The Ups and Downs of Large Language Model Inference with Vocabulary Trimming by Language Heuristics

Nikolay Bogoychev Pinzhen Chen Barry Haddow Alexandra Birch
School of Informatics, University of Edinburgh
{nbogoych,pinzhen.chen,bhaddow,a.birch}@ed.ac.uk

Abstract

Deploying large language models (LLMs) encounters challenges due to intensive computational and memory requirements. Our research examines vocabulary trimming (VT) inspired by restricting embedding entries to the language of interest to bolster time and memory efficiency. While such modifications have been proven effective in tasks like machine translation, tailoring them to LLMs demands specific modifications given the diverse nature of LLM applications. We apply two language heuristics to trim the full vocabulary—Unicode-based script filtering and corpus-based selection—to different LLM families and sizes. The methods are straightforward, interpretable, and easy to implement. It is found that VT reduces the memory usage of small models by nearly 50% and has an upper bound of 25% improvement in generation speed. Yet, we reveal the limitations of these methods in that they do not perform consistently well for each language with diminishing returns in larger models.

1 Introduction

Large language models (LLMs) are gaining increasing attention given their strong performance (Radford et al., 2019; Brown et al., 2020; Scao et al., 2022; Touvron et al., 2023). LLMs, especially multilingual ones, hold vocabulary items for many languages and scripts, which entail a costly matrix multiplication $H \times |V|$ in the output layer, where H is the hidden size and $|V|$ is the size of a vocabulary V. This expensive operation leads to increased costs of both memory and time given the autoregressive nature of LLM decoding. Given their substantial size, this latency in inference significantly escalates the expense of LLM deployment.

In practice, creating a sub-vocabulary V' with $|V'| \ll |V|$ and only loading its corresponding embedding entries for inference seems favourable since most logits from the output layer do not affect the hypothesis token(s) at each time step. Vocabulary trimming (VT) has been actively explored in machine translation (often called *shortlisting*, Schwenk et al., 2007; Le et al., 2012; Devlin et al., 2014)—it computes token-level alignments and makes potential target tokens a subvocabulary. While anticipating certain limitations such as domain mismatch (Bogoychev and Chen, 2021; Domhan et al., 2022), vocabulary shortlisting in LLMs poses a fundamental challenge: often LLM outputs are variable and open-ended, complicating the determination of the required lexicons. Recent attempts at multilingual pre-trained models select tokens in a task's language (Abdaoui et al., 2020; Ushio et al., 2023). Nonetheless, research in this direction is still limited, especially in speed considerations.

We follow the idea of fitting vocabulary to the language of the downstream task. Specifically, We examine two strategies: *Unicode-based filtering* where vocabulary items are removed if they do not belong to the task language, and *corpus-based selection* where we record vocabulary hits from a large representative corpus. After experimenting with LLMs from two families of different sizes, we identify a good upper bound of memory reduction with several limitations and outlooks: 1) Unicode-based script filtering maintains quality for Latin-based languages but harms languages requiring code-mixing. 2) Corpus-based selection leads to fewer alterations but is less effective in reducing the embedding size. 3) Embeddings are proportionally smaller in larger models (with smaller vocabularies). Yet we argue that VT can be applied orthogonally to other efficiency methods like efficient attention, quantization, etc.

2 Language-Based Vocabulary Trimming

We explore two ways to prepare sub-vocabulary for LLMs, focusing on only retaining tokens relevant to the language being generated. We test a batched setting on the fly: we determine a sub-

Proceedings of the Fifth Workshop on Insights from Negative Results in NLP, pages 148–153

vocabulary for an entire batch because creating the sub-vocabulary separately for each input is too expensive in practice. Furthermore, we always include all tokens appearing in the inputs.

Script-based filtering This is done by filtering token strings that fall out of a language's Unicode range—keeping tokens in the writing script of that language. It should be especially effective for languages operating on unique scripts, such as Armenian, Chinese, Korean, etc since it allows for concise vocabulary restriction. This method might be less practical if a writing system is shared among multiple languages (e.g. Cyrillic or Latin alphabets), because it would be infeasible to limit the lexicons to those solely belong to a specific language, resulting in a relatively large sub-vocabulary. Moreover, this method would strictly rule out code-mixed texts, emojis, etc, which are used in real-world communications.

Corpus-based selection Another way is to tokenize a representative corpus in the desired language in advance and use the vocabulary entries that have been recorded to build a sub-vocabulary. This method is non-exhaustive because we could miss rare but valid tokens or suffer from domain mismatch between the vocabulary selection corpus and the downstream tasks at inference time.

3 Experimental Setup

Languages and test sets We experimented on four languages: Bulgarian, Chinese, English, and Spanish, to offer distinct conditions that cover different degrees of writing script overlap, code-mixing, etc, with details in Appendix A. We sample 50 prompt questions from OpenAssistant (Köpf et al., 2023) which are then human-translated into test languages. We decode them with beam size 1.

Metrics We consider efficiency-quality trade-offs. In terms of speed, we report end-to-end time to decode the entire test, including model loading and embedding slicing. As a quality indicator, we count the chances a model fails to produce *the exact same output* (miss) with a full vocabulary and with VT. In addition, we report the BLEU and chrF of the VT output w.r.t. to the *original output* with the full vocabulary (not the reference). Note that there is no gold reference due to the open domain nature; we hence prefix the two string metrics with an "o-".

Large language models We experiment with instruction-tuned LLMs based on BLOOM at various sizes (Scao et al., 2022) as well as LLaMA-7B (Touvron et al., 2023). We adopt Chen et al. (2024)'s models fine-tuned on machine translations of the Alpaca dataset (Taori et al., 2023) to test for open domain question answering, which maximizes the difficulty for VT as explained earlier.

BLOOM is multilingual and explicitly supports English, Spanish, and Chinese, but not Bulgarian. Consequently, it has a sizeable vocabulary of 250K and is therefore a prime and tempting candidate to reduce vocabulary for a specific language during inference. We experiment with the 560M, 1B7, and 7B1 checkpoints, with diminishing computational burden on the embedding and output layers.

LLaMA is an English-centric LLM with a small 32K vocabulary. We might have reduced benefit from VT because a proportionally lower amount of computation occurs in the output layer. On the other hand, since the LLM is European language-focused, we expect drastic vocabulary reductions compared to BLOOM for Bulgarian and Chinese.

Vocabulary trimming details We tokenize the test prompts and always include input tokens in the sub-vocabulary. We then apply either of the proposed selection methods. Script-based filtering checks whether a vocabulary entry belongs to a Unicode subset: Cyrillic for Bulgarian, ASCII for English, Latin Extended-A for Spanish, and Chinese characters for Chinese. Whereas for corpus-based selection, we tokenize a subset of WikiMatrix (Schwenk et al., 2021) containing Wikipedia texts for each language and record vocabulary hits.

For both selection methods, we keep the first 300 vocabulary entries of each LLM too, as those usually correspond to special tokens, Unicode bytes (for byte-level BPE), numbers, etc. We compute the sub-vocabulary offline and we do not record the time spent on pre-tokenizing a large corpus or extracting a Unicode subset in the measurements, as once done, these can be reused for every batch during inference. Script-based filtering takes under 60 seconds and corpus-based selection takes up to 10 minutes. Adding the inputs' tokens to the sub-vocabulary takes negligible time.

Hardware We conduct experiments both on CPU and GPU devices. For the CPU tests, we use Xeon Gold 6248 (40 Cores, 80 Threads), and for GPU tests, we use a single Nvidia RTX 3090. CPU in-

Language	$\|V\|$	BLOOM-560M				BLOOM-1B7				BLOOM-7B1			
		time	miss	o-BLEU	o-chrF	time	miss	o-BLEU	o-chrF	time	miss	o-BLEU	o-chrF
bg full	250680	05:26	–			15:18	–			65:01	–		
Unicode	22912	04:39	1	99.04	99.73	13:44	4	91.67	96.36	51:46	10	83.42	87.03
corpus	58642	04:49	0			09:34	1	99.49	99.85	60:28	3	91.68	95.84
oracle	1408	04:22	0			12:31	0			61:06	0		
en full	250680	07:37	–			16:35	–			55:08	–		
Unicode	186752	07:40	1	98.21	98.68	16:05	0			58:18	0		
corpus	113024	07:00	1	99.22	99.45	15:08	3	96.59	98.88	54:20	2	98.91	99.40
oracle	4736	06:14	0			13:06	0			48:46	0		
es full	250680	05:58	–			12:26	–			63.15	–		
Unicode	187008	05:48	0			12:01	0			59:15	0		
corpus	112128	05:37	0			11:34	4	95.91	97.71	57:41	4	94.46	96.25
oracle	4736	04:53	0			09:26	0			51:43	0		
zh full	250680	06:29	–			15:27	–			55:09	–		
Unicode	51584	05:54	16	53.32	70.38	13:09	21	47.66	63.66	50:50	22	47.76	63.60
corpus	104320	06:08	11	66.17	78.44	14:08	16	63.26	73.99	46:39	17	62.37	76.55
oracle	4096	05:16	0			12:07	0			50:50	0		

Table 1: CPU VT results for the BLOOM family.

Language	$\|V\|$	LLaMA-7B			
		time	miss	o-BLEU	o-chrF
bg full	32000	117:15	–		
Unicode	4736	125:55	19	74.13	81.51
corpus	26496	132:24	5	97.75	98.44
oracle	2048	123:06	0		
en full	32000	113:52	–		
Unicode	27520	125:57	6	79.43	88.91
corpus	30720	111:30	19	93.07	97.14
oracle	4480	119:32	0		
es full	32000	131:03	–		
Unicode	27648	128:00	8	89.60	91.62
corpus	30336	129:26	2	97.08	99.15
oracle	3456	123:25	0		
zh full	32000	130:42	–		
Unicode	2688	114:39	13	75.43	83.36
corpus	28160	119:58	2	95.47	97.80
oracle	1536	126:16	0		

Table 2: CPU VT results for LLaMA-7B.

ference is performed in float32 precision, whereas GPU inference is in int8 (Dettmers et al., 2022).

4 CPU Results and Discussions

Upper bound performance First of all, we conduct an *oracle* vocabulary selection experiment to find the theoretical upper bound for speed and memory improvements: we run inference using the full vocabulary and we add the used vocabulary items to a *oracle* sub-vocabulary.

BLOOM versus LLaMA We present CPU results for the BLOOM family in Table 1 and those for LLaMA-7B in Table 2. We observe around 20% time improvements with the smaller BLOOM at 560M and 1B7, but only 5–10% in the 7B models. As the model grows in size, the oracle upper bound sees decreasing gains, due to the proportion of the embedding matrices becoming smaller in a larger model. By comparing BLOOM-7B1 with LLaMA-7B, we also find that the larger the base vocabulary, the more effective VT is. The oracle vocabulary is more than an order of magnitude smaller than our VT approaches, but in practice, it would be difficult to reduce the vocabulary size by as much.

Speed numbers of LLaMA-7B on CPU are relatively inconsistent and had wide variance across test runs. We attribute this to the small vocabulary size and thus less computational footprint in the output layer affected by VT. Also, there could be various scheduling issues and non-deterministic cache accesses as GEMM operations are split across the 40 cores of the CPU.

4.1 Script-based vocabulary trimming

When applying script-based filtering, we observe different trends in English and Spanish compared to Bulgarian and Chinese. For BLOOM, the sub-vocabulary size for Bulgarian and Chinese can be reduced to 10–20%, whereas for English and Spanish, it remains at 60%. This is potentially because BLOOM allocated more vocabulary items for European languages which are the dominant ones when the tokenizer is trained. Generally, the inference time reduces to between full and oracle vocabulary. In terms of misses, the model can maintain almost the same outputs with and without VT for English

and Spanish. However, there are 10–20% misses for Bulgarian and 30–40% for Chinese.

LLaMA-7B results are less favourable: script-based filtering does not significantly reduce the vocabulary size for English and Spanish, and all languages suffer from relatively high misses between 10–40%. Specifically for Bulgarian and Chinese, we argue that Unicode filtering could be too harsh as sometimes English characters are code-mixed in the language and cannot be avoided, e.g., when generating a website link. Therefore, we conclude that VT based on the writing script can improve inference efficiency without degrading performance for a multilingual LLM to generate Latin languages, but it is less feasible for non-Latin languages or English-centric LLMs with a smaller vocabulary.

4.2 Corpus-based vocabulary trimming

Corpus-based selection leaves a much larger vocabulary for Bulgarian and Chinese but reduces the vocabulary to half or less for English and Spanish. This method produces a more balanced sub-vocabulary for each language likely due to the inclusion of tokens outside of the desired language. However, for LLaMA-7B which has a small vocabulary in the first place, this approach keeps most of the entries for all languages and is thus not useful.

The corpus-based selection also ameliorates the quality problem to some extent by allowing for code-mixing (usually English), although the Chinese VT models still struggle to produce identical output as the full vocabulary models. Overall, we see a small but consistent reduction in runtime with BLOOM for this VT approach, indicating its practicality at least for English.

4.3 Memory

Besides speed considerations, VT can lead to ample memory footprint reduction, especially for smaller models like BLOOM-560M, where the model size is dominated by the vocabulary (nearly 50% of all model parameters). In practice, these models are small enough to fit in modern GPUs and CPUs, so the reduced memory is not game-changing. On the other hand, when looking at bigger models like BLOOM-7B1 or LLaMA-7B, vocabulary makes up just a tiny portion of the overall number of parameters and thus the relative reduction in model size is modest and could not enable the use of smaller GPUs. We can view this as a proxy judgement about the computational distribution of the model: The larger the model, the less time is spent in the output layer, and thus the smaller the impact of VT is. Exact memory numbers are available in Table 3.

Language	BLOOM				LLaMA	
	$\|V\|$	560M	1B7	7B1	$\|V\|$	7B
Full model	250680	2.10	6.10	27.10	32000	27.10
Embedding matrix or output layer						
full vocab	250680	0.90	1.90	3.80	32000	0.50
bg Unicode	22912	0.09	0.18	0.36	4736	0.07
bg corpus	58642	0.22	0.45	0.90	26496	0.41
en Unicode	186752	0.70	1.40	2.80	27520	0.43
en corpus	113024	0.44	0.88	1.70	30720	0.48
es Unicode	187008	0.70	1.40	2.80	27648	0.43
es corpus	112128	0.43	0.86	1.70	30336	0.47
zh Unicode	51584	0.20	0.40	0.80	2688	0.04
zh corpus	104320	0.40	0.80	1.60	28160	0.44

Table 3: Theoretical memory footprint (in GB) for BLOOM and LLaMA with float32 featuring the embedding matrix.

5 GPU results

In addition to CPU tests, we performed the same BLOOM experiments on a GPU and observed that all three selection criteria including the oracle do not lead to improved inference speed. Small performance differences might amount to little more than noise when the overhead of model slicing is considered. We hypothesize that GPUs are designed for multiplying large matrices, so reducing the matrix size, even to the extremity of an oracle sub-vocabulary, is not able to offer any speedup. This is consistent with Bogoychev et al. (2020)'s findings in applying shortlists to neural machine translation on GPUs. We list GPU results for BLOOM in Appendix B Table 4.

6 Conclusion

We presented a study of two straightforward language-inspired vocabulary trimming methods to speed up inference and save memory for large language model deployment. Experiments reveal ups and downs. While we can achieve speed improvements, it does not guarantee that the output is not altered compared to full vocabulary generation. With the models tested, we see the feasibility of our proposed approaches for English and Spanish, but there are shortcomings when considering languages written in non-Latin script and requiring code-mixing. In terms of efficiency, the reduction in inference time is less pronounced compared with memory saving.

Ethical Considerations

Our study aimed solely at reducing the computational resource consumption for deploying large language models. Our analysis contributes to the understanding of language heuristics in trimming an LLM vocabulary. While there is minimal risk associated with generating harmful content, it is no different for other research on large language models. We believe research into this direction has a positive impact in terms of energy saving and service deployment.

Acknowledgement

This work has received funding from UK Research and Innovation (UKRI) under the UK government's Horizon Europe funding guarantee [grant numbers 10052546 and 10039436].

References

Amine Abdaoui, Camille Pradel, and Grégoire Sigel. 2020. Load what you need: Smaller versions of mutililingual BERT. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*.

Nikolay Bogoychev and Pinzhen Chen. 2021. The highs and lows of simple lexical domain adaptation approaches for neural machine translation. In *Proceedings of the Second Workshop on Insights from Negative Results in NLP*.

Nikolay Bogoychev, Roman Grundkiewicz, Alham Fikri Aji, Maximiliana Behnke, Kenneth Heafield, Sidharth Kashyap, Emmanouil-Ioannis Farsarakis, and Mateusz Chudyk. 2020. Edinburgh's submissions to the 2020 machine translation efficiency task. In *Proceedings of the Fourth Workshop on Neural Generation and Translation*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*.

Pinzhen Chen, Shaoxiong Ji, Nikolay Bogoychev, Andrey Kutuzov, Barry Haddow, and Kenneth Heafield. 2024. Monolingual or multilingual instruction tuning: Which makes a better alpaca. In *Findings of the Association for Computational Linguistics: EACL 2024*.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Tobias Domhan, Eva Hasler, Ke Tran, Sony Trenous, Bill Byrne, and Felix Hieber. 2022. The devil is in the details: On the pitfalls of vocabulary selection in neural machine translation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, et al. 2023. Openassistant conversations - democratizing large language model alignment. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Hai Son Le, Alexandre Allauzen, and François Yvon. 2012. Continuous space translation models with neural networks. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Openai.com.

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

Holger Schwenk, Vishrav Chaudhary, Shuo Sun, Hongyu Gong, and Francisco Guzmán. 2021. WikiMatrix: Mining 135M parallel sentences in 1620 language pairs from Wikipedia. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*.

Holger Schwenk, Marta R. Costa-jussà, and Jose A. R. Fonollosa. 2007. Smooth bilingual n-gram translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. GitHub repository.

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

Asahi Ushio, Yi Zhou, and Jose Camacho-Collados. 2023. Efficient multilingual language model compression through vocabulary trimming. In *Findings of the Association for Computational Linguistics: EMNLP 2023*.

A Languages

We experimented with Bulgarian, Chinese, English, and Spanish, to cover different conditions and use cases regarding writing scripts and text usage. English and Spanish use the same script and have a high overlap in vocabulary with many other languages after tokenization. Since LLMs are English-centric, we examine how effective of a sub-vocabulary we can find when it is not possible to shortlist merely based on the script. Bulgarian is a low-resource language written in the Cyrillic script. Most multilingual language models have lower amounts of Cyrillic tokens, so we expect that script-based filtering will leave a small sub-vocabulary; however, since Cyrillic is used by other languages, we will inevitably end up with vocabulary items that do not belong to Bulgarian. Finally, Chinese is a high-resource language with a unique script; Unicode filtering would be the most effective in this case.

B GPU Performance

We provide GPU performance numbers in Table 4. Unfortunately, neither of the language-based vocabulary trimming methods can improve time efficiency.

Language	$\|V\|$	BLOOM -560M		BLOOM -1B7		BLOOM -7B1	
		time	miss	time	miss	time	miss
bg full	250680	05:22	–	08:29	–	14:43	–
Unicode	22912	05:23	0	08:45	6	14:35	17
corpus	58642	05:22	0	08:38	1	14:33	10
oracle	1408	05:21	0	09:06	0	14:33	0
en full	250680	06:50	–	09:02	–	11:54	–
Unicode	186752	06:54	0	08:52	0	11:46	0
corpus	113024	06:38	2	08:56	3	11:59	3
oracle	4736	06:43	0	09:00	0	11:52	0
es full	250680	06:17	–	07:05	–	12:35	–
Unicode	187008	06:13	0	07:03	0	12:17	0
corpus	112128	06:15	1	7:10	3	12:30	3
oracle	4736	06:26	0	07:23	0	12:17	0
zh full	250680	05:37	–	08:47	–	11:58	–
Unicode	51584	06:10	15	08:34	20	11:22	29
corpus	104320	06:03	11	09:01	16	11:35	13
oracle	4096	05:35	0	08:42	0	11:46	0

Table 4: GPU VT results for the BLOOM family.