Large Language Model Inference with Lexical Shortlisting

Anonymous ACL submission

Abstract

Deploying large language models (LLMs) often encounters challenges due to intensive computational and memory requirements. Our research delves into lexical shortlisting, aiming to bolster efficiency and deployment readiness. While lexical shortlisting has been shown effective in tasks like machine translation, tailoring them to LLMs demands specific modifications given the diverse nature of their applications. We study two heuristics to shortlist sub-vocabulary at LLM inference time: Unicode-based script filtering and corpus-based selection. The work explores different LLM families and sizes. It is observed that lexical shortlisting can reduce the memory usage of some models by nearly 50% and has an upper bound of 25% improvement in generation speed. This preliminary study delineates the strengths of vocabulary selection, acknowledges the limitations of these methods, and finally proposes future avenues for refining.

1 Introduction

Large language models are gaining increasing attention given their strong performance in various natural language tasks (Radford et al., 2019; Brown et al., 2020; Kaplan et al., 2020; Ouyang et al., 2022). Most LLMs are Transformer-based (Vaswani et al., 2017; Scao et al., 2022; Touvron et al., 2023), which entail a costly matrix multiplication $D \times |V|$ in the output layer, where D is the output layer hidden size and $|V|$ denotes the size of a vocabulary V. This expensive operation leads to increased inference cost of both memory and speed given the autoregressive nature of LLM decoding. Given their substantial size, this latency in inference significantly escalates the expense of LLM deployment.

In practical scenarios, choosing a sub-vocabulary V' with $|V'| \ll |V|$, and only loading the corresponding sub-embedding matrix for inference seems favourable since the majority of the logits from the output layer do not affect the hypothesis token(s) at each time step. This has been actively explored in machine translation (Schwenk et al., 2007; Le et al., 2012; Devlin et al., 2014; Bogoychev et al., 2022). It can result in a 10-to-30-fold reduction in vocabulary size, massively speeding up the decoding process.

For translation, vocabulary selection involves pre-computing word-level alignments and making potential translated words a sub-vocabulary. On the other hand, shortlisting in LLMs poses a fundamental challenge: often LLM outputs are variable and open-ended, complicating the straightforward determination of the required lexicons.

This pilot study adapts lexical shortlisting to LLMs. We propose and experiment with two strategies: script-based token filtering where vocabulary items are removed if they do not belong to the output language, and corpus-based preselection where we keep items based on vocabulary hits from a large representative corpus. Our contributions can be summarized as follows:

- We propose two selection methods: writing script-based and corpus-based.
- We experiment with various LLM families and sizes (LLaMA and BLOOM) and report varying behaviours.
- We measure speed-ups for different hardware and the upper bound of memory reduction.
- We discuss the strengths and limitations of applying these two methods in the wild.

2 Vocabulary Shortlisting

Large language models, especially multilingual ones, hold vocabulary items for many languages and scripts, which are rarely required simultaneously in a single generation pass. We remove un-

necessary vocabulary tokens according to the language of the query. We describe two ways to prepare sub-vocabulary for LLMs, both focusing on retaining tokens relevant only to the language being generated. We test on the fly in a batched setting: we determine a sub-vocabulary for an entire batch because creating the sub-vocabulary separately for each input is too expensive. Furthermore, we also always include all tokens appearing in the inputs.

Script-based selection We propose to only keep the tokens which match the writing script of a desired language. This can be done by filtering token strings based on the Unicode range. 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 effective if the writing system is used in many languages, e.g. Cyrillic or Latin alphabets. It would be infeasible to limit the sub-vocabulary to the lexicons that solely belong to a specific language, resulting in a relatively large vocabulary. Moreover, this method would strictly rule out code-mixed tokens, emojis, etc which are used in real-world communications.

Corpus-based selection A more comprehensive 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 inference prompts.

3 Experimental Setup

3.1 Languages

We experimented on four languages: Bulgarian, Chinese, English, and Spanish, to offer distinct conditions that cover most use cases. English and Spanish use the same script and would have a high overlap in vocabulary after tokenization. Since we test on English-centric models, we examine how good 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 a number of languages, we will inevitably end up with vocabulary items that do not belong to the Bulgarian. Finally, Chinese is a high-resource language with a unique script; Unicode filtering would be the most effective in this case.

3.2 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. (2023)'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 shortlisting as explained earlier.

BLOOM is a multilingual LLM that explicitly supports English, Spanish and Chinese, but not Bulgarian. Consequently, it has a sizeable vocabulary of 250K and is therefore a prime 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. In this case, we might have reduced benefit from vocabulary shortlisting, because a proportionally lower amount of computation occurs in the output layer. On the other hand, since the LLM is English-focus, we expect drastic vocabulary reductions compared to BLOOM for Bulgarian and Chinese.

3.3 Evaluation

We test on 50 prompt questions from OpenAssistant (Köpf et al., 2023); we human-translate these into all testing languages. We decode them with beam size 1. The time taken to decode the entire test set is measured end-to-end, including model loading, and embedding slicing. As a quality indicator, we count the number of times a model fails to produce *the exact same output* with a full vocabulary and with a shortlisted vocabulary, given the same input. We refer to this as *miss*, which ideally should be zero suggesting no quality impact.

3.4 Shortlisting details

For vocabulary selection, we tokenize the test inputs and always include the tokens in the sub-vocabulary. We then apply either of the proposed selection methods. Script-based selection checks whether a token falls in a specific Unicode subset: Cyrillic for Bulgarian, ASCII for English, Latin Extended-A for Spanish, and Chinese

Language	$\|V\|$	BLOOM-560M		BLOOM-1B7		BLOOM-7B1	
		time	miss	time	miss	time	miss
bg full	250680	05:26	–	15:18	–	65:01	–
Unicode	22912	04:39	1	13:44	4	51:46	10
corpus	58642	04:49	0	09:34	1	60:28	3
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	16:05	0	58:18	0
corpus	113024	07:00	1	15:08	3	54:20	2
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	57:41	4
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	13:09	21	50:50	22
corpus	104320	06:08	11	14:08	16	46:39	17
oracle	4096	05:16	0	12:07	0	50:50	0

Table 1: CPU shortlisting results for BLOOM.

Language	$\|V\|$	LLaMA-7B	
		time	miss
bg full	32000	117:15	–
Unicode	4736	125:55	19
corpus	26496	132:24	5
oracle	2048	123:06	0
en full	32000	113:52	–
Unicode	27520	125:57	6
corpus	30720	111:30	19
oracle	4480	119:32	0
es full	32000	131:03	–
Unicode	27648	128:00	8
corpus	30336	129:26	2
oracle	3456	123:25	0
zh full	32000	130:42	–
Unicode	2688	114:39	13
corpus	28160	119:58	2
oracle	1536	126:16	0

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

characters for Chinese. For corpus-based selection, we utilize a subset of the WikiMatrix corpus (Schwenk et al., 2021) which contains Wikipedia texts for each language. For both selection methods, we keep the first 300 vocabulary entries too, as those usually correspond to special tokens, Unicode bytes (for byte-level BPE), numbers, etc.

We pre-compute the vocabulary subset offline and we do not record the time spent on pre-tokenizing a large corpus or extracting a Unicode subset in the measurements, as these can be reused for every batch during inference once done. Script-based filtering takes under 60 seconds and corpus-based selection takes up to 10 minutes. Adding the inputs' tokens to the existing pre-selected sub-vocabulary is extremely fast.

3.5 Upper bound performance

We conduct an oracle vocabulary selection experiment to find the theoretical upper bound for speed and memory improvements: we run inference using full vocabulary and we select the used vocabulary items for the oracle sub-vocabulary.

3.6 Hardware

We perform our 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 an RTX 3090. Inference on CPU is performed in float32 precision, whereas on GPU it is in int8 following Dettmers et al. (2022).

4 CPU Results and Discussions

We present CPU results on the BLOOM family in Table 1 and the results on LLaMA-7B in Table 2. We observe around 20% improvements with the smaller BLOOM-560M and BLOOM-1B7, but only 5-10% in the 7B models. As the models grow in size, the *oracle* upper bound gain decreases, due to the proportion of the embedding and output matrices becomes smaller in a larger model. By comparing BLOOM-7B1 with LLaMA-7B, we also find that the larger the base vocabulary, the more effective shortlisting is. We note that the oracle vocabulary is more than an order of magnitude smaller than the other shortlisting 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 shortlisting. Also, there could be various scheduling issues and non-deterministic cache accesses as GEMM operations are split across the 40 cores of the CPU we used.

4.1 Script-based shortlisting

When applying script-based shortlisting, 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 only 10-20%, whereas for English and Spanish, it has about 60% of the origi-

nal size. 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 using the full vocabulary and the oracle shortlisting. In terms of misses, the model can maintain almost the same outputs with and without shortlisting 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 shortlisting 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 shortlisting 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 shortlisting

Corpus-based shortlisting leaves a much larger vocabulary for Bulgarian and Chinese, but reduces the vocabulary to half or less for English and Spanish. This method achieves a more balanced shortlisting effect for each language potentially because of the inclusion of many tokens outside of the output language's writing system. However, for LLaMA-7B which has a small vocabulary in the first place, this approach keeps the majority of the entries for all languages and is thus not useful.

Corpus-based shortlisting also ameliorates the quality problem to some extent, although the Chinese models still struggle to produce identical output as the full vocabulary models. Overall, we see a small but consistent reduction in runtime with BLOOM too for this shortlisting approach, indicating its practicality at least for English.

4.3 Memory

Lexical shortlisting leads 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 use 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 shortlisting is. Exact memory numbers are available in the appendix.

5 GPU results

In addition to CPU tests, we performed the same BLOOM experiments on GPU and we 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. Exact GPU performance numbers available in the appendix.

6 Related Work

Gee et al. (2022) switched the vocabulary of a pre-trained language model to a smaller one that improves inference speed, but it requires fine tuning.

The most similar to our work are vocabulary trimming by Abdaoui et al. (2020) and Ushio et al. (2023). They slice the vocabulary of a model based on language criteria. Their work focuses on reducing the model size with no speed considerations. Furthermore, they only target models with very large vocabularies.

7 Conclusion

We presented a study of using lexical shortlisting to speed up inference with large language models. 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.

8 Limitations

We used (mis)matches to indicate the quality of generation when applying shortlisting. However, this is a strict metric that penalizes a different output regardless of its length or the number of incorrect tokens. Since the inference time of an autoregressive LLM can be affected by the number of tokens generated for each input, it is reasonable to include metrics like BLEU and ROUGE to take the generation length or recall into account. Furthermore, the shortlisted vocabulary, inference time, and mismatch are entangled and we are not aware of an ideal evaluation setup.

9 Risks

This work is aimed solely at reducing the computational resources necessary for running large language models, thus we see no risks associated with it.

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*, pages 119–123, Online. Association for Computational Linguistics.

Nikolay Bogoychev, Maximiliana Behnke, Jelmer Van Der Linde, Graeme Nail, Kenneth Heafield, Biao Zhang, and Sidharth Kashyap. 2022. Edinburgh's submission to the WMT 2022 efficiency task. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 661–667, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

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*, pages 218–224, Online. Association for Computational Linguistics.

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, Barry Haddow, and Kenneth Heafield. 2023. Monolingual or multilingual instruction tuning: Which makes a better Alpaca. *arXiv preprint arXiv:2309.08958*.

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)*, pages 1370–1380, Baltimore, Maryland. Association for Computational Linguistics.

Leonidas Gee, Andrea Zugarini, Leonardo Rigutini, and Paolo Torroni. 2022. Fast vocabulary transfer for language model compression. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 409–416, Abu Dhabi, UAE. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint*.

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*, pages 39–48, Montréal, Canada. Association for Computational Linguistics.

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

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*, pages 1351–1361, Online. Association for Computational Linguistics.

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)*, pages 430–438, Prague, Czech Republic. Association for Computational Linguistics.

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. An efficient multilingual language model compression through vocabulary trimming. *arXiv preprint arXiv:2305.15020*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

A Memory

Detailed memory footprints shown on 3.

B GPU Performance

Exact GPU performance numbers on Table 4

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.

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 shortlisting results for BLOOM.