Sheer Luck or Linguistic Properties Behind LLM Pre-Pretraining? An Investigation on Multiple Languages

Anonymous ACL submission

Abstract

Pretraining LLMs on artificial languages ("pre-pretraining") is a technique that could reportedly increase token efficiency by 30%, i.e., save up to 30% of training tokens needed to reach a certain performance. We validate this prior result for English on a larger set of natural languages across four language families, using two different tokenizers and varying model sizes. We relate the observed gains (or losses) in token efficiency to quantified linguistic properties of the languages, such as sentence length, morphological richness, and features of dependency syntactic trees (tree depth, maximum number of children, number of crossing dependencies). Our empirical results indicate that the reported gains should be primarily attributed to luck in random seed choice, although we can confirm the trend of stable gains with 128-Dyck pretraining of small models with the Llama tokenizer and the trend of diminishing gains with increasing model size.

1 Introduction

Language models require more training data as their size grows, which poses a challenge for low-resource languages and specific domains. Transfer learning, i.e., benefiting from the knowledge in a different dataset, is one of the established techniques for this situation.

Previous work in the machine translation area (Kocmi and Bojar, 2018; Lin et al., 2019) have shown that initial training on a related or even fully unrelated language pairs improves performance on the language pair of interest. Later, gains were also observed when pretraining (or "pre-pretraining") large language models (LLMs) on artificial languages (Ri and Tsuruoka, 2022; Papadimitriou and Jurafsky, 2023; Hu et al., 2025). However, these recent studies on LLMs were solely focused on English. To our knowledge, it remains unclear whether the method consistently improves token efficiency across other natural languages, model sizes, and architectures.

In this work, we go beyond the previous studies of LLM transfer learning from artificial languages in two main directions: we extend (1) the set of formal languages used, and (2) the set of natural languages of different families, seeking a relation between the properties of the artificial and natural languages affecting the effectiveness of the method.

Experimental results suggest that for small model sizes, the pretraining method helps across different natural languages, although the observed gains are also highly dependent on the particular training run and are sensitive to the setup details.

2 Related Work

Ri and Tsuruoka (2022) have shown that training LSTM- and Transformer-based language models on artificial languages can provide knowledge transfer to natural ones. They conclude that `Nesting Dependency` (see below) outperforms `Flat Dependency` in importance for language modeling.

Subsequent studies in Papadimitriou and Jurafsky (2023) and the ACL outstanding paper by Hu et al. (2025) claim that applying the same method to a decoder-only LLM can improve its language acquisition, reducing perplexity and resulting in a 33% token efficiency gain for the Pythia 1B model.

Kumar et al. (2025) note that pretraining the model on artificial languages (`NEST` and `CROSS`)[1] improves upon standard pretraining on textual data. But unlike Ri and Tsuruoka (2022), they observe that pretraining on `NEST` degrades performance, while `CROSS` leads to improvements.

Budnikov and Yamshchikov (2025) suggest that some artificial languages used in pretraining lead to structurally richer token embeddings but they fail to see any direct transfer of this to English.

[1] `NEST` is equivalent to `Nesting Dependency`; `CROSS` is equivalent to `Flat Dependency`

3 Dataset

3.1 Formal Languages for Pretraining

Among formal languages, we consider k-Dyck (demonstrating `Nesting Dependency`) and k-Shuffle Dyck (`Flat Dependency`) languages for $k = 64$ and $k = 128$, which are defined as follows:

k-Dyck contains k different types of structurally-nested dependencies, each of which is denoted by matching pair of opening and closing brackets, e.g., `( [ { } ] )` or `( [ ] ) { }`.

k-Shuffle Dyck allows breaking the well-nestedness (e.g., `( [ ] { ) }`) and including cross-serial dependencies (e.g., `( [ { ) ] }`).

We follow the approach of Hu et al. (2025) and represent the bracket symbols with non-negative integers. In other words, if an opening symbol is denoted as the number n, its corresponding closing symbol is represented as $n + k$, where k is the number of distinct dependencies, and $0 \leq n < k$. An example pair of opening and closing symbols for $k = 128$ is `("81", "209")`.

3.2 Natural Languages

Aside from English, we analyze the effects of pretraining for five other typologically diverse European languages from four language families, namely Czech (Slavic), Finnish (Uralic), Danish and Dutch (Germanic), and Albanian (Albanoid).

We construct the training dataset for each language as a monolingual, multi-domain dataset by mixing either FineWeb (for English; Penedo et al. 2024) or FineWeb 2 (for the other five languages; Penedo et al. 2025) with OpenSubtitles[2] (Lison and Tiedemann, 2016) and EUBookshop[3] (Tiedemann, 2012), in proportions of 8:1:1, respectively.

Importantly, we attempt to isolate the effect of pretraining corpus size observed by Kocmi and Bojar (2018): a larger pretraining corpus always delivers bigger gains. Across all experiments, our pretraining corpus consists of 16,000 Dyck sequences of 2,048 tokens (which corresponds to 500 training steps with the batch size of 32) and our natural language corpus used afterwards is 320,000 sequences[4] of 2,048 tokens (i.e., 10,000 steps with the batch size of 32), subject to minor variations due to tokenization.

For evaluation of Llama-based models, we use the first 2,000 rows of the validation split of the Multilingual Colossal Clean Crawled Corpus (mC4) dataset.[5] For the other set of models (§4.1), we use a separate and smaller portion of the mixed multi-domain dataset.

4 Experiments and Results

4.1 Experiment Setups

We experiment with four model sizes, based on Llama 3 architecture (Grattafiori et al., 2024). Each model is accompanied by either Llama 3 or Gemma 3 tokenizer (Team et al., 2025). This results in eight different models—ranging from 250M to 1.4B—which are outlined in Table 1.

Config	# of parameters			Name
	Model itself	Embedding	Total	
0	154M	99M	253M	`154M+LT`
		201M	355M	`154M+GT`
1	308M	131M	440M	`308M+LT`
		268M	577M	`308M+GT`
2	481M	197M	678M	`481M+LT`
		403M	884M	`481M+GT`
3	856M	263M	1.1B	`856M+LT`
		537M	1.4B	`856M+GT`

Table 1: Overview of models. `LT` and `GT` stand for models with Llama and Gemma tokenizer, respectively.

An important difference for our artificial language presentation is that Gemma tokenizer splits each number into separate digits,[6] different from Llama tokenizer.[7]

Each model is trained for 1 epoch on the full training dataset (§3.2) with three approaches: (1) without pretraining (`Base`); (2) after being pretrained with k-Dyck (`{64,128}-Dyck`); and (3) after being pretrained with k-Shuffle Dyck (`{64,128}-Shuff Dyck`).

In addition, we conduct a trial investigation on two other models of bigger sizes, namely `HPLT/hplt2c_eng_checkpoints` (a 2.15B model also based on Llama 3 architecture) and `EleutherAI/pythia-2.8b` (a 2.8B model from the Pythia collection; Biderman et al. 2023). We ignore the published weights and train from scratch (1) on our natural language data only, and (2) on 128-Shuffle Dyck followed by our natural language data. For these models, the evaluation dataset is a small portion of the mixed dataset (same as §3.2).

[2] `https://opus.nlpl.eu/datasets/OpenSubtitles`

[3] `https://opus.nlpl.eu/datasets/EUbookshop`

[4] In line with the practice of training LLMs but different from machine translation area, our sequences span sentence boundaries.

[5] `allenai/c4`

[6] e.g., "81 209" → `['8', '1', '_', '2', '0', '9']`

[7] e.g., "81 209" → `['81', 'Ġ', '209']`

Details of hyper-parameters used in our training are described in Appendix A.

We quantify the improvement by token efficiency gain, which reflects how many training tokens sooner the model pretrained on the artificial language has reached the same validation loss as the corresponding `Base` model. If the pretrained model performs worse, we negate the value. The exact calculation is provided in Appendix A.3.

4.2 Results

Our results show that for small models, pretraining helps across all the languages (Figures 1 and 4) in certain setups. We also observe that gains diminish with increasing model size (Figures 2 and 5).

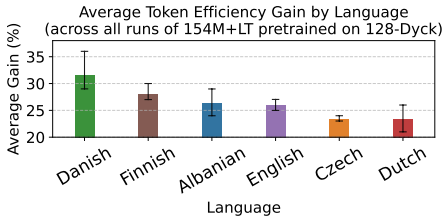

Figure 1: Average token efficiency gains by language (for `154M+LT` pretrained on 128-Dyck)

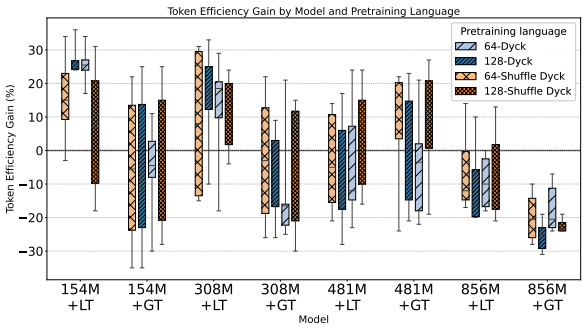

Figure 2: Token efficiency gain across single runs of all models, natural and pretraining languages. The gain for each method is averaged across all six languages.

Notably, smaller models (`154M` and `308M`) with Llama tokenizer (`LT`) on average have higher token efficiency gains and lower gain variance than those with Gemma tokenizer (`GT`).

Models with `LT` show improvements across all artificial languages, with k-Dyck often outperforming k-Shuffle Dyck. In contrast, those with `GT` benefit more from {64,128}-Shuffle Dyck. Details of all results are provided in Appendix B.

4.3 (In)Consistency of Results

For the smallest models (`154M+{LT,GT}`), across all pretraining and natural languages, we collect three training runs with identical setups but different seed choices. The results reveal unexpectedly high variance leading to inconsistent trends.

Pretraining `154M+LT` on 128-Dyck yields stable efficiency gains for all languages. Pretraining it on 64-Dyck is beneficial for five languages, but is inconsistent for Czech. Pretraining `154M+GT` on 128-Shuffle Dyck is helpful only for Albanian and English, whereas for the rest of the languages, it does not bring any gain. Pretraining `154M+GT` on any setup and `154M+LT` on {64,128}-Shuffle Dyck leads to different results across setups, and may be inconsistent (Table 4 and Figures 6 and 7).

An interesting pattern is observed across all pretraining languages: `154M+LT` trained on English gains efficiency, whereas `154M+GT` trained on Finnish degrades (Figure 8).

5 Language Specific Analysis

5.1 Linguistic Metrics

We estimate the following linguistic metrics from samples of 1M tokens per language: ① *Percentage of crossing dependencies*; ② *Mean dependency distance*; ③ *Maximum dependency distance*; ④ *Mean dependency tree depth*; ⑤ *Maximum dependency tree depth*; ⑥ *Vocabulary size*; ⑦ *Average sentence length*; ⑧ *Average number of children per head*; ⑨ *Maximum number of children per head*;

These properties are calculated both including and excluding punctuation. The head of a punctuation token is typically the syntactic head of the clause or phrase it punctuates, which can make the properties replicate measures of sentence length rather than providing structural insights.

One exception feature is ⑩ *Morphological richness*, which is estimated on the Bible dataset (Christodouloupoulos and Steedman, 2015), covering languages in a multi-parallel manner.

We use the multi-lingual `gemma-3-12B-pt` model (Team et al., 2025) to measure two other special features on the same 1M tokens, namely ⑪ *Average perplexity* and ⑫ *Variance of the perplexity* to quantify the predictability and repetitiveness of data (i.e., higher values should indicate more diverse data).

See Appendix C for definitions and details.

5.2 Correlation Analysis

Might there be any correlation between the efficiency gain and the linguistic properties? We attempt to identify the relationship, but fail to detect any clear pattern (Figures 19 and 20).

Nevertheless, we can still name several features that most strongly influence gains on average across all setups with noteworthy correlations. Languages with high values of ④, ⑦, ⑧, and ⑪ in the test set seem to benefit most from pretraining (Figure 11). The correlations with the training data (reflecting the general language properties, not the particular test set) show somewhat different tendencies. The relationships with ② and ⑫ become slightly stronger, while weaker with ④ (Figure 12).

From these moderate correlations, we assume that the diversity of syntactic structures and longer sentences contribute to the benefits of pretraining.

More precise analysis shows that models with `LT` benefit when test data has high ④ and ⑫. While models with `GT` take advantage of training data with high ⑫ and ②, as well as of high ⑦, ⑧, and ⑪ in either dataset (Figures 13 to 16). For boxplots summarizing gains across model sizes and tokenizers, see Figure 9.

Relation to pretraining data. We assume that high ④, ⑦, and ⑧ require tracking multiple heads simultaneously, similarly to tracking multiple opening symbols. High ② relates to the large distance between an opening and its corresponding closing symbol. High ⑪ and ⑫ reflect the diversity of syntactic structures. This is captured by k-Shuffle Dyck, where pairs of one symbol type can cross pairs of other types, complicating the structure.

Consequently, high values for these features should correlate positively with the efficiency gains from pretraining on k-Dyck and k-Shuffle Dyck.

Crossing dependencies. Our primary hypothesis was that models pretrained on k-Shuffle Dyck would achieve the largest performance gains on data rich in crossing dependencies (with high ①).

Based on the distribution of crossing dependencies (Figure 3), we expected the performance gains to follow the order Dutch > Czech > Finnish > Danish > Albanian > English. However, we do not observe this trend in our experiments (Figure 4), and do not detect a correlation between gains of models pretrained on k-Shuffle Dyck and the property ① in either dataset (Figures 17 and 18).

Perplexity. Notably, the Pythia 2.8B model, pretrained on 128-Shuffle Dyck and subsequently trained on Albanian multi-domain data, achieves an efficiency gain of nearly 62%, which is almost twice as high as for the rest of the models.

We hypothesize that data in low-resource languages, such as Albanian, can be repetitive, and the data from training and evaluation datasets may be very similar, leading to outlying performance. To examine this, we measure ⑪ and ⑫ to quantify the diversity of data. Analysis shows that the Albanian data is comparable in diversity to other languages across both datasets, suggesting that the gain is not due to data repetitiveness (Table 6).

All details of our language-specific analysis are provided in Appendix D.

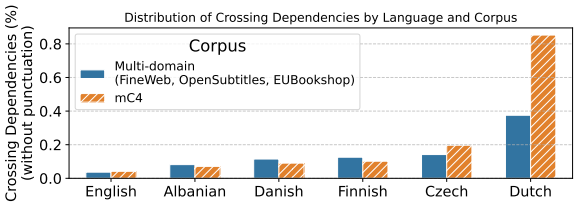

Figure 3: Distribution of crossing dependencies by language and corpus

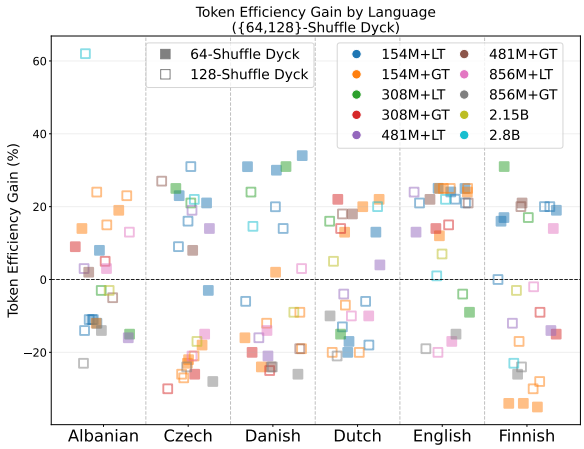

Figure 4: Efficiency gain comparison across a single run of models pretrained on {64,128}-Shuffle Dyck

6 Conclusion

In this work, we investigate the benefit of pretraining LLMs with artificial languages by extending the experiments to five other natural languages aside from English. We observe that k-Dyck pretraining outperforms k-Shuffle Dyck, with 128-Dyck surpassing 64-Dyck. Our results suggest the approach is language-independent, but it is highly sensitive to setup, and the gains diminish with increasing model scale. We show that the most stability is achieved with small models using the Llama tokenizer and pretrained on 128-Dyck; while models with the Gemma tokenizer pretrained on 128-Shuffle Dyck produce language-dependent results. Importantly, we note that results for other setups are related to pure luck in random seed choice, requiring multiple runs to achieve desirable and reliable performance.

Limitations

The work presented in this paper faces several limitations that restricted us from having more comprehensive results.

First, one noticeable issue concerns the easily understandable lack of a large-scale, multi-parallel dataset in multiple languages, both in training and evaluation. Having this would instead allow us to have better and fairer comparisons between models. The currently available datasets are mostly from the Internet, which might not reflect the highest quality in the language modeling task.

Second, due to the sheer scale of the experiments, we can only train each model with 10,000 training steps (which corresponds to approximately 655M tokens). In addition, on the technical side, we fail to conduct more experiments with a wider range and combination of training hyper-parameters. Even though the resulting models successfully converged during training, we cannot claim that our chosen set of training hyper-parameters is the best one.

References

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, and 1 others. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Ondřej Bojar. 2015. Machine Translation. In *The Oxford Handbook of Inflection*. Oxford University Press.

Mikhail Budnikov and Ivan Yamshchikov. 2025. Transfer of Structural Knowledge from Synthetic Languages. In *Proceedings of the 1st Joint Workshop on Large Language Models and Structure Modeling (XLLM 2025)*, pages 242–251, Vienna, Austria. Association for Computational Linguistics.

Christos Christodouloupoulos and Mark Steedman. 2015. A massively parallel corpus: the bible in 100 languages. *Language resources and evaluation*, 49(2):375–395.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The Llama 3 Herd of Models. *Preprint*, arXiv:2407.21783.

Michael Y. Hu, Jackson Petty, Chuan Shi, William Merrill, and Tal Linzen. 2025. Between Circuits and Chomsky: Pre-pretraining on Formal Languages Imparts Linguistic Biases. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9691–9709, Vienna, Austria. Association for Computational Linguistics.

Tom Kocmi and Ondřej Bojar. 2018. Trivial Transfer Learning for Low-Resource Neural Machine Translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 244–252, Brussels, Belgium. Association for Computational Linguistics.

Nalin Kumar, Mateusz Lango, and Ondrej Dusek. 2025. Pretraining Language Models with LoRA and Artificial Languages. In *Proceedings of the First BabyLM Workshop*, pages 525–530, Suzhou, China. Association for Computational Linguistics.

Yu-Hsiang Lin, Chian-Yu Chen, Jean Lee, Zirui Li, Yuyan Zhang, Mengzhou Xia, Shruti Rijhwani, Junxian He, Zhisong Zhang, Xuezhe Ma, Antonios Anastasopoulos, Patrick Littell, and Graham Neubig. 2019. Choosing Transfer Languages for Cross-Lingual Learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3125–3135, Florence, Italy. Association for Computational Linguistics.

Pierre Lison and Jörg Tiedemann. 2016. OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 923–929, Portorož, Slovenia. European Language Resources Association (ELRA).

Isabel Papadimitriou and Dan Jurafsky. 2023. Injecting structural hints: Using language models to study inductive biases in language learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8402–8413, Singapore. Association for Computational Linguistics.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Guilherme Penedo, Hynek Kydlíček, Vinko Sabolčec, Bettina Messmer, Negar Foroutan, Amir Hossein Kargaran, Colin Raffel, Martin Jaggi, Leandro Von Werra, and Thomas Wolf. 2025. FineWeb2: One Pipeline to Scale Them All – Adapting Pre-Training Data Processing to Every Language. *Preprint*, arXiv:2506.20920.

Ryokan Ri and Yoshimasa Tsuruoka. 2022. Pretraining with Artificial Language: Studying Transferable Knowledge in Language Models. In *Proceedings of the 60th Annual Meeting of the Association for*

Computational Linguistics (Volume 1: Long Papers), pages 7302–7315, Dublin, Ireland. Association for Computational Linguistics.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 Technical Report. *Preprint*, arXiv:2503.19786.

Jörg Tiedemann. 2012. Parallel Data, Tools and Interfaces in OPUS. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

A Details of Experiment Setups

A.1 Model hyper-parameters

We take the model configuration of the published `meta-llama/Llama-3.2-1B` checkpoint, then modify four hyper-parameters as described in Table 2.

Config	0	1	2	3
`num_hidden_layers`	8	12	16	16
`num_attn_heads`	8	8	16	16
`head_dim`	16	16	32	32
`hidden_size`	768	1,024	1,536	2,048

Table 2: Details of models' hyper-parameters.

A.2 Training hyper-parameters

Table 3 details chosen training hyper-parameters.

Training hyper-parameters	Value
Effective batch size	32
Learning rate	5e-4
Optimizer	AdamW
LR Scheduler	cosine
Min. learning rate	0.1
Warmup steps	1,000
Weight decay	0.1
Max. grad. norm.	1.0

Table 3: Training hyper-parameters

A.3 Algorithm to Calculate Token Efficiency Gain

The algorithm to calculate the token efficiency gain is described in Algorithm 1.

Algorithm 1: Calculate Token Efficiency Gain

Input: Sequence of the baseline model's validation losses B, Sequence of the pretrained model's validation losses P

Output: Token Efficiency Gain (%)

$BaselineTokens \leftarrow$ Total number of tokens used by the baseline model
$TargetLoss \leftarrow$ Final value in B

if $i \geq TargetLoss \; \forall i \in P$ **then**
  $Gain \leftarrow$ Calculate Token Efficiency Gain(P, B)
  return $-Gain$
end
else
  $n \leftarrow length(P)$
  $k \leftarrow \min\{j \in \{0, \ldots, n-1\}$ such that $\forall i, j \leq i < n : P[i] \leq TargetLoss\}$
  $PretrainedTokens \leftarrow$ Number of tokens used by the pretrained model at index k
  $Gain \leftarrow (1 - \frac{PretrainedTokens}{BaselineTokens}) \times 100$
  return $Gain$
end

B Details of Token Efficiency Gain

Table 4 describes the consistency of evaluation results for the same model (Llama `154M`) with different tokenizers (Llama - `LT` or Gemma - `GT`) across all pretraining and natural languages. Symbols '+', '−', and '×' denote consistent gain, consistent loss, and inconsistency across three runs (i.e., some of the runs resulted in efficiency gain, while others in loss or negligible (near-zero) gain), respectively.

Figure 5 describes how the token efficiency gain of the Llama models with `LT` pretrained on 128-Dyck decreases as the model scale grows.

Figures 6 to 8 describe how token efficiency gain differs between Llama `154M` with different tokenizers (`LT` or `GT`) across all pretraining and natural languages, and three runs.

Figures 9 and 10 describe how the token efficiency gain differs between all model setups, runs, pretraining, and natural languages.

Table 5 provides performance gain (or loss) results for all experiments.

Natural Language	Pretraining Language								Summary		
	64-Dyck		128-Dyck		64-Shuff Dyck		128-Shuff Dyck		LT	GT	Overall
Albanian	+	×	+	−	×	×	−	+	2+ 1× 1−	1+ 2× 1−	3+ 3× 2−
Czech	×	×	+	×	×	−	+	−	2+ 2× 0−	0+ 2× 2−	2+ 4× 2−
Danish	+	+	+	−	+	×	×	−	3+ 1× 0−	1+ 1× 2−	4+ 2× 2−
Dutch	+	×	+	+	×	+	−	−	2+ 1× 1−	2+ 1× 1−	4+ 2× 2−
English	+	×	+	+	+	+	+	+	4+ 0× 0−	3+ 1× 0−	7+ 1× 0−
Finnish	+	−	+	−	+	−	×	−	3+ 1× 0−	0+ 0× 4−	3+ 1× 4−

Table 4: Consistency of the results of pretraining with the 154M models. We only indicate if the setup lead to a consistent gain (+) over the Base model in all three runs, to a consistent loss (−), or to a mixed result (×) where some of the runs were helpful and some harmful. In each pretraining language group, the left and right columns represent the result with LT and GT, respectively. The summaries report the number of consistent gains, mixed outcomes and losses across the respective setting.

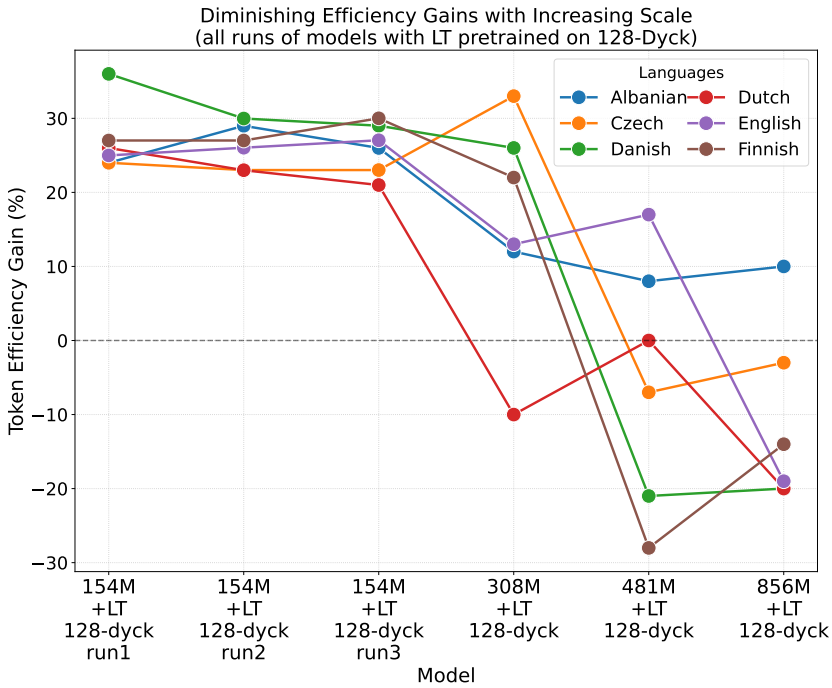

Figure 5: Token efficiency gains diminishing with increasing scale (across the models with LT pretrained on 128-Dyck)

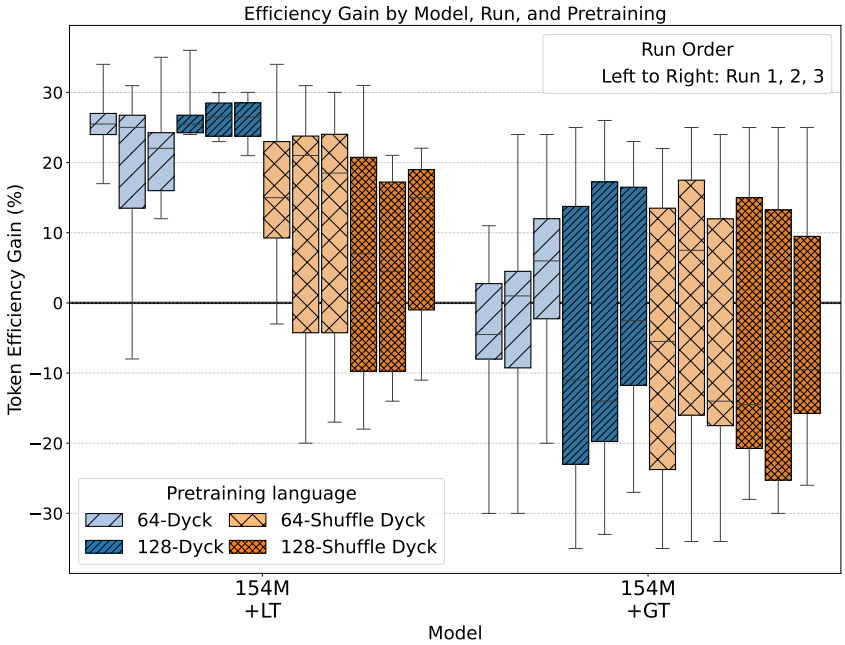

Figure 6: Efficiency gains of 154M+{LT,GT} across three runs, all pretraining and natural languages

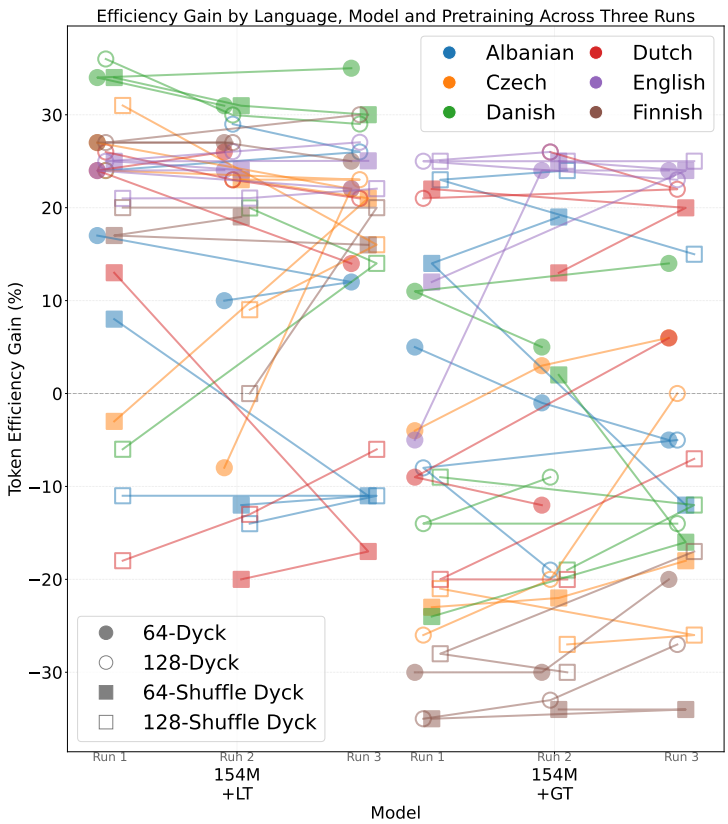

Figure 7: Efficiency gains of 154M+{LT,GT} across three runs, all pretraining and natural languages (with language-specific details)

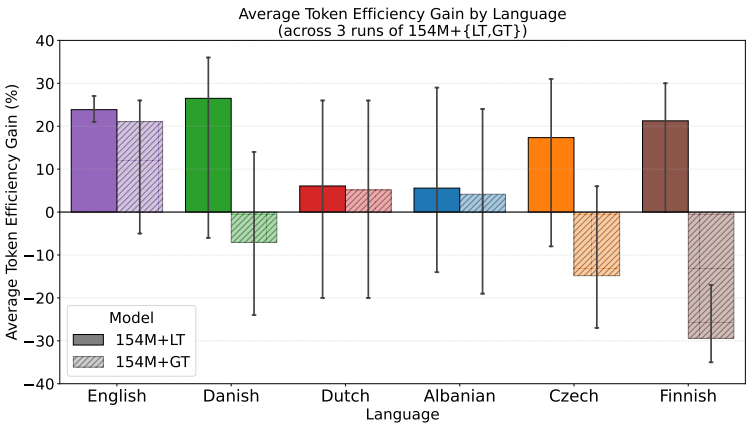

Figure 8: Average token efficiency gains of `154M+{LT,GT}` models across all pretraining languages grouped by natural language

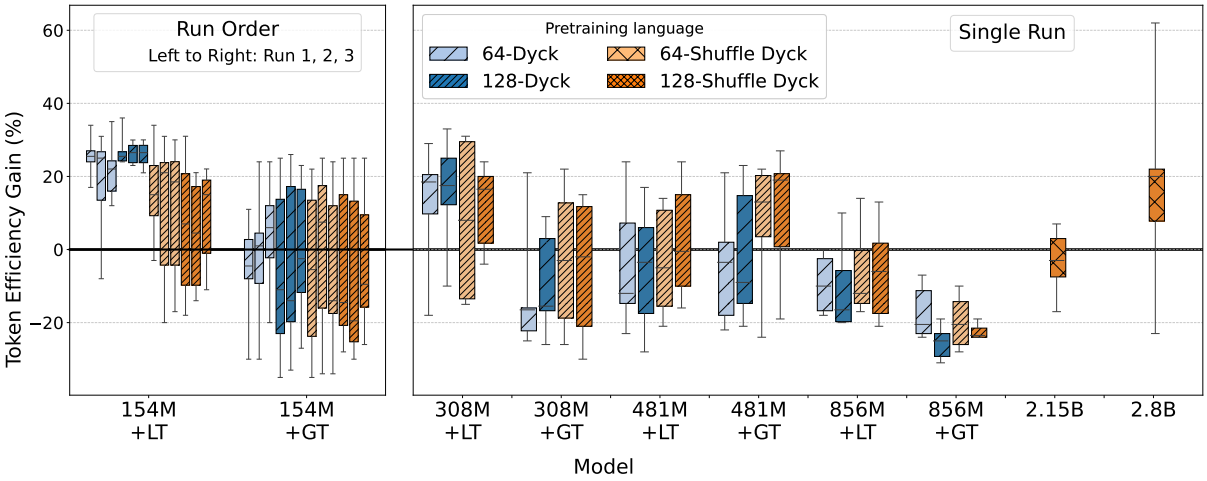

Figure 9: Efficiency gain across all models, pretraining, and natural languages. Except for the `154M` models (left panel), each training setup was run only once.

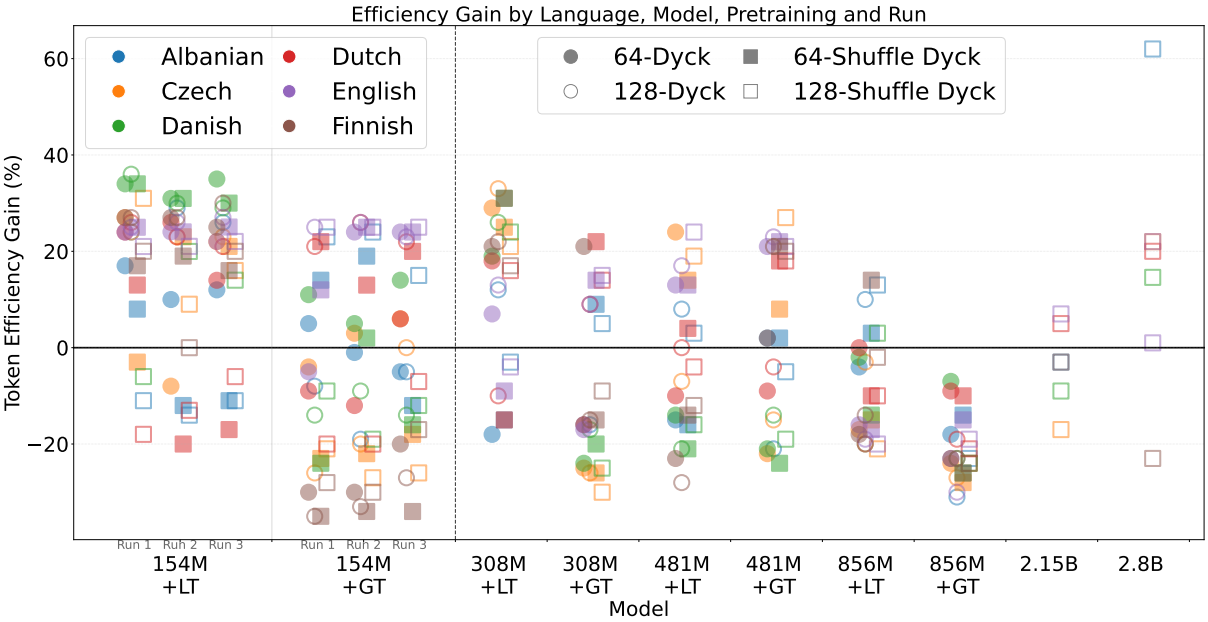

Figure 10: Efficiency gain across all models, pretraining, and natural languages (with language-specific details)

Table 5: Token efficiency gain across experiments

Model	Language	Gain % ↑
154M+GT_128-dyck_run1	Finnish	-35.005
154M+GT_64-shuff-dyck_run1	Finnish	-35.005
154M+GT_64-shuff-dyck_run3	Finnish	-34.009
154M+GT_64-shuff-dyck_run2	Finnish	-34.004
154M+GT_128-dyck_run2	Finnish	-33.002
856M+GT_128-dyck	Albanian	-31.002
154M+GT_64-dyck_run1	Finnish	-30.010
154M+GT_64-dyck_run2	Finnish	-30.007
154M+GT_128-shuff-dyck_run2	Finnish	-30.007
308M+GT_128-shuff-dyck	Czech	-30.004
856M+GT_128-dyck	English	-29.999
481M+LT_128-dyck	Finnish	-28.010
154M+GT_128-shuff-dyck_run1	Finnish	-28.008
856M+GT_64-shuff-dyck	Czech	-28.003
154M+GT_128-dyck_run3	Finnish	-27.009
154M+GT_128-shuff-dyck_run2	Czech	-27.004
856M+GT_128-dyck	Czech	-27.002
856M+GT_64-shuff-dyck	Finnish	-26.008
154M+GT_128-dyck_run1	Czech	-26.001
308M+GT_128-dyck	Czech	-26.001
308M+GT_64-shuff-dyck	Czech	-26.001
856M+GT_64-shuff-dyck	Danish	-26.000
154M+GT_128-shuff-dyck_run3	Czech	-25.999
308M+GT_64-dyck	Czech	-25.001
308M+GT_128-shuff-dyck	Danish	-25.000
856M+GT_128-shuff-dyck	Finnish	-24.008
856M+GT_64-dyck	Czech	-24.002
856M+GT_128-shuff-dyck	Czech	-24.002
154M+GT_64-shuff-dyck_run1	Danish	-24.000
308M+GT_64-dyck	Danish	-24.000
856M+GT_128-shuff-dyck	Danish	-24.000
481M+GT_64-shuff-dyck	Danish	-24.000
856M+GT_64-dyck	Finnish	-23.007
856M+GT_128-dyck	Finnish	-23.007
481M+LT_64-dyck	Finnish	-23.004
856M+GT_128-shuff-dyck	Albanian	-23.002
154M+GT_64-shuff-dyck_run1	Czech	-23.001
856M+GT_128-dyck	Danish	-23.000
856M+GT_64-dyck	English	-23.000
2.8B_128-shuff-dyck	Finnish	-22.995
154M+GT_64-shuff-dyck_run2	Czech	-22.004
481M+GT_64-dyck	Czech	-22.002
856M+LT_128-shuff-dyck	Czech	-21.005
856M+GT_128-shuff-dyck	Dutch	-21.004
481M+GT_128-dyck	Albanian	-21.003
154M+GT_128-shuff-dyck_run1	Czech	-21.001
481M+GT_64-dyck	Danish	-21.001
481M+LT_128-dyck	Danish	-20.998
481M+LT_64-shuff-dyck	Danish	-20.998
154M+GT_128-shuff-dyck_run2	Dutch	-20.015
154M+GT_64-dyck_run3	Finnish	-20.007
154M+GT_128-shuff-dyck_run1	Dutch	-20.004
154M+GT_128-dyck_run2	Czech	-20.003
154M+LT_64-shuff-dyck_run2	Dutch	-20.002
856M+LT_128-shuff-dyck	English	-20.002
308M+GT_64-shuff-dyck	Danish	-20.001
856M+LT_128-dyck	Dutch	-19.997
856M+LT_128-dyck	Danish	-19.994
154M+GT_128-dyck_run2	Albanian	-19.004
856M+LT_128-dyck	English	-19.003
856M+GT_128-dyck	Dutch	-19.003
154M+GT_128-shuff-dyck_run2	Danish	-19.001
481M+GT_128-shuff-dyck	Danish	-19.000
856M+GT_128-shuff-dyck	English	-19.000
856M+LT_64-dyck	Finnish	-18.006
856M+GT_64-dyck	Albanian	-18.002
154M+GT_64-shuff-dyck_run3	Czech	-18.000
308M+LT_64-dyck	Albanian	-18.000
154M+LT_128-shuff-dyck_run1	Dutch	-17.993
154M+GT_128-shuff-dyck_run3	Finnish	-17.005
2.15B_128-shuff-dyck	Czech	-17.003
856M+LT_64-dyck	Czech	-17.003
154M+LT_64-shuff-dyck_run3	Dutch	-17.002
308M+GT_128-dyck	Danish	-17.001
308M+GT_64-dyck	English	-17.000
856M+LT_64-shuff-dyck	English	-17.000
856M+LT_64-dyck	English	-16.004
154M+GT_64-shuff-dyck_run3	Danish	-16.004
308M+GT_64-dyck	Albanian	-16.001
308M+GT_128-dyck	Albanian	-16.001
481M+LT_64-shuff-dyck	Albanian	-16.001
481M+LT_128-shuff-dyck	Danish	-15.993
308M+GT_64-dyck	Dutch	-15.993
308M+GT_64-shuff-dyck	Finnish	-15.002
308M+GT_128-dyck	Finnish	-15.002
481M+GT_128-dyck	Czech	-15.001
481M+LT_64-dyck	Albanian	-15.001
308M+LT_64-shuff-dyck	Albanian	-15.001
856M+GT_64-shuff-dyck	English	-14.999
856M+LT_64-shuff-dyck	Czech	-14.997
308M+LT_64-shuff-dyck	Dutch	-14.997
154M+LT_128-shuff-dyck_run2	Albanian	-14.008
154M+GT_128-dyck_run3	Danish	-14.004
481M+LT_64-shuff-dyck	Finnish	-14.004
856M+LT_128-dyck	Finnish	-14.004
856M+GT_64-shuff-dyck	Albanian	-14.002
481M+GT_128-dyck	Danish	-14.002
154M+GT_128-dyck_run1	Danish	-14.002
856M+LT_64-shuff-dyck	Danish	-13.997
481M+LT_64-dyck	Danish	-13.997
154M+LT_128-shuff-dyck_run2	Dutch	-13.005
154M+GT_64-dyck_run2	Dutch	-12.015
154M+GT_64-shuff-dyck_run3	Albanian	-12.012
154M+LT_64-shuff-dyck_run2	Albanian	-12.009
481M+LT_128-shuff-dyck	Finnish	-12.006
154M+GT_128-shuff-dyck_run3	Danish	-12.003
154M+LT_64-shuff-dyck_run3	Albanian	-11.006
154M+LT_128-shuff-dyck_run3	Albanian	-11.006
154M+LT_128-shuff-dyck_run1	Albanian	-11.003
856M+LT_64-shuff-dyck	Dutch	-10.003
856M+LT_128-shuff-dyck	Dutch	-10.003
308M+LT_128-dyck	Dutch	-10.003
481M+LT_64-dyck	Dutch	-10.003
856M+GT_64-shuff-dyck	Dutch	-10.002
154M+GT_64-dyck_run1	Dutch	-9.002
481M+GT_64-dyck	Dutch	-9.002
856M+GT_64-dyck	Dutch	-9.002
154M+GT_128-dyck_run2	Danish	-9.001
308M+GT_128-shuff-dyck	Finnish	-9.000
154M+GT_128-shuff-dyck_run1	Danish	-9.000

Model	Language	Gain % ↑
2.15B_128-shuff-dyck	Danish	-8.998
308M+LT_64-shuff-dyck	English	-8.995
154M+GT_128-dyck_run1	Albanian	-8.001
154M+LT_64-dyck_run2	Czech	-7.989
154M+GT_128-shuff-dyck_run3	Dutch	-7.027
856M+GT_64-dyck	Danish	-6.999
481M+LT_128-dyck	Czech	-6.998
154M+LT_128-shuff-dyck_run3	Dutch	-6.008
154M+LT_128-shuff-dyck_run1	Danish	-6.000
154M+GT_128-dyck_run3	Albanian	-5.009
154M+GT_64-dyck_run3	Albanian	-5.009
154M+GT_64-dyck_run1	English	-5.001
481M+GT_128-shuff-dyck	Albanian	-5.000
856M+LT_64-dyck	Albanian	-4.003
308M+LT_128-shuff-dyck	English	-4.002
154M+GT_64-dyck_run1	Czech	-4.001
481M+LT_128-shuff-dyck	Dutch	-4.000
481M+GT_128-dyck	Dutch	-4.000
308M+LT_128-shuff-dyck	Albanian	-3.003
2.15B_128-shuff-dyck	Albanian	-3.000
154M+LT_64-shuff-dyck_run1	Czech	-2.999
856M+LT_128-dyck	Czech	-2.999
2.15B_128-shuff-dyck	Finnish	-2.998
856M+LT_128-shuff-dyck	Finnish	-2.001
856M+LT_64-dyck	Danish	-2.000
154M+GT_64-dyck_run2	Albanian	-1.002
154M+LT_128-shuff-dyck_run2	Finnish	-0.004
856M+LT_64-dyck	Dutch	0.000
481M+LT_128-dyck	Dutch	0.000
154M+GT_128-dyck_run3	Czech	0.002
2.8B_128-shuff-dyck	English	0.997
481M+GT_64-dyck	Finnish	1.997
481M+GT_64-dyck	Albanian	2.000
481M+GT_64-shuff-dyck	Albanian	2.000
154M+GT_64-shuff-dyck_run2	Danish	2.001
154M+GT_64-dyck_run2	Czech	2.996
856M+LT_128-shuff-dyck	Danish	3.002
856M+LT_64-shuff-dyck	Albanian	3.003
481M+LT_128-shuff-dyck	Albanian	3.003
481M+LT_64-shuff-dyck	Dutch	4.000
2.15B_128-shuff-dyck	Dutch	4.996
308M+GT_128-shuff-dyck	Albanian	5.000
154M+GT_64-dyck_run1	Albanian	5.000
154M+GT_64-dyck_run2	Danish	5.002
154M+GT_64-dyck_run3	Dutch	5.972
154M+GT_64-dyck_run3	Czech	6.002
308M+LT_64-dyck	English	6.999
2.15B_128-shuff-dyck	English	7.001
481M+GT_64-shuff-dyck	Czech	8.002
154M+LT_64-shuff-dyck_run1	Albanian	8.004
481M+LT_128-dyck	Albanian	8.004
308M+GT_128-dyck	English	9.000
308M+GT_64-shuff-dyck	Albanian	9.001
308M+GT_128-dyck	Dutch	9.002
154M+LT_128-shuff-dyck_run2	Czech	9.019
154M+LT_64-dyck_run2	Albanian	9.993
856M+LT_128-dyck	Albanian	10.002
154M+GT_64-dyck_run1	Danish	11.001
154M+GT_64-shuff-dyck_run1	English	12.000
154M+LT_64-dyck_run3	Albanian	12.001
308M+LT_128-dyck	Albanian	12.004

Model	Language	Gain % ↑
154M+GT_64-shuff-dyck_run2	Dutch	12.983
154M+LT_64-shuff-dyck_run1	Dutch	12.999
481M+LT_64-dyck	English	13.001
308M+LT_128-dyck	English	13.001
481M+LT_64-shuff-dyck	English	13.001
856M+LT_128-shuff-dyck	Albanian	13.003
154M+LT_64-dyck_run3	Dutch	13.989
308M+GT_128-shuff-dyck	Dutch	13.997
481M+LT_64-shuff-dyck	Czech	13.999
308M+GT_64-shuff-dyck	English	14.000
154M+GT_64-dyck_run3	Danish	14.002
154M+GT_64-shuff-dyck_run1	Albanian	14.002
856M+LT_64-shuff-dyck	Finnish	14.004
154M+LT_128-shuff-dyck_run3	Danish	14.007
2.8B_128-shuff-dyck	Danish	14.593
154M+GT_128-shuff-dyck_run3	Albanian	14.991
308M+GT_128-shuff-dyck	English	14.999
308M+LT_128-shuff-dyck	Dutch	15.997
154M+LT_64-shuff-dyck_run3	Finnish	15.998
154M+LT_128-shuff-dyck_run3	Czech	16.015
154M+LT_64-dyck_run1	Albanian	17.000
481M+LT_128-dyck	English	17.000
154M+LT_64-shuff-dyck_run1	Finnish	17.005
308M+LT_128-shuff-dyck	Finnish	17.005
308M+LT_64-dyck	Dutch	17.993
481M+GT_64-shuff-dyck	Dutch	17.999
481M+GT_128-shuff-dyck	Dutch	17.999
308M+LT_64-dyck	Danish	18.992
154M+GT_64-shuff-dyck_run2	Albanian	18.999
481M+LT_128-shuff-dyck	Czech	19.005
154M+LT_64-shuff-dyck_run2	Finnish	19.008
154M+LT_128-shuff-dyck_run2	Danish	19.970
154M+GT_64-shuff-dyck_run3	Dutch	19.984
154M+LT_128-shuff-dyck_run3	Finnish	20.001
481M+GT_128-shuff-dyck	Finnish	20.004
2.8B_128-shuff-dyck	Dutch	20.005
154M+LT_128-shuff-dyck_run1	Finnish	20.010
154M+LT_128-dyck_run3	Dutch	20.993
481M+GT_64-dyck	English	21.000
481M+GT_128-shuff-dyck	English	21.000
154M+LT_128-shuff-dyck_run1	English	21.001
154M+GT_128-dyck_run1	Dutch	21.004
481M+GT_128-dyck	Finnish	21.005
481M+GT_64-shuff-dyck	Finnish	21.005
308M+GT_64-dyck	Finnish	21.005
308M+LT_128-shuff-dyck	Czech	21.005
308M+LT_64-dyck	Finnish	21.006
154M+LT_64-shuff-dyck_run3	Czech	21.016
154M+LT_128-shuff-dyck_run2	English	21.033
154M+GT_128-dyck_run3	Dutch	21.983
2.8B_128-shuff-dyck	Czech	21.997
481M+GT_64-shuff-dyck	English	22.000
308M+LT_128-dyck	Finnish	22.005
154M+GT_64-shuff-dyck_run1	Dutch	22.006
308M+GT_64-shuff-dyck	Dutch	22.006
154M+LT_64-dyck_run3	Czech	22.015
2.8B_128-shuff-dyck	English	22.017
154M+LT_64-dyck_run3	English	22.048
154M+LT_128-shuff-dyck_run3	English	22.048
154M+LT_128-dyck_run2	Dutch	22.995
481M+GT_128-dyck	English	23.000

Model	Language	Gain % ↑
154M+GT_128-shuff-dyck_run1	Albanian	23.002
154M+GT_128-dyck_run3	English	23.005
154M+LT_64-shuff-dyck_run2	Czech	23.015
154M+LT_128-dyck_run2	Czech	23.015
154M+LT_128-dyck_run3	Czech	23.016
154M+LT_64-dyck_run1	English	23.995
481M+LT_128-shuff-dyck	English	23.995
154M+LT_64-dyck_run1	Dutch	23.995
154M+GT_128-shuff-dyck_run2	Albanian	23.997
154M+LT_128-dyck_run1	Albanian	23.997
308M+LT_128-shuff-dyck	Danish	23.999
481M+LT_64-dyck	Czech	24.000
154M+LT_128-dyck_run1	Czech	24.000
154M+GT_64-dyck_run2	English	24.003
154M+GT_64-dyck_run3	English	24.004
154M+GT_64-shuff-dyck_run3	English	24.004
154M+LT_64-shuff-dyck_run2	English	24.036
154M+LT_64-dyck_run2	English	24.036
154M+LT_64-shuff-dyck_run1	English	24.992
154M+LT_128-dyck_run1	English	24.992
154M+GT_128-dyck_run1	English	24.999
154M+GT_128-shuff-dyck_run1	English	24.999
308M+LT_64-shuff-dyck	Czech	25.001
154M+LT_64-dyck_run3	Finnish	25.001
154M+GT_128-shuff-dyck_run2	English	25.002
154M+GT_64-shuff-dyck_run2	English	25.002
154M+GT_128-shuff-dyck_run3	English	25.005
154M+LT_64-shuff-dyck_run3	English	25.042
154M+GT_128-dyck_run2	Dutch	25.990
154M+LT_128-dyck_run1	Dutch	25.996
154M+LT_64-dyck_run2	Dutch	25.997
154M+LT_128-dyck_run3	Albanian	26.000
154M+GT_128-dyck_run2	English	26.002
308M+LT_128-dyck	Danish	26.003
154M+LT_128-dyck_run2	English	26.035
154M+LT_64-dyck_run1	Czech	27.001
481M+GT_128-shuff-dyck	Czech	27.002
154M+LT_128-dyck_run2	Finnish	27.007
154M+LT_64-dyck_run2	Finnish	27.007
154M+LT_64-dyck_run1	Finnish	27.008
154M+LT_128-dyck_run1	Finnish	27.008
154M+LT_128-dyck_run3	English	27.040
154M+LT_128-dyck_run2	Albanian	28.990
308M+LT_64-dyck	Czech	29.001
154M+LT_128-dyck_run3	Danish	29.005
154M+LT_128-dyck_run2	Danish	29.976
154M+LT_128-dyck_run3	Finnish	30.003
154M+LT_64-shuff-dyck_run3	Danish	30.006
154M+LT_64-shuff-dyck_run2	Danish	30.975
154M+LT_64-dyck_run2	Danish	30.975
154M+LT_128-shuff-dyck_run1	Czech	30.999
308M+LT_64-shuff-dyck	Danish	31.005
308M+LT_64-shuff-dyck	Finnish	31.006
308M+LT_128-dyck	Czech	33.000
154M+LT_64-dyck_run1	Danish	34.006
154M+LT_64-shuff-dyck_run1	Danish	34.006
154M+LT_64-dyck_run3	Danish	35.009
154M+LT_128-dyck_run1	Danish	36.004
2.8B_128-shuff-dyck	Albanian	61.997

C Description of Linguistic Features

Below is a detailed list of linguistic features we analyze. Most linguistic features are considered in two variants: (1) punctuation is taken into account when processing the data (`*_with_punct`); and (2) punctuation is ignored (`*_without_punct`).

① **Percentage of crossing dependencies:** ratio of crossing dependent pairs to all dependent pairs.
(`%crossing_dep_with_punct`; `%crossing_dep_without_punct`)

② **Mean dependency distance:** average distance between the head and the child in the dependent pair.
(`mean_dep_dist_with_punct`; `mean_dep_dist_without_punct`)

③ **Maximum dependency distance:** maximum distance between the head and the child in the dependent pair.
(`max_dep_dist_with_punct`; `max_dep_dist_without_punct`)

④ **Mean dependency tree depth:** average depth of the dependency tree.
(`mean_tree_depth_with_punct`; `mean_tree_depth_without_punct`)

⑤ **Maximum dependency tree depth:** maximum depth of the dependency tree.
(`max_tree_depth_with_punct`; `max_tree_depth_without_punct`)

⑥ **Vocabulary size:** the number of unique tokens.
(`vocab_size_with_punct`; `vocab_size_without_punct`)

⑦ **Average sentence length:** average number of tokens in the sentence.
(`avg_sentence_length_with_punct`; `avg_sentence_length_without_punct`)

⑧ **Average number of children per head:** average number of children per head across all dependency pairs.
(`mean_children_with_punct`; `mean_children_without_punct`)

⑨ **Maximum number of children per head:** maximum number of children per head across all dependency pairs.
(`max_children_with_punct`; `max_children_without_punct`)

⑩ **Morphological richness:** the number of distinct word forms seen in a parallel corpus relative to English (Bojar, 2015).
(`morph_richness`)

⑪ **Average perplexity:** the average of perplexities measured on batches of 2,048 tokens.
(`avg_perplexity`)

⑫ **Variance of the perplexity:** the variance of perplexities measured on batches of 2,048 tokens.
(`var_perplexity`)

D Details of Language Specific Analysis

Language	Training		Evaluation	
	Avg.	Var.	Avg.	Var.
Finnish	5.53	3.71	9.15	150.52
Czech	6.40	4.73	8.67	210.16
Dutch	8.30	6.01	11.08	279.27
Albanian	6.93	13.19	9.81	474.59
English	9.08	11.55	18.37	797.93
Danish	7.40	7.40	14.11	3776.23

Table 6: Perplexity of the datasets

Figures 11 and 12 provide the average correlations between efficiency gain across experiments and linguistic features (see Appendix C) estimated on evaluation and training data, respectively.

Figures 13 and 14 provide the average correlations between efficiency gain across models with `LT` and linguistic features (see Appendix C) estimated on evaluation and training data, respectively.

Figures 15 and 16 provide the average correlations between efficiency gain across models with `GT` and linguistic features (see Appendix C) estimated on evaluation and training data, respectively.

Figures 17 and 18 provide the average correlations between efficiency gain across models pretrained on {64,128}-Shuffle Dyck and linguistic features (see Appendix C) estimated on evaluation and training data, respectively.

Figures 19 and 20 provide the correlations between efficiency gain across experiments and linguistic features (see Appendix C) estimated on evaluation and training data, respectively.

Linguistic features are sorted from top to bottom and from left to right (i.e., higher variance at the bottom right side) for Figures 11 to 18, and from left to right (i.e., higher variance at the right side) for Figures 19 and 20. They are sorted based on their reliability which is quantified by the variance of the feature value across selected natural languages.

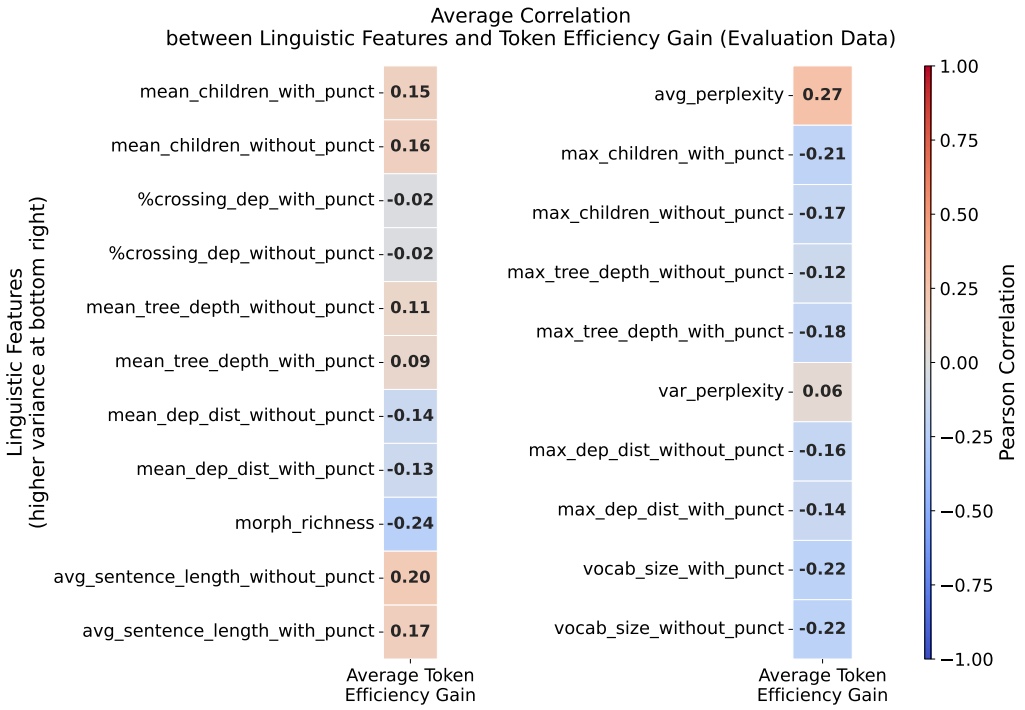

Figure 11: Average correlation between linguistic features and token efficiency gain (evaluation data)

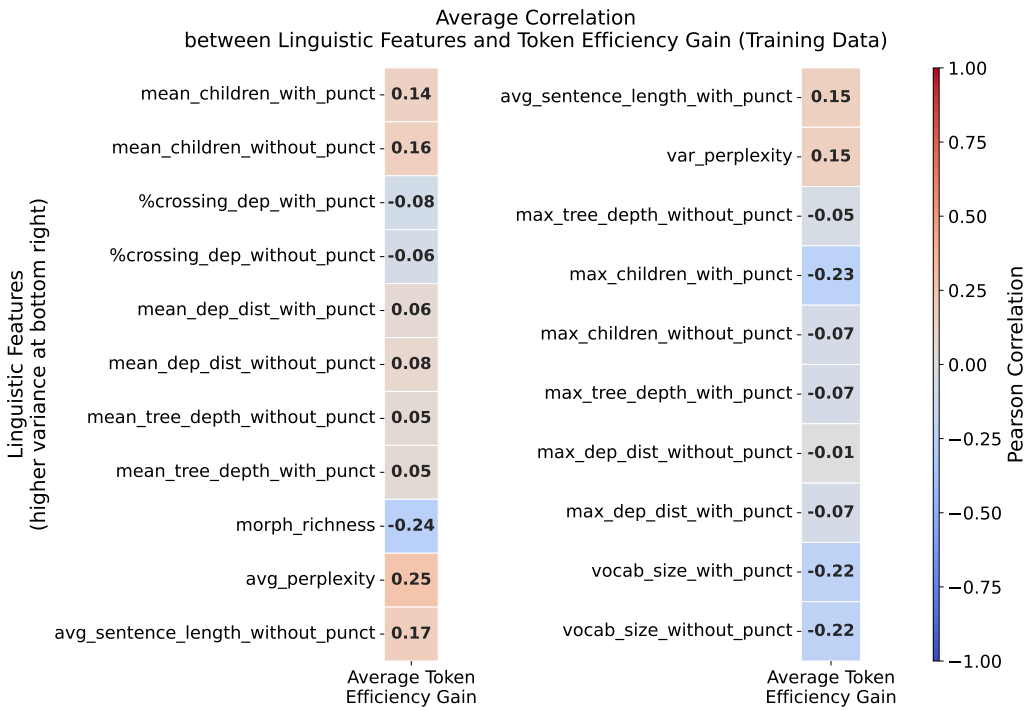

Figure 12: Average correlation between linguistic features and token efficiency gain (training data)

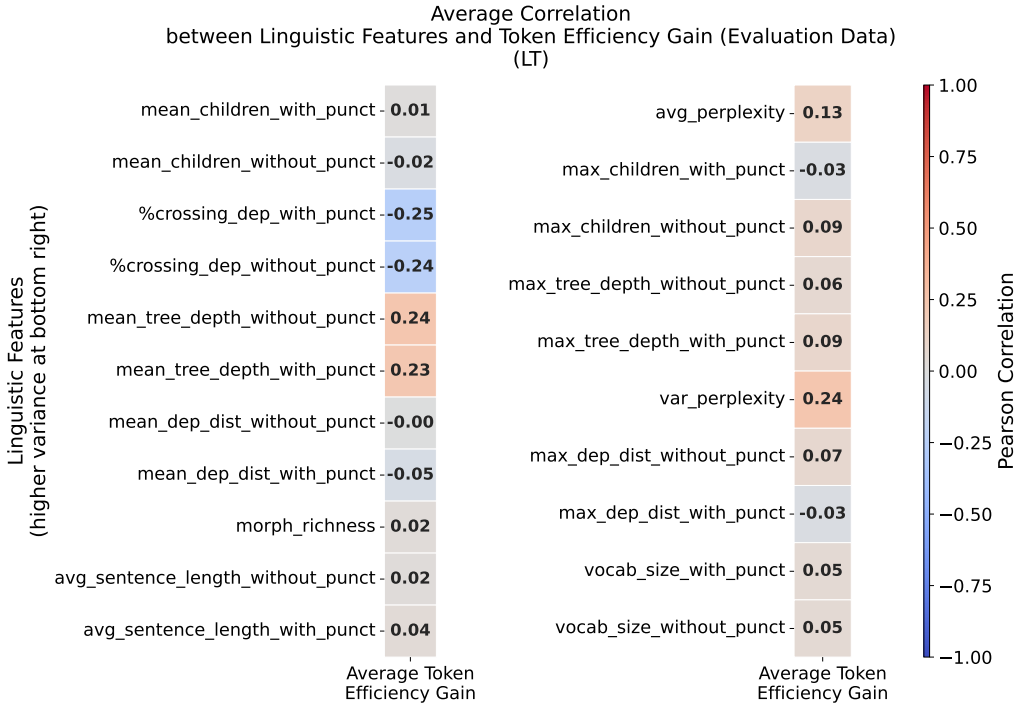

Figure 13: Average correlation between linguistic features of evaluation data and token efficiency gain of models with LT

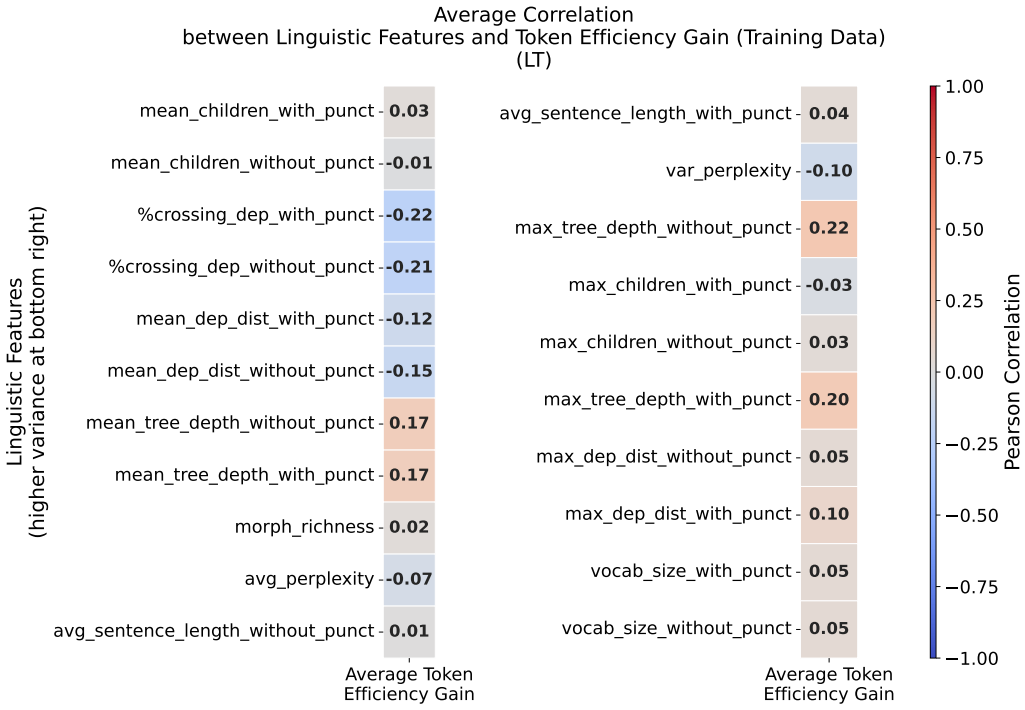

Figure 14: Average correlation between linguistic features of training data and token efficiency gain of models with LT

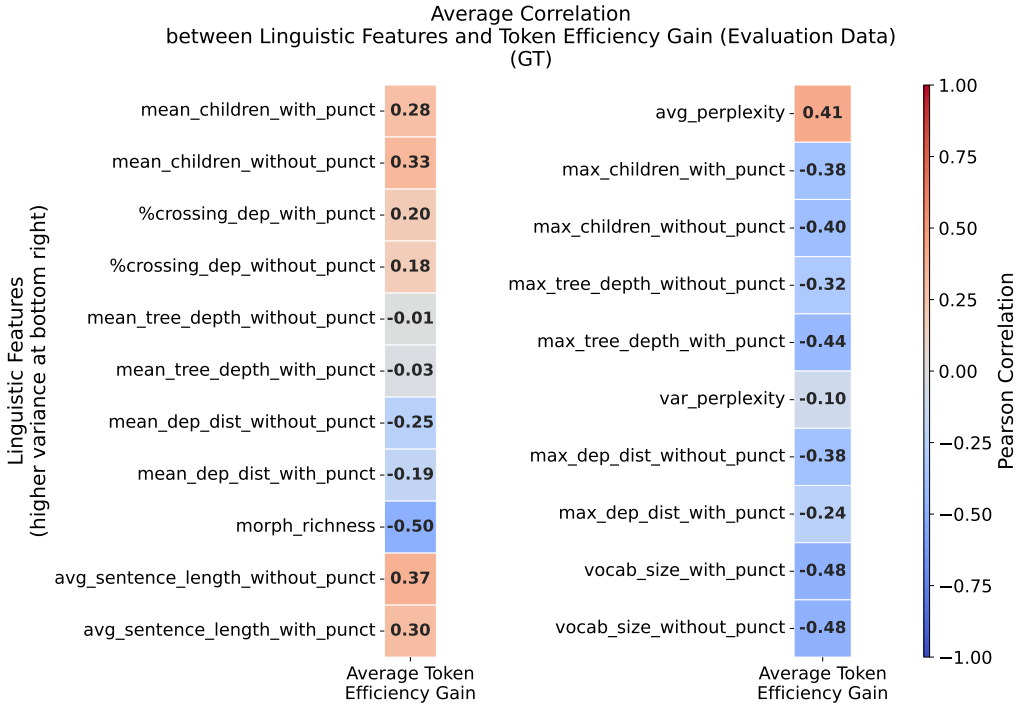

Figure 15: Average correlation between linguistic features of evaluation data and token efficiency gain of models with GT

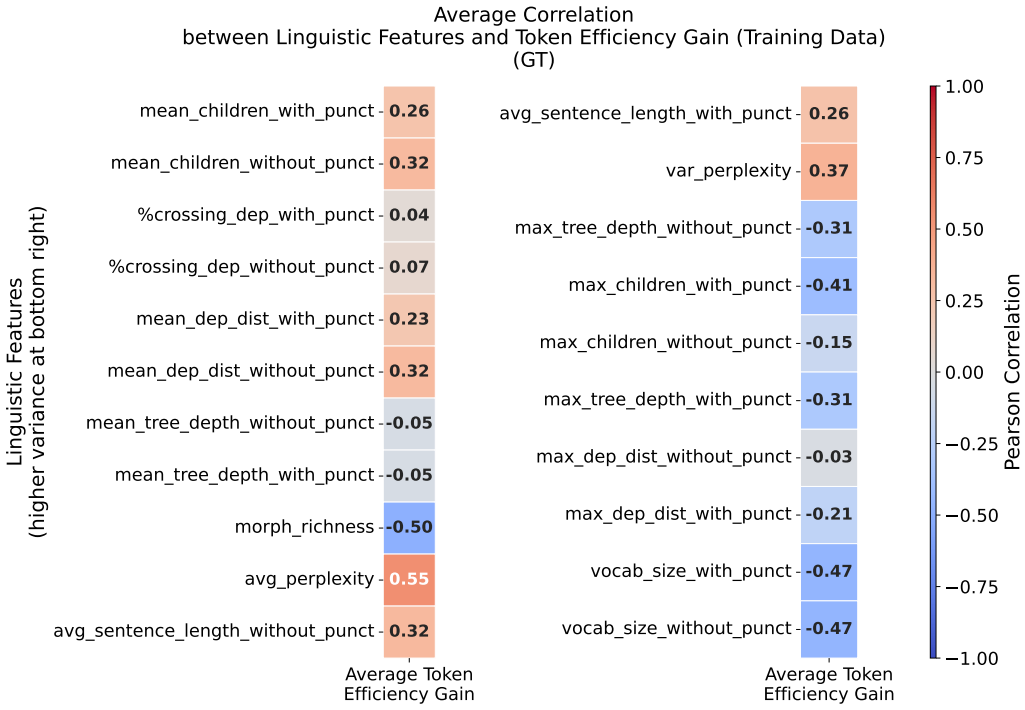

Figure 16: Average correlation between linguistic features of training data and token efficiency gain of models with GT

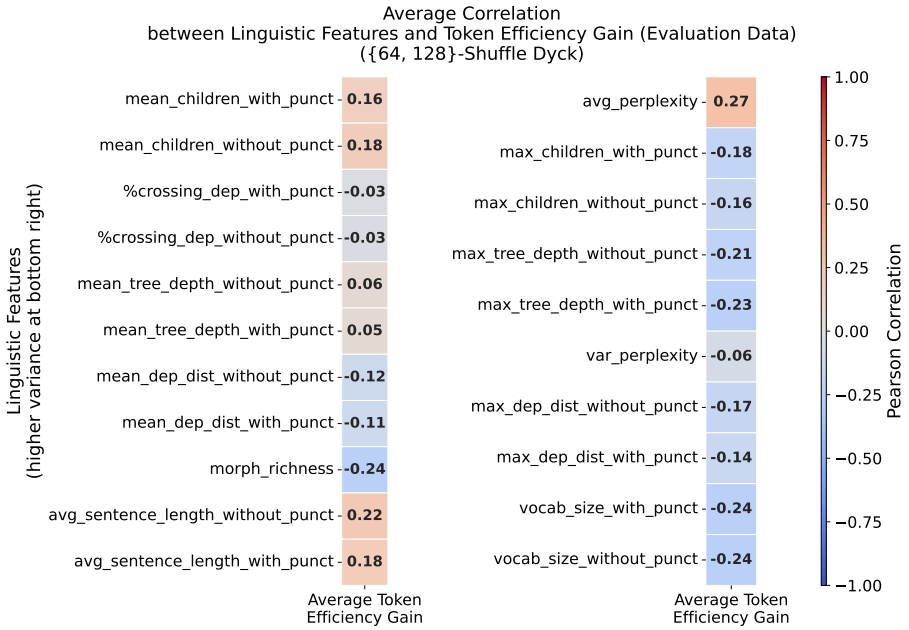

Figure 17: Average correlation between linguistic features of evaluation data and token efficiency gain of models pretrained on {64,128}-Shuffle Dyck

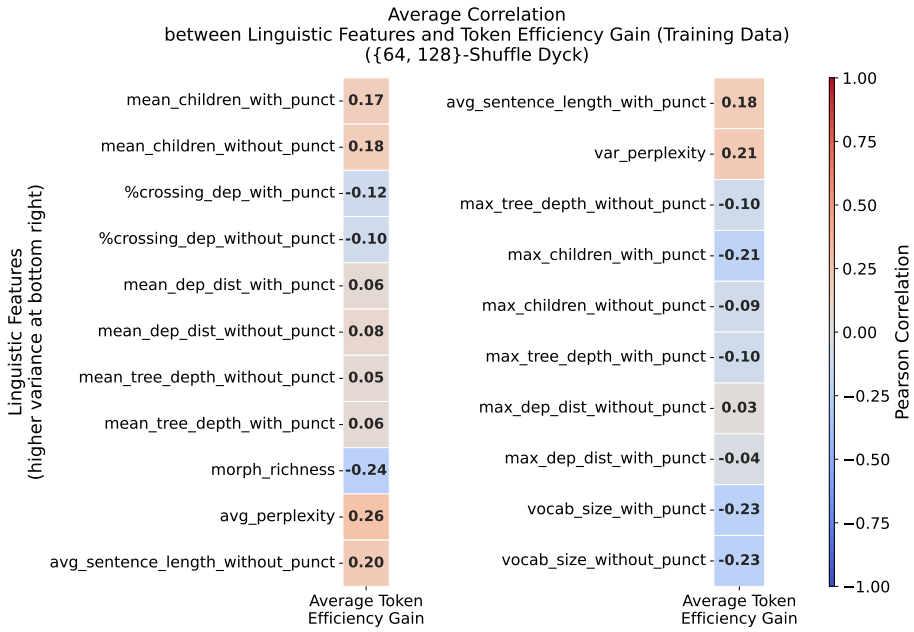

Figure 18: Average correlation between linguistic features of training data and token efficiency gain of models on {64,128}-Shuffle Dyck

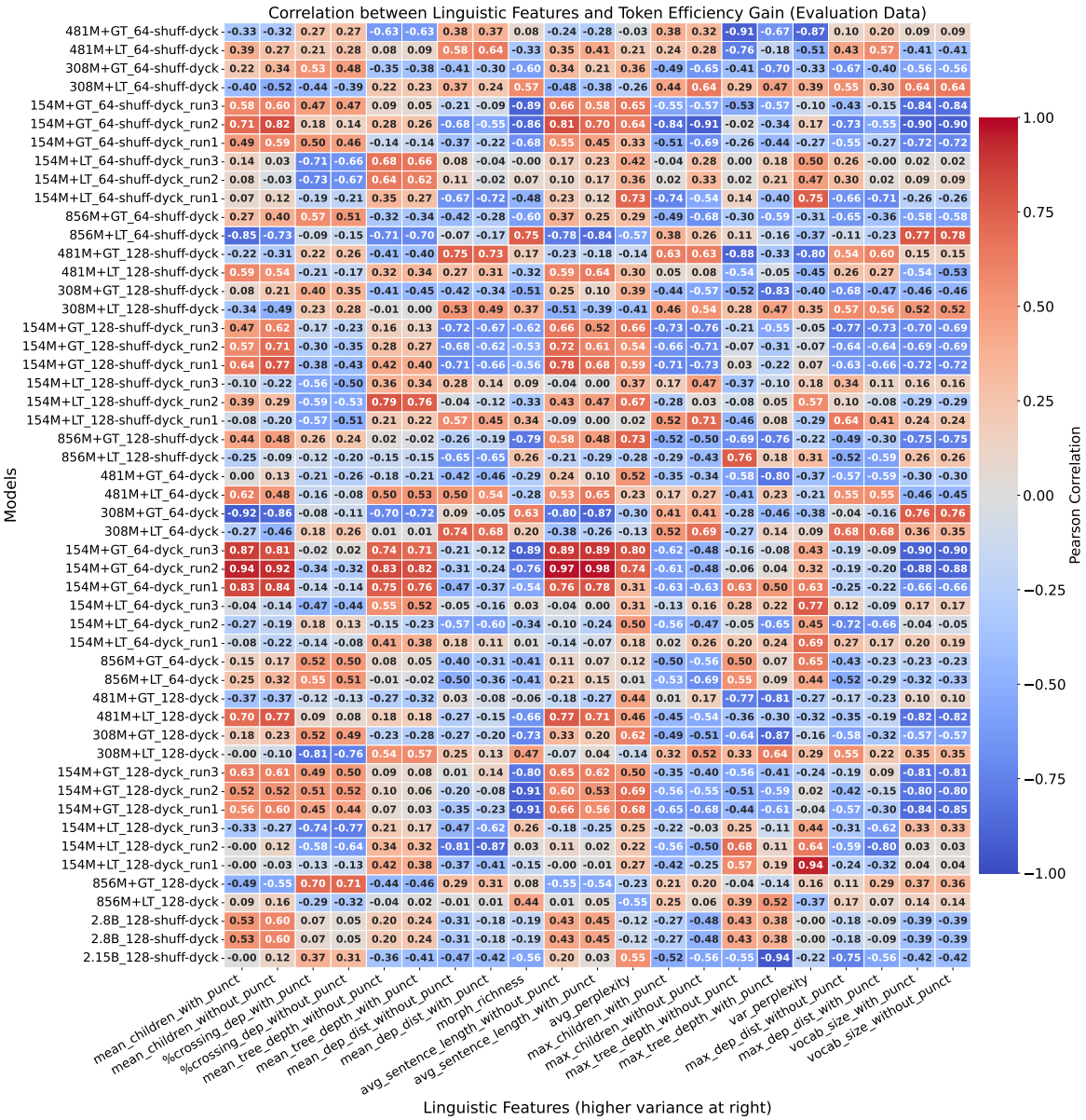

Figure 19: Correlation between linguistic features and token efficiency gain (evaluation data)

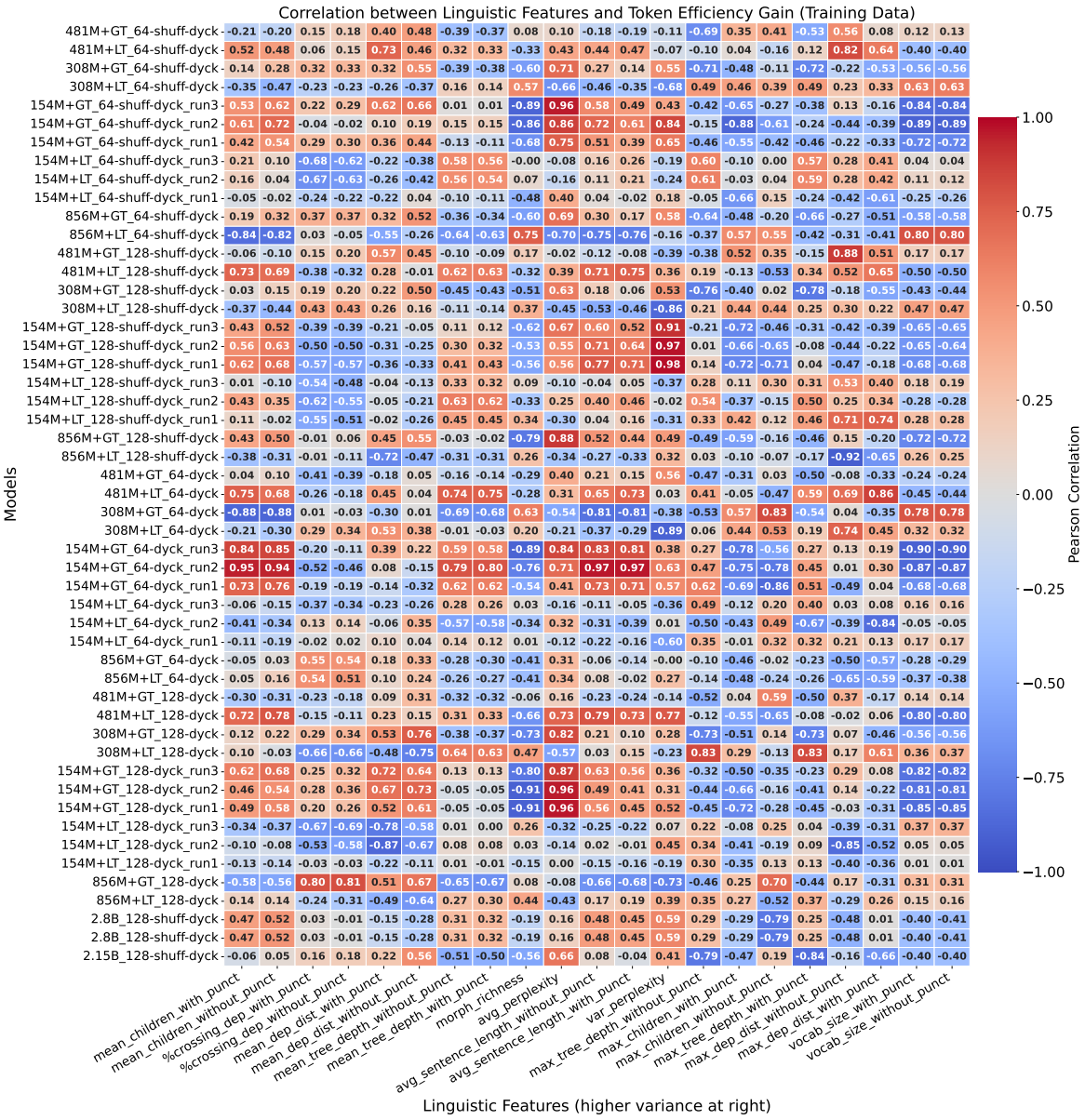

Figure 20: Correlation between linguistic features and token efficiency gain (training data)