track.

REFERENCES

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. Glm: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 320–335, 2022.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters. *arXiv preprint arXiv:2310.07820*, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

BigScience Workshop, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*, 2022.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

A APPENDIX

A.1 EXPERIMENT RESULTS

For the pre-training model, we chose the smaller LLaMA2-7B, as our aim was to investigate suitable ways of pre-tokenization, rather than to specifically improve the performance of time series forecasting.

Our experiments were completed using a Nvidia GeForce RTX 3090 24 GB with CUDA 12.1. Due to the large consumption of computational resources by the large language model, we quantized the model with GPTQ (Frantar et al., 2022). Since this experiment does not involve comparison of models, quantization should not affect the conclusions of the experiment.

We conducted experiments on the following dataset. The table shows the performance of our method and other different pre-tokenization methods on different data. Due to the high computational cost of large language models, we sampled the following experiments on all datasets and ensured that every sequence in each dataset was in the sample.

When pre-processing the dataset, we normalized all the data by deflating the data within the interval of $[0, 1]$ in order to compare the effect of different deflations on the results.

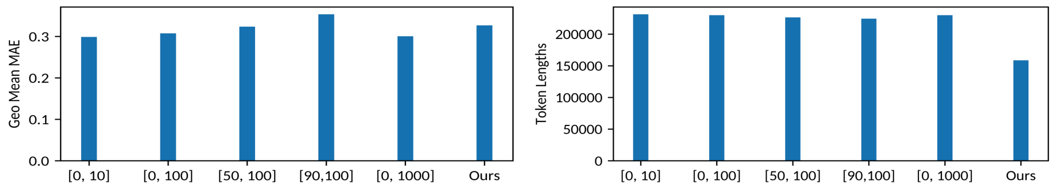

Figure 2: Experiemnts. The left part shows that despite additional rounding applied, our method has similar performance to empirical ones. The right part shows that our method costs less memory.

	Scaling Range					
Datasets	$[0, 10]$	$[0, 100]$	$[50, 100]$	$[90, 100]$	$[0, 1000]$	**Ours**
exchange rate	0.343	0.349	0.319	0.374	0.305	0.365
traffic	0.223	0.200	0.245	0.282	0.233	0.313
electricity	0.266	0.302	0.321	0.379	0.307	0.285
ETTh1	0.332	0.392	0.352	0.386	0.328	0.346
ETTh2	0.351	0.332	0.345	0.346	0.329	0.359
ETTm1	0.325	0.347	0.332	0.427	0.324	0.324
ETTm2	0.282	0.288	0.302	0.310	0.300	0.339
illness	0.325	0.273	0.372	0.373	0.334	0.306
weather	0.269	0.326	0.341	0.325	0.260	0.311

Table 1: Experiments Result. Result are aggregated by geographic mean due to the series has been normalized prior to forecasting.