
Which Tokens Can a Device Afford to Score? Vocabulary Pruning by Log-Mass Maximization

Egor Krasheninnikov
Amazon Web Services
egorkr@amazon.com

Sergey Ivanov
Amazon Web Services
ssivanov@amazon.com

Nikita Kozodoi
Amazon Web Services
kozodoi@amazon.com

Natalie Tarasova
Amazon Web Services
natltar@amazon.com

Aoyu Zhang
Amazon Web Services
aoyuzhan@amazon.com

Abstract

A language model on an edge device runs under tight compute and memory constraints, and its output vocabulary incurs two distinct costs. The embedding matrix can hold up to a third of the weights at sub-billion parameter scale. On top of that, every decoding step has to score the full vocabulary. Restricting inference to K tokens shrinks that layer by up to $256\times$ in memory and $92\times$ in the time it contributes per decoded token. However, which K tokens to keep is an open question. The standard practice keeps the most frequent tokens, which optimizes average coverage and can strip the tokens minority languages depend on. We instead choose the tokens by greedily maximizing an explicit objective: the expected log retained probability mass under the deployment distribution. Maximizing this objective is equivalent to minimizing reverse KL to the teacher, and it is monotone submodular, which is what lets a standard greedy algorithm inherit a $(1 - 1/e)$ guarantee. On a ten-language multilingual dataset, this recovers the quality that pruning otherwise takes away from minority scripts: Arabic reverse KL falls by 0.53 nats and Russian generation gains +1.4 chrF, at negligible cost to English. The same selection of tokens decodes faster, raising speculative-decoding acceptance by +5.4pp on Arabic and measured throughput by up to 6.1%, and it shrinks memory further when combined with quantization, reaching a $3.5\times$ reduction while keeping 95% of full-precision quality. Because the tokens are chosen offline, a device pays nothing for this at inference time.

1 Introduction

A language model small enough to run on an edge device spends a substantial share of its parameters on its own vocabulary. For instance, in Qwen3.5-0.8B the 248k-token embedding holds 254M of 752M parameters, a third of the weights, and in Llama-3.2-1B the output matrix alone is 263M. The share shrinks as models grow, to 7% on Qwen2.5-7B, which is why it matters most in the sub-billion parameter range that a small edge device can host. The output vocabulary incurs two distinct costs: the matrix occupies scarce weight memory, and every decoding step has to score it again, which is why the LM head can dominate drafting time in speculative decoding [18, 17] and eat the savings from early exiting [16].

Output vocabulary pruning addresses this problem: it keeps the model and tokenizer fixed and restricts the tokens scored at inference, shrinking the output matrix from $|V|$ rows to K . Unlike input-side pruning, which retokenizes the prompt and changes the predictive process, it leaves the model’s computation intact. The savings are large as indicated in Table 1: at $K=1,000$ the output

layer shrinks by 152 to $256\times$ in memory and contributes 21 to $92\times$ less to decode time. So the open question is not whether to prune, but what tokens to keep:

If a device can score only K output tokens per step, which K tokens should it keep?

The standard answer in the literature is frequency: keep the tokens that appear most often in a calibration corpus. It is simple, interpretable, and often effective. At the same time, it cannot express the fact that some tokens are globally rare yet locally important. For example, tokens representing a Cyrillic subword, a code token such as `def`, or a biomedical term may carry modest marginal frequency while holding much of the predictive mass in the contexts where it matters. A rule tuned for average popularity can therefore fail on the long tail. On-device deployment often works with long-tail token distributions: a model shipped to one region, language, or domain sees a stream far from the balanced mixture a pretraining corpus suggests.

Our contributions are twofold. First, we give this token selection problem an explicit objective, the expected log retained probability mass under the deployment distribution, and prove two things about it. We show that maximizing it is equivalent to minimizing reverse KL between the teacher and its support-restricted renormalization, and the resulting set function is monotone submodular, so greedy selection inherits a $(1 - 1/e)$ guarantee with an exact lazy acceleration. The same analysis places frequency pruning as the linear surrogate of this objective, optimal under a *common-ranking* condition on contexts and suboptimal once it fails. Second, we validate the method on quality, decoding speed, and weight memory across six models and multiple dataset distributions, and show that the size of its advantage tracks deployment heterogeneity exactly as the theory predicts.

2 Related Work

Restricting the output vocabulary is a recurring inference optimization, most directly inside speculative decoding. FR-Spec [18] confines the drafter to a frequency-ranked subset, reporting that the LM head and softmax take 62% of drafting time on large-vocabulary targets. VocabTrim [6] targets the same reduction for memory-bound deployment, SpecVocab [17] decouples the draft vocabulary from the target’s, and Vincenti et al. [16] prune at inference to speed up confidence estimation in early-exit models. Static reduction has also compressed multilingual models by dropping tokens a target language never uses [15, 1, 3]. These are effective engineering results that share one gap: each fixes a rule for choosing tokens without saying what the rule optimizes, so there is no way to tell when it is the right rule. We supply that objective, and FR-Spec’s ranking turns out to be its linear surrogate, which is why our *frequency* baseline implements that policy.

Quantization is the other post-training axis, cutting bytes per parameter rather than parameter count. The two compose, since pruning drops embedding rows and quantization rescales those that remain (Appendix H). Magnitude saliency [7] gives our data-free baseline. The structure we exploit is classical: monotone submodular maximization under a cardinality constraint has the greedy $(1 - 1/e)$ guarantee [11, 4], and $\log(\text{linear})$ objectives of our form appear in sensor placement [8]. Our contribution is reducing output-vocabulary pruning to that family.

3 Method

Figure 1 sketches our greedy vocabulary pruning method compared to the frequency-based baseline.

Let V be the vocabulary of a pretrained model, c a context (a token prefix, or its final hidden state), and p_c the teacher’s next-token distribution at c . The deployment induces a distribution μ over contexts, and the retained support should reflect that mixture rather than any corpus-independent notion of importance. We fix a small *fallback support* B that is always retained, such as byte-level fallbacks, which keeps the objective finite, and select A from the optional tokens $U := V \setminus B$ under $|A| = K$. The *retained mass* at c is then

$$M_c(A) := \sum_{v \in B} p_c(v) + \sum_{v \in A} p_c(v). \quad (1)$$

We select the support by maximizing expected log retained mass,

$$A^* \in \operatorname{argmax}_{|A|=K} F(A), \quad F(A) := \mathbb{E}_{c \sim \mu} [\log M_c(A)]. \quad (2)$$



Figure 1: Mechanism of greedy vocabulary selection. **Left:** the full vocabulary holds a large English cluster beside smaller minority-domain clusters (Arabic, code), and the output layer dominates model memory. **Center:** greedy picks the token of highest marginal gain across calibration contexts. As a domain becomes well covered its gains diminish, so the algorithm switches to the least-covered domain, balancing coverage after K steps. **Right:** at inference only K tokens are scored.

We use the logarithm because it measures a distortion the pruned model actually suffers. Restricting the teacher to $S_A := B \cup A$ and renormalizing gives

$$q_{c,A}^*(v) = p_c(v) \mathbf{1}\{v \in S_A\} / M_c(A), \quad (3)$$

which is exactly the reverse-KL projection onto the restricted support: $q_{c,A}^*$ minimizes $D_{\text{KL}}(q \| p_c)$ over all q supported on S_A , with minimum value $-\log M_c(A)$ (Theorem 2). Maximizing F is therefore precisely minimizing expected reverse KL between the teacher and its best support-restricted form. Reverse KL is the only choice of direction available: the forward divergence $D_{\text{KL}}(p_c \| q)$ is infinite whenever q drops mass the teacher assigns, which is exactly what pruning does.

The gain from adding token u in context c is $\log(1 + p_c(u)/M_c(A))$, which is nonnegative and decreasing in $M_c(A)$. Hence F is monotone submodular, and greedy maximization satisfies $F(A_K) \geq F(\emptyset) + (1 - 1/e)(F(A^*) - F(\emptyset))$, a bound no polynomial-time algorithm can beat unless $P=NP$ (Theorems 3 and 4). Read practically, submodularity is the mechanism of Figure 1: a token earns its slot not when its mass is large outright, but when it is large *relative to how poorly its context is already covered*. Once a domain is well covered its gains shrink, so the budget moves to whichever domain is least covered. In deployment the expectation becomes a calibration average $\hat{F}(A) = \frac{1}{n} \sum_i \log M_i(A)$, whose exact marginal gain for a candidate $u \notin A$ is

$$\Delta(u | A) = \frac{1}{n} \sum_{i=1}^n \log(1 + p_i(u)/M_i(A)), \quad (4)$$

and diminishing returns permit an exact lazy-greedy acceleration (Algorithm 1).

The same objective also explains frequency pruning. Replacing the log with the linear objective $\mathbb{E}_c[M_c(A)]$ recovers it exactly, its maximizer keeping the K tokens of largest marginal predictive frequency (Theorem 1), so frequency is the first-order surrogate of our objective (Theorem 5). This pins down when it suffices: if all contexts share one token ranking frequency is *exactly* optimal (Theorem 6), but once they disagree it can be strictly suboptimal (Theorem 7), because a globally frequent generalist token can win on average mass while two specialist tokens would cover both contexts better. Frequency is therefore right in the homogeneous case, and heterogeneity is what breaks it, which is the hypothesis Section 4 tests. Proofs of all results in this section are in Appendix A.

4 Experiments

We compare *greedy* against two baselines at equal budget K and with the same fallback support, so any difference reflects *which* tokens a rule keeps, not how many. *Frequency* keeps the K most frequent tokens, FR-Spec’s policy [18] calibrated on the deployment; *magnitude* keeps the largest-norm output rows [7], using no calibration data. Our main model is Qwen3.5-0.8B-Base ($|V|=248k$) on FLORES-200 [12] across ten languages in five scripts, and we report reverse KL (nats), chrF [13] on 80-token continuations, draft acceptance, and measured wall-clock time, at $\beta = 0.01$ fallback mass throughout. A synthetic study (Figure 2, setup in Appendix D) verifies the mechanism where heterogeneity is directly controllable: the rules coincide exactly at one domain, as Theorem 6 requires, then diverge monotonically as domains are added. The full protocol and all extended results are in the appendix.

Table 1 quantifies what pruning buys on an edge device. Restricting the output layer to $K=1,000$ tokens shrinks its $|V| \times d$ matrix by 152 to $256\times$ and cuts its contribution to decode time by 21 to $92\times$. The pruned layer costs a near-constant ~ 0.045 s per thousand tokens on every model, since at this width fixed per-call overhead dominates, so the saving grows with the size of the full layer: the 256k-token output layer of Gemma-2-2B falls from 1.18 GB to 4.6 MB. Both rules realize identical reductions here, keeping the same token count on the same kernel path.

An edge target budgets gigabytes of weights rather than tokens, so pruning also has to coexist with quantization. It does, and more than additively. Qwen3.5-0.8B ties its input embedding and output head, and bitsandbytes leaves a tied embedding in BF16, so the full vocabulary imposes a 0.509 GB floor that no amount of backbone quantization can cross. Only pruning removes it. Combining 4-bit GGUF with pruning to $K=128k$ reaches 0.428 GB, a $3.5\times$ reduction from BF16, while macro chrF slips only from 11.67 to 11.12 and every language keeps at least 92% of its full-precision score. Which operation runs first does not matter, since both orders deploy byte-identical weights and differ by 0.006 nats over 672 pairs (Appendix H).

Those savings accrue to any rule at a given budget. What the rule decides is the quality retained in exchange, and Table 2 measures it over four deployments spanning low to high heterogeneity. One ordering holds everywhere: full vocab < greedy < frequency $\ll$ magnitude on reverse KL. Magnitude collapses in every cell, above 3.4 nats with chrF no better than 2.5, so a data-free rule is not competitive and the real contest is between the two calibrated ones. Between those two, the gap tracks heterogeneity as predicted. In the single-script Latin deployment, where contexts largely agree on a token ranking, greedy and frequency nearly coincide at 1.21 against 1.22 nats, the empirical signature of Theorem 6. Where rankings diverge the gap is wide: 1.55 against 1.95 nats under extreme en+ar skew, with chrF 8.9 against 6.9. Greedy never loses on either metric. The per-language view explains why (Figure 3): under English-dominant calibration frequency drives Arabic to 2.39 nats and Chinese to 2.73, and greedy recovers 0.53 and 0.47 of that at a cost of only 0.08 to English, lifting Russian generation from 3.0 to 4.4 chrF. The pattern reproduces on Llama-3.2-1B (Appendix F.3), which holds the sharpest case we found: at $K=1,000$ under high skew greedy keeps Arabic usable at 6.55 chrF while frequency starves it to broken output at 0.27, leaving English untouched.

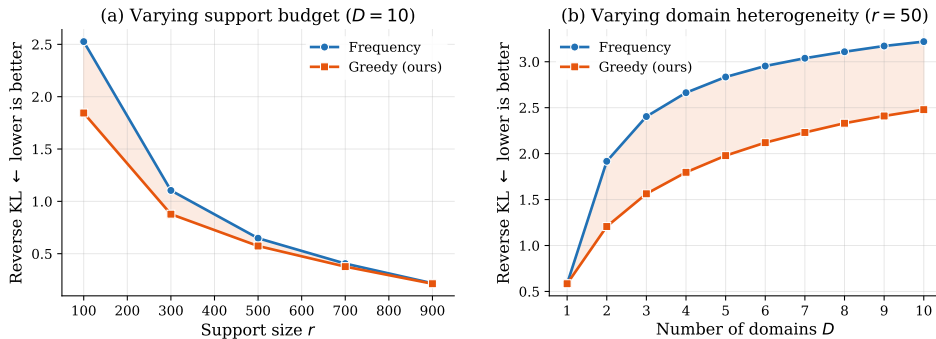


Figure 2: Synthetic multi-domain experiment, reverse KL (lower is better). (a) Budget sweep at $D=10$. (b) Heterogeneity sweep at $r=50$: the rules coincide at $D=1$ where common-ranking holds, then diverge monotonically. Shading marks the improvement.

Name	Model size			Output-layer memory (MB) ↓		Output-layer latency (s / 1K tok) ↓	
	$ V $	d	(GB)	Full	Pruned ($K=1,000$)	Full	Pruned ($K=1,000$)
Qwen2.5-0.5B	151,936	896	0.99	272	1.8 (152×)	0.97	0.045 (21×)
Qwen3.5-0.8B-Base	248,320	1,024	1.75	509	2.0 (248×)	1.84	0.045 (41×)
Qwen2.5-7B-Instruct	152,064	3,584	15.2	1090	7.2 (152×)	3.82	0.048 (80×)
Gemma-2-2B	256,000	2,304	5.22	1180	4.6 (256×)	4.30	0.046 (92×)

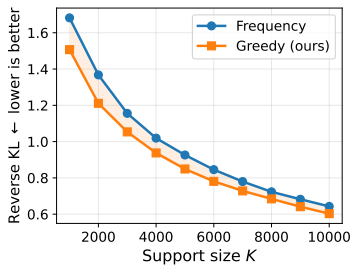
Table 1: Output-layer memory and latency when pruning to $K=1,000$ tokens across models and vocabulary sizes. Memory is the bf16 size of the $|V| \times d$ output projection, and latency is the wall-clock time it contributes over a 1,000-token generation (median per-step time, PyTorch fp16, batch of one, scaled to 1,000 tokens). The pruned layer costs ~ 0.045 s regardless of model, so savings grow with the size of the full layer. End-to-end decode speedup depends on the share of wall-clock spent in the output layer, which grows with vocabulary size and batch.

Deployment	Reverse KL ↓ (nats)				chrF ↑			
	Full	Magn.	Freq.	Greedy	Full	Magn.	Freq.	Greedy
Latin (5 langs)	0	+3.50	+1.22	+1.21	18.6	2.5	10.8	11.9
Multi-script (10 langs)	0	+3.78	+1.71	+1.53	12.1	1.5	5.9	6.6
Multi-script (5 langs)	0	+3.84	+1.32	+1.25	14.0	0.5	7.3	7.7
High skew (en+ar, 0.99)	0	+3.86	+1.95	+1.55	13.9	1.2	6.9	8.9

Table 2: Four deployments on Qwen3.5-0.8B-Base at $K=1,000$, ordered by calibration heterogeneity. Bold marks the best *pruned* method. The greedy-frequency gap is near zero where a common token ranking nearly holds and largest under extreme skew, as Theorems 6 and 7 predict.

Does better fidelity translate into faster decoding? Reverse KL of the retained set is the negative log of the expected accept probability under Leviathan-Chen sampling [9, 2], so the theory says lower reverse KL has to raise speculative-decoding acceptance. Table 3 confirms it and then goes further, to wall-clock time. On acceptance, greedy improves every non-Latin script by +3.2 to +6.6pp and ties on English, for +1.4pp globally ($p=0.004$). Under high skew the aggregate gain is +3.3pp ($p<10^{-3}$), stable across draft lengths $\gamma \in \{2, 4, 8\}$ (Appendix G). Hindi is the one reversal, and it is a property of the tokenizer rather than the algorithm, since on Gemma-3-1B greedy improves Hindi reverse KL by 0.10 nats (Appendix F.1).

Acceptance is still a sampling-level proxy, so we also run the FR-Spec/EAGLE-2 pipeline [18, 10] and time it directly, pruning only the drafter’s vocabulary while the target keeps the full one. Two findings matter for on-device serving. First, pruning pays off only when calibrated to the deployment: FR-Spec’s shipped English-derived index *slows* multilingual decoding by 2.3%, whereas re-selecting a same-size index on the actual deployment gains 12.5% over the unpruned drafter (Appendix G.2). Second, at equal budget the choice of rule is itself worth up to 6.1% throughput: greedy wins in every cell of the sweep, and all five languages clear Bonferroni correction ($\alpha/5=0.01$) under a one-sided paired Wilcoxon test on accepted length.



Regime	Method	en	fr	ar	ja	ru	zh	All
Uniform	Frequency	1.27	1.32	1.57	1.86	1.87	2.16	1.36
	Greedy	1.28	1.31	1.51	1.65	1.70	1.87	1.33
Weighted	Frequency	0.79	1.35	2.39	2.34	2.42	2.73	1.68
	Greedy	0.87	1.26	1.86	2.03	2.17	2.26	1.51

Figure 3: Reverse KL on Qwen3.5-0.8B-Base with FLORES-200. *Left*: vs. support size K under English-dominant calibration. *Right*: per-language at $K=1,000$ (bold is better). Frequency starves minority-script languages, while greedy rescues them.

Lang	Draft acceptance (Qwen3.5-0.8B)			Accepted length, FR-Spec (Llama-3.2-1B)			
	Freq.	Greedy	Δ	Freq.	Greedy	Δ tok/s	W/L
ar	0.23	0.28	+23.5%	1.139	1.233	+6.1%	33/1
ru	0.25	0.28	+12.8%	1.372	1.402	+1.9%	19/2
zh	0.20	0.27	+33.0%	1.159	1.185	+2.1%	20/0
ja	0.22	0.26	+17.3%	1.277	1.295	+1.3%	14/5
hi	0.57	0.46	-20.4%	1.543	1.558	+0.7%	20/7

Table 3: Efficiency at $K=1,000$, both Δ columns relative. *Left*: per-token draft acceptance on the 10-language deployment ($n=200$). The global gap is +1.4pp ($p=0.004$) and English ties at 0.56 vs. 0.55. *Right*: mean accepted length and measured throughput in the FR-Spec/EAGLE-2 pipeline ($n=40$ per cell, en 0.99 calibration), with per-prompt paired wins/losses.

5 Conclusion

Pruning the output vocabulary offers a cheap and effective method to optimize LLMs for edge device deployment, shrinking the corresponding LLM layer by up to $256\times$ in memory and $92\times$ in its share of decode time. Which tokens to keep, though, has usually been settled by a frequency-based heuristic. This paper proposes an objective for the token selection problem, the expected log retained mass, and shows it is equivalent to reverse KL against the teacher and monotone submodular, and selects tokens by greedily maximizing it under a $(1 - 1/e)$ guarantee. We compare this greedy approach against the two rules a deployment would otherwise use: frequency pruning and a magnitude baseline.

The savings in memory and latency come from pruning itself, and they compose with quantization, together reaching $3.5\times$ less weight memory at 95% of full-precision quality. However, the retained quality depends on the used token selection approach. At an equal budget of $K=1,000$ tokens, greedy selection is consistently better than frequency on the skewed, multi-script data an edge deployment typically sees, cutting Arabic reverse KL by 0.53 nats and gaining +1.4 chrF on Russian at negligible cost to English, and the data-free magnitude rule is not competitive anywhere. That better support also decodes faster, adding +5.4pp of draft acceptance on Arabic and up to 6.1% throughput in a speculative-decoding pipeline. On a homogeneous single-script deployment the two rules coincide, exactly as the theory predicts, so the gain appears where the deployment is heterogeneous and is free where it is not. The wider lesson is that a model shipped to a device serves a narrow and largely known slice of the world. Any fixed inference budget, not only the vocabulary, is better allocated against that slice than against a global average.

References

- [1] Nikolay Bogoychev, Pinzhen Chen, Barry Haddow, and Alexandra Birch. The ups and downs of large language model inference with vocabulary trimming by language heuristics. In *Proceedings of the 1st Workshop on Insights from Negative Results in NLP*, 2024.
- [2] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.
- [3] Aleksei Dorkin, Taïdo Purason, and Kairit Sirts. Prune or retrain: Optimizing the vocabulary of multilingual models for Estonian. *arXiv preprint arXiv:2501.02631*, 2025.
- [4] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4): 634–652, 1998.
- [5] Gemma Team. Gemma 3 technical report, 2025.
- [6] Raghav Goel, Sudhanshu Agrawal, Mukul Gagrani, Junyoung Park, Yifan Zao, He Zhang, Tian Liu, Yiping Yang, Xin Yuan, Jiuyan Lu, et al. Vocabtrim: Vocabulary pruning for efficient speculative decoding in llms. *arXiv preprint arXiv:2506.22694*, 2025.
- [7] Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. *Advances in Neural Information Processing Systems*, 28, 2015.

- [8] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9, 2008.
- [9] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*. PMLR, 2023.
- [10] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: Faster inference of language models with dynamic draft trees. *arXiv preprint arXiv:2406.16858*, 2024.
- [11] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1):265–294, 1978.
- [12] NLLB Team. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*, 2022.
- [13] Maja Popović. chrF: Character n -gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, 2015.
- [14] Qwen Team. Qwen3.5: Towards native multimodal agents. <https://qwen.ai/blog?id=qwen3.5>, February 2026.
- [15] Asahi Ushio, Yi Zhou, and Jose Camacho-Collados. An efficient multilingual language model compression through vocabulary trimming. In *Findings of the Association for Computational Linguistics: EMNLP*, 2023.
- [16] Jort Vincenti, Karim Abdel Sadek, Joan Velja, Matteo Nulli, and Metod Jazbec. Dynamic vocabulary pruning in early-exit LLMs. *arXiv preprint arXiv:2410.18952*, 2024.
- [17] Miles Williams, Young D Kwon, Rui Li, Alexandros Kouris, and Stylianos I Venieris. Speculative decoding with a speculative vocabulary. *arXiv preprint arXiv:2602.13836*, 2026.
- [18] Weilin Zhao, Tengyu Pan, Xu Han, Yudi Zhang, Ao Sun, Yuxiang Huang, Kaihuo Zhang, Weilun Zhao, Yuxuan Li, Jianyong Wang, Zhiyuan Liu, and Maosong Sun. FR-spec: Accelerating large-vocabulary language models via frequency-ranked speculative sampling. *arXiv preprint arXiv:2502.14856*, 2025.

A Theory and Proofs

This appendix states every formal result referenced in Section 3 and proves it, in the order the results appear in the main text. Each is restated in full so the appendix stands alone.

A.1 Linear retained mass yields frequency pruning

Proposition 1 (Linear retained mass yields frequency pruning). *For any feasible $A \subseteq U$, $L(A) = \mathbb{E}[M_c(\emptyset)] + \sum_{u \in A} m(u)$. Consequently, any maximizer of $L(A)$ under $|A| = r$ is obtained by taking the r tokens with largest marginal predictive frequency $m(u)$.*

Proof. From the definition of retained mass (Eq. (1)),

$$M_c(A) = \sum_{v \in B} p_c(v) + \sum_{u \in A} p_c(u).$$

Taking expectation over $c \sim \mu$ and using linearity of expectation:

$$\begin{aligned} L(A) &= \mathbb{E} \left[\sum_{v \in B} p_c(v) \right] + \sum_{u \in A} \mathbb{E}[p_c(u)] \\ &= \mathbb{E}[M_c(\emptyset)] + \sum_{u \in A} m(u). \end{aligned}$$

The first term $\mathbb{E}[M_c(\emptyset)]$ depends only on the fallback support B and is independent of the choice of A . Therefore, maximizing $L(A)$ subject to $|A| = r$ is equivalent to maximizing $\sum_{u \in A} m(u)$, which is a modular (additive) objective. The maximum is attained by the r tokens of largest $m(u)$. $\square$

A.2 Exact reverse-KL characterization

Theorem 2 (Exact reverse-KL characterization). *Fix a context c and a feasible support A . Then $q_{c,A}^*$ defined in Eq. (3) minimizes $D_{\text{KL}}(q \| p_c)$ over all $q \in \mathcal{Q}_A$. Moreover, the minimum value is $D_{\text{KL}}(q_{c,A}^* \| p_c) = -\log M_c(A)$.*

Proof. Take any $q \in \mathcal{Q}_A$. Because q is supported on S_A , we can write

$$D_{\text{KL}}(q \| p_c) = \sum_{v \in S_A} q(v) \log \frac{q(v)}{p_c(v)}.$$

For $v \in S_A$, Eq. (3) gives $q_{c,A}^*(v) = p_c(v)/M_c(A)$. We can therefore rewrite the log-ratio as

$$\begin{aligned} \log \frac{q(v)}{p_c(v)} &= \log \frac{q(v)}{q_{c,A}^*(v) \cdot M_c(A)} \\ &= \log \frac{q(v)}{q_{c,A}^*(v)} - \log M_c(A). \end{aligned}$$

Substituting back into the KL expression:

$$\begin{aligned} D_{\text{KL}}(q \| p_c) &= \sum_{v \in S_A} q(v) \log \frac{q(v)}{q_{c,A}^*(v)} \\ &\quad - \log M_c(A) \sum_{v \in S_A} q(v). \end{aligned}$$

Since q is a probability distribution supported on S_A , we have $\sum_{v \in S_A} q(v) = 1$. Therefore

$$D_{\text{KL}}(q \| p_c) = D_{\text{KL}}(q \| q_{c,A}^*) - \log M_c(A).$$

The first term is a KL divergence, which is nonnegative and equals zero if and only if $q = q_{c,A}^*$. The second term $-\log M_c(A)$ is a constant that does not depend on q . Therefore the minimum over $q \in \mathcal{Q}_A$ is achieved at $q = q_{c,A}^*$, and the minimum value is $-\log M_c(A)$. $\square$

A.3 Monotonicity and submodularity

Theorem 3 (Monotonicity and submodularity). *For every context c , the set function $f_c(A) = \log M_c(A)$ is monotone and submodular on subsets of U . Consequently, the population objective $F(A) = \mathbb{E}[f_c(A)]$ is also monotone and submodular.*

Proof. Fix a context c . Let $A \subseteq U$ and let $u \in U \setminus A$. The marginal gain of adding u is

$$\begin{aligned} f_c(A \cup \{u\}) - f_c(A) &= \log(M_c(A) + p_c(u)) - \log M_c(A) \\ &= \log\left(1 + \frac{p_c(u)}{M_c(A)}\right). \end{aligned}$$

Because $p_c(u) \geq 0$ and $M_c(A) > 0$, this quantity is nonnegative. Hence f_c is monotone: adding any token to the support can only increase (or maintain) the objective.

For submodularity, let $A \subseteq A' \subseteq U$ and let $u \in U \setminus A'$. Since $A \subseteq A'$, we have $M_c(A) \leq M_c(A')$. The function $x \mapsto \log(1 + p_c(u)/x)$ is strictly decreasing on $(0, \infty)$ (its derivative with respect to x is $-p_c(u)/(x(x + p_c(u))) \leq 0$). Therefore

$$\log\left(1 + \frac{p_c(u)}{M_c(A)}\right) \geq \log\left(1 + \frac{p_c(u)}{M_c(A')}\right),$$

which can be rewritten as

$$f_c(A \cup \{u\}) - f_c(A) \geq f_c(A' \cup \{u\}) - f_c(A').$$

This is exactly the diminishing-returns condition that defines submodularity: the marginal value of adding u is at least as large when the current set is smaller.

Finally, both monotonicity and submodularity are preserved under nonnegative linear combinations. Since $F(A) = \mathbb{E}_{c \sim \mu}[f_c(A)]$ is an expectation (i.e., a nonnegative weighted sum) of monotone submodular functions, F is itself monotone and submodular. $\square$

A.4 Hardness and greedy approximation

Theorem 4 (Hardness and greedy approximation).

- (i) *Maximizing a monotone submodular function under a cardinality constraint is NP-hard.*
- (ii) *No polynomial-time algorithm achieves a ratio better than $(1 - 1/e)$ unless $P=NP$.*
- (iii) *Greedy achieves this bound:*

$$F(A_r) \geq F(\emptyset) + (1 - \frac{1}{e})(F(A^*) - F(\emptyset)).$$

All three parts are classical and we do not reprove them. Cardinality-constrained monotone submodular maximization is NP-hard and inapproximable beyond $(1 - 1/e)$ [4], and the matching greedy guarantee is due to Nemhauser et al. [11]. What our Theorem 3 supplies is the precondition: once F is shown monotone submodular, the bound transfers to output-support selection unchanged.

A.5 First-order approximation and error bound

Theorem 5 (First-order approximation). *Assume $R_c(A) \leq \rho < 1$ for all feasible A and c . Then*

$$F(A) = -\mathbb{E}[R_c(A)] - \mathbb{E}[\eta(R_c(A))],$$

where $\eta(r) := \sum_{k=2}^{\infty} r^k/k$ satisfies $0 \leq \eta(r) \leq r^2/(2(1-\rho))$. The first-order surrogate is frequency.

Proof. For $r \in [0, 1)$, the Taylor expansion of the logarithm gives

$$\log(1 - r) = -\sum_{k=1}^{\infty} \frac{r^k}{k} = -r - \underbrace{\sum_{k=2}^{\infty} \frac{r^k}{k}}_{=: \eta(r)}.$$

The remainder $\eta(r)$ is clearly nonnegative. To bound it from above for $r \in [0, \rho]$:

$$\begin{aligned}\eta(r) &= \sum_{k=2}^{\infty} \frac{r^k}{k} \leq \frac{1}{2} \sum_{k=2}^{\infty} r^k \\ &= \frac{r^2}{2(1-r)} \leq \frac{r^2}{2(1-\rho)},\end{aligned}$$

where the first inequality uses $1/k \leq 1/2$ for $k \geq 2$, and the second uses $r \leq \rho$.

Now substitute $r = R_c(A) = 1 - M_c(A)$ and take expectations:

$$\begin{aligned}F(A) &= \mathbb{E}[\log(1 - R_c(A))] \\ &= -\mathbb{E}[R_c(A)] - \mathbb{E}[\eta(R_c(A))].\end{aligned}$$

To identify the first-order term, expand $\mathbb{E}[R_c(A)]$:

$$\mathbb{E}[R_c(A)] = 1 - \mathbb{E}\left[\sum_{v \in B} p_c(v)\right] - \sum_{u \in A} m(u).$$

The first two terms are independent of A . So minimizing the first-order term $\mathbb{E}[R_c(A)]$ is equivalent to maximizing $\sum_{u \in A} m(u)$, which is frequency pruning (Theorem 1). $\square$

A.6 Common-ranking condition

Theorem 6 (Common-ranking condition). *If there exists a total order $u_1 \succ \dots \succ u_{|U|}$ on U such that $p_c(u_1) \geq \dots \geq p_c(u_{|U|})$ for every context c , then $A_{\text{rank}} := \{u_1, \dots, u_r\}$ maximizes $F(A)$ over all $|A| = r$, and coincides with the top- r set by marginal frequency.*

Proof. Fix any feasible set $A \subseteq U$ with $|A| = r$, and fix any context c . Because $u_1, \dots, u_r$ are the r tokens with the largest probabilities under p_c (by the common-ranking assumption), we have

$$\sum_{j=1}^r p_c(u_j) \geq \sum_{u \in A} p_c(u).$$

Adding the baseline mass $\sum_{v \in B} p_c(v)$ to both sides gives

$$M_c(A_{\text{rank}}) \geq M_c(A).$$

Since $\log$ is an increasing function:

$$\log M_c(A_{\text{rank}}) \geq \log M_c(A).$$

This holds for *every* context c . Taking expectation over $c \sim \mu$:

$$F(A_{\text{rank}}) \geq F(A).$$

Since A was arbitrary, A_{rank} is optimal.

For the second claim: if $u_i \succ u_j$, then $p_c(u_i) \geq p_c(u_j)$ for every context c . Taking expectations gives $m(u_i) = \mathbb{E}[p_c(u_i)] \geq \mathbb{E}[p_c(u_j)] = m(u_j)$. Hence the common ranking is also the ranking by marginal predictive frequency, and A_{rank} is the frequency solution. $\square$

A.7 Frequency can be strictly suboptimal

Theorem 7 (Frequency pruning can be strictly suboptimal). *There exist a deployment distribution over contexts and a budget r such that the support maximizing $\sum_{u \in A} m(u)$ is not a maximizer of $F(A)$.*

Proof. Let the optional vocabulary be $U = \{u, b, c\}$ and let the budget be $r = 2$. Assume the fallback support contributes probability 0.01 in every context. Consider two contexts, c_1 and c_2 , each occurring with probability $1/2$, with teacher probabilities:

$$p_{c_1}(u) = 0.34, \quad p_{c_1}(b) = 0.65, \quad p_{c_1}(c) = 0,$$

$$p_{c_2}(u) = 0.34, \quad p_{c_2}(b) = 0, \quad p_{c_2}(c) = 0.65.$$

The marginal predictive frequencies are $m(u) = 0.34$, $m(b) = 0.325$, $m(c) = 0.325$. Since $m(u) > m(b) = m(c)$, frequency selects u and one of b, c ; take $A_{\text{freq}} = \{u, b\}$.

Under A_{freq} , the retained masses are:

$$M_{c_1}(A_{\text{freq}}) = 0.01 + 0.34 + 0.65 = 1.00,$$

$$M_{c_2}(A_{\text{freq}}) = 0.01 + 0.34 + 0 = 0.35.$$

Context c_1 is perfectly covered, but context c_2 retains only 35% of its mass. The objective value is

$$\begin{aligned} F(A_{\text{freq}}) &= \frac{1}{2} \log 1.00 + \frac{1}{2} \log 0.35 \\ &= \frac{1}{2} \log 0.35 \approx -0.525. \end{aligned}$$

Now consider the alternative $A_{\text{spec}} = \{b, c\}$. Its retained masses are:

$$M_{c_1}(A_{\text{spec}}) = 0.01 + 0.65 + 0 = 0.66,$$

$$M_{c_2}(A_{\text{spec}}) = 0.01 + 0 + 0.65 = 0.66.$$

Both contexts are covered equally. The objective value is

$$F(A_{\text{spec}}) = \log 0.66 \approx -0.416.$$

Since $-0.416 > -0.525$ we have $F(A_{\text{spec}}) > F(A_{\text{freq}})$, so frequency is strictly suboptimal.

The intuition is clear: token u is the “generalist”: it has moderate mass in both contexts. Tokens b and c are “specialists”: each is critical in exactly one context and useless in the other. The linear criterion prefers the generalist because it contributes more mass on average. But the log criterion recognizes that the generalist’s contribution to c_1 (where coverage is already excellent) is worth much less than the specialist’s contribution to c_2 (where coverage is catastrophically low). This is exactly the diminishing-returns mechanism formalized by Theorem 3. $\square$

B Lazy-Greedy Algorithm

Algorithm 1 gives the procedure used in all experiments. Submodularity makes marginal gains non-increasing in A , so any previously computed gain upper-bounds its current value. Keying a max-heap on those bounds and recomputing only on pop gives the same selection far more cheaply.

Algorithm 1 Lazy Greedy for Log-Retained-Mass Maximization

Require: Calibration distributions $p_1, \dots, p_n$ over V ; budget r ; fallback mass β

Ensure: Selected support A with $|A| = r$

```

1:  $M_i \leftarrow \beta$  for all  $i = 1, \dots, n$ 
2: Compute  $\Delta(v) \leftarrow \frac{1}{n} \sum_{i=1}^n \log(1 + p_i(v)/M_i)$  for all  $v \in U$ 
3: Insert all  $(v, \Delta(v))$  into max-heap  $\mathcal{H}$ 
4:  $A \leftarrow \emptyset$ 
5: for  $k = 1$  to  $r$  do
6:   repeat
7:     Pop  $(v, \text{key})$  from  $\mathcal{H}$ 
8:      $\Delta \leftarrow \frac{1}{n} \sum_{i=1}^n \log(1 + p_i(v)/M_i)$  {recompute exact gain}
9:     if  $\Delta \geq \text{top key of } \mathcal{H}$  then
10:       $A \leftarrow A \cup \{v\}$ ;  $M_i \leftarrow M_i + p_i(v)$  for all  $i$ 
11:    else
12:      Push  $(v, \Delta)$  back into  $\mathcal{H}$ 
13:    end if
14:  until  $|A| = k$ 
15: end for
16: return  $A$ 

```

Proposition 8 (Lazy greedy is exact). *Let u be the candidate accepted at termination of a given round of Algorithm 1, and let $U(w)$ denote the stored upper bound on each remaining candidate $w \in U \setminus A$. Then u has the largest exact current marginal gain among all remaining candidates, and is precisely the token that standard greedy would select.*

Proof. By the stopping condition, after recomputation the exact gain of u satisfies $\Delta(u \mid A) \geq U(w)$ for every remaining candidate w . Since each stored key $U(w)$ is an upper bound on $\Delta(w \mid A)$ (submodularity: marginal gains cannot increase as A grows), $\Delta(w \mid A) \leq U(w) \leq \Delta(u \mid A)$ for all w . Hence u is the exact argmax of $\Delta(\cdot \mid A)$, which is what standard greedy selects. $\square$

C Experimental Details

C.1 Multilingual and continuation experiments

Qwen3.5-0.8B-Base [14] ($|V|=248,320$) on ten FLORES-200 languages [12]: English, French, Spanish, German, Swahili (Latin); Russian (Cyrillic); Arabic; Hindi (Devanagari); Japanese, Chinese (CJK). We tokenize the dev split for calibration and devtest for evaluation, extract a context at every token position for $\sim 1,000$ contexts per language per split, and store the top-8,192 next-token probabilities as a sparse teacher representation. Calibration is either *uniform* ($w_\ell = 1/10$) or *English-dominant* ($w_{\text{en}} = 0.6$, the other nine sharing 0.4). Evaluation is always uniform. Support sizes are $K \in \{1, 2, 4, 6, 10\} \times 10^3$.

For continuation we reuse those support sets, prompt with the first 60% of each evaluation sentence, and generate up to 80 tokens by greedy decoding with all non-retained logits set to $-\infty$. We score with chrF [13] ($n \leq 6$), reporting per-language values and the macro average over the six evaluation languages (en, fr, ar, ja, ru, zh).

C.2 Robustness across deployment types

The robustness study (Section 4) evaluates four deployments at a fixed budget $K=1,000$, each defined by a set of labels and a calibration weight on the dominant label (the remainder split equally among the others): *Latin (5 langs)*, five Latin-script languages from Wikipedia (en 0.60; fr/de/es/it 0.10 each); *Multi-script (10 langs)*, the ten FLORES-200 languages with graduated weights (en 0.60; fr/es 0.08; de 0.05; ru/ar 0.04; zh/hi/ja 0.03; sw 0.02), spanning five scripts; *Multi-script (5 langs)*, five disjoint scripts from Wikipedia (Latin/en 0.60; Cyrillic/ru, Arabic/ar, CJK/zh, Devanagari/hi 0.10 each); and *High skew (en+ar, 0.99)*, two FLORES-200 languages at extreme skew (en 0.99; ar 0.01). Calibration weights the contexts of each label by its share, and evaluation is uniform over labels. Context construction, the top-8,192 sparse teacher representation, and the chrF generation protocol are identical to Appendix C.1.

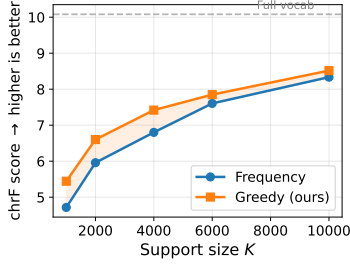
We compare only rules that share an output-vocabulary-only intervention at equal budget. Vocab-Trim [6] and SpecVocab [17] also reduce a drafter vocabulary but bundle orthogonal architectural changes, so they are not numerically comparable on this axis and we do not include them.

D Synthetic Multi-Domain Study

To test the mechanism where heterogeneity is directly controllable, we build a synthetic vocabulary of $|V|=10,000$ tokens with up to $D=10$ domains, each contributing 50 domain-critical (“hot”) tokens. Contexts are Dirichlet draws around domain prototypes, and domains are mixed with harmonic weights $w_d \propto 1/(d+1)$, so the majority domain holds $\sim 34\%$ of contexts and the smallest only $\sim 3\%$. We sample 5,000 calibration and 2,500 evaluation contexts per configuration.

Figure 2 confirms both predictions of Section 3. Greedy wins at every budget and by the widest margin at the tightest: at $r=100$ it reaches 1.84 nats against 2.53 for frequency, and strands only 31% of evaluation contexts at the fallback floor where frequency strands 49%. Sweeping D isolates heterogeneity as the cause. At $D=1$ the common-ranking condition holds exactly and the two rules select the same support, as Theorem 6 requires. For $D \geq 2$ a gap opens and widens monotonically, reaching 2.48 against 3.22 at $D=10$. The mechanism is the one Theorem 7 identifies: under harmonic weights frequency spends the whole budget on the one or two most prevalent domains, while greedy spreads it roughly in proportion to domain weight, so even the smallest domain retains usable mass.

The vocabulary of $|V|=10,000$ tokens splits into a shared block of 500 tokens common to all domains, up to 10 domain blocks of 500 tokens each, and a tail block of 4,500. Within each domain block, 50 tokens form the *hot subset*, drawn once under a fixed seed and reused across configurations. Each domain d has a prototype $\pi^{(d)}$ placing mass 0.55 uniformly on its hot tokens, 0.30 on the shared



K	Method	en	fr	ar	ja	ru	zh	Avg
–	Full vocab	13.4	14.5	10.6	5.1	12.5	3.9	10.0
1k	Freq.	10.1	9.2	4.2	0.8	3.0	0.9	4.7
	Greedy	10.0	9.7	6.3	1.0	4.4	1.1	5.4
4k	Freq.	11.3	10.9	8.5	1.8	6.7	1.4	6.8
	Greedy	11.2	11.3	8.5	2.3	9.1	1.8	7.4
10k	Freq.	12.8	11.8	9.4	3.3	10.3	2.2	8.3
	Greedy	12.7	11.7	9.4	3.6	10.8	2.5	8.5

Figure 4: Text continuation under English-dominant calibration. *Left*: macro chrF vs. K . *Right*: per-language chrF at selected K (bold is better). Greedy improves ru, ja, zh at negligible cost to English.

Method	en	fr	es	de	ru	ar	zh	hi	ja	sw
Frequency	0.28	0.62	0.58	0.77	1.21	0.75	0.97	0.84	1.02	0.58
Greedy	0.32	0.64	0.59	0.77	1.12	0.73	0.89	0.74	0.93	0.58

Table 4: Per-language reverse KL on Gemma-3-1B-PT at $K=10,000$. Greedy improves Russian, Chinese, Hindi, and Japanese (-0.08 to -0.10 nats each) at a cost of $+0.04$ on English, mirroring the Qwen3.5-0.8B pattern of Figure 3 (right). Frequency is slightly better on three Latin-script languages, but greedy is strictly better on every non-Latin script.

block, 0.10 on the rest of its own block, and 0.05 on everything else. Contexts are Dirichlet draws around that prototype, $p_i \sim \text{Dirichlet}(\alpha\pi^{(d)})$ with concentration $\alpha = 80$, so higher α means contexts closer to the prototype. Domain d receives harmonic weight $w_d \propto 1/(d+1)$, giving $\lfloor w_d \cdot 5,000 \rfloor$ calibration and proportional evaluation contexts. The two splits use different seeds but the same hot subsets. The budget sweep fixes $D=10$ and varies $r \in \{100, 300, 500, 700, 900\}$, reusing one calibration/evaluation set throughout. The domain sweep fixes $r=50$ and varies $D \in \{1, \dots, 10\}$.

E Per-Language Multilingual Results

Section 4 summarizes the multilingual comparison at the deployment level. This section gives the per-language breakdown behind those numbers on Qwen3.5-0.8B-Base with FLORES-200, under both calibration regimes: *uniform* (all ten languages equal) and *English-dominant* (60% English, 40% split among the remaining nine). Evaluation is always uniform across languages.

Figure 3 reports reverse KL. Under English-dominant calibration greedy leads at every support size, most at $K=1,000$ (0.17 nats globally). The per-language columns show the mechanism: frequency spends the budget on English-relevant tokens, driving Arabic to 2.39 nats and Chinese to 2.73, while greedy recovers 0.53 and 0.47 of that at a cost of 0.08 to English. Under uniform calibration the global gap narrows to 0.03, yet greedy still improves Chinese (-0.29), Japanese (-0.21), and Russian (-0.17), consistent with the prediction that the advantage grows with heterogeneity.

Figure 4 confirms that the distributional gain reaches generation. Macro chrF favours greedy at every K , most at small budgets (5.4 vs. 4.7 at $K=1,000$). Russian is the clearest case, 4.4 vs. 3.0 chrF ($+47\%$ relative, against a full-vocabulary ceiling of 12.5). At $K=1,000$ both rules produce low absolute quality on non-Latin scripts and usable output emerges only from $K \geq 4,000$, but the ordering holds at every budget, greedy rescuing the Cyrillic, Arabic, and CJK tokens frequency drops.

F Additional Empirical Results

F.1 Cross-family validation on Gemma-3-1B

To verify that the qualitative pattern of Appendix E is not specific to the Qwen tokenizer family, we re-run the multilingual distributional experiment on Gemma-3-1B-PT [5] ($V=262,144$), using the same FLORES-10 corpus and the same graduated English-dominant calibration weights. Table 4 gives the per-language breakdown at $K=10,000$, where greedy leads by ~ 0.03 nats.

en weight	Magnitude	Frequency	Greedy	Δ (freq–greedy)
0.50	3.78	1.57	1.44	+0.13
0.75	3.78	1.83	1.57	+0.26
0.90	3.78	2.18	1.84	+0.35
0.99	3.78	2.92	2.51	+0.41

Table 5: Global reverse KL on FLORES-10 at $K=1,000$ as a function of calibration skew (Qwen3.5-0.8B-Base). The greedy–frequency gap grows monotonically with skew, in line with Theorem 7. Magnitude is independent of calibration weights, so its value is constant across rows.

Method	rev-KL	chrF
<i>Latin-5 (low heterogeneity)</i>		
Full vocab	≈ 0	19.84
Greedy	1.317	13.26
Frequency	1.354	12.83
Magnitude	4.461	5.19
<i>Multi-script-5 (disjoint scripts)</i>		
Full vocab	≈ 0	15.10
Greedy	1.416	8.72
Frequency	1.463	8.82
Magnitude	4.502	0.85
<i>High skew (en+ar 0.99/0.01)</i>		
Full vocab	≈ 0	14.42
Greedy	1.571	9.18
Frequency	2.386	6.11
Magnitude	4.445	1.93

Table 6: Cross-family quality on Llama-3.2-1B ($|V|=128,256$), reverse KL and macro chrF at $K=1,000$ on three deployment cells of increasing heterogeneity (Tatoeba prompts, so absolute values are not comparable to the FLORES cells, though both rules see identical prompts). The rules tie on Latin-5 as common-ranking predicts, and greedy dominates under high skew, where at $K=1000$ it keeps Arabic usable (per-language chrF 6.55) against frequency’s broken 0.27. Full-vocabulary reverse KL is ≈ 0 by Theorem 2 since nothing is pruned.

F.2 Calibration-skew sweep on FLORES-10

Appendix E reports two calibration regimes (uniform and 60% English). To map the dependence of the greedy advantage on calibration skew, we re-run the FLORES-10 cell at $K=1,000$ with the dominant English weight varied across $\{0.50, 0.75, 0.90, 0.99\}$ (remainder split equally among the other nine languages); Table 5 reports global reverse-KL for each method.

F.3 Cross-family quality on Llama-3.2-1B

To check that the deployment-heterogeneity pattern of Table 2 is not specific to the Qwen tokenizer, we repeat three of its deployment cells on Llama-3.2-1B ($|V|=128,256$). Table 6 reports reverse KL and macro chrF at $K=1,000$; by $K=4,000$ all three cells are near-ties, as the theory expects once the budget covers the per-context tops.

F.4 Cross-model generality

Table 7 repeats the greedy-versus-frequency quality comparison on three additional models beyond the main-text Qwen3.5-0.8B (Qwen2.5-0.5B, Qwen2.5-7B, and Gemma-2-2B), spanning two families and 0.5 to 7B parameters, on the ten-language multi-script deployment. Greedy improves both reverse KL and chrF on every model, so the advantage is not tied to a single model, family, or scale.

Model	Reverse KL ($\downarrow$)				chrF ($\uparrow$)			
	Full	Mag.	Freq.	Greedy	Full	Mag.	Freq.	Greedy
Qwen2.5-0.5B	≈ 0	4.592	1.472	1.279	12.52	1.04	7.92	8.41
Qwen2.5-7B	≈ 0	4.600	1.707	1.518	15.94	1.00	10.77	11.47
Gemma-2-2B	≈ 0	4.064	1.317	1.208	4.06	0.21	1.92	1.98

Table 7: Cross-model quality at $K=1,000$ on the ten-language multi-script deployment, for three models beyond the main-text Qwen3.5-0.8B spanning two families and 152k to 256k vocabularies. Greedy dominates both baselines on both metrics for every model, while magnitude collapses throughout.

Deployment	Magn.	Freq.	Greedy	Δ pp, 95% CI
Latin (5 langs)	0.08	0.49	0.48	$-0.9 [-3.0, +1.3]$
Multi-script (10 langs)	0.05	0.37	0.38	$+1.4 [+0.4, +2.5]$
Multi-script (5 langs)	0.05	0.44	0.43	$-0.7 [-2.2, +0.9]$
High skew (en+ar, 0.99)	0.05	0.37	0.40	$+3.3 [+1.6, +4.9]$

Table 8: Aggregate sample-based speculative-decoding draft acceptance on all four deployments of Table 2 at $K=1,000$ (Qwen3.5-0.8B-Base). Magnitude is the data-free baseline, and bold marks the best pruned method. The two deployments with the smallest reverse-KL gaps in Table 2 (Latin and Multi-script-5, both ≤ 0.08 nats) have $|\Delta| \leq 0.9$ pp with CIs straddling zero, consistent with the theoretical prediction that small reverse-KL gaps yield small acceptance gaps.

G Speculative Decoding: Additional Results

G.1 Sample-based acceptance: full matrix and γ sweep

The body Table 3 reports per-language results for the two deployments where the underlying reverse-KL gap is non-trivial. Table 8 gives the full aggregate-acceptance matrix on all four deployments, with the magnitude data-free baseline included.

Repeating the high-skew cell at $\gamma \in \{2, 4, 8\}$ leaves greedy’s per-token acceptance essentially flat (~ 0.40) while frequency drifts down, so the paired gap widens from $+3.4$ pp at $\gamma=2$ to $+4.3$ pp at $\gamma=8$ ($n=80$ prompts). The finding does not depend on this hyper-parameter, and longer chains slightly favour greedy, since a longer chain is more likely to cross a specialist-token boundary.

G.2 End-to-end FR-Spec/EAGLE-2: calibration corpus, budget, and second family

Section 4 reports the headline end-to-end comparison. Two supporting panels complete it. Table 9 isolates the effect of the *calibration corpus*, at a budget large enough that the two selection rules coincide, and Table 10 sweeps the budget and adds a second model family at equal K .

Eval text	Drafter vocabulary	$\bar{\ell}$	tok/s	vs. full
English	Full (no pruning, $V=128$ k)	1.408	200.9	—
	FR-Spec frequency ($K=16$ k)	1.494	213.4	$+6.2\%$
Multilingual	Full (no pruning, $V=128$ k)	1.408	128.0	—
	FR-Spec index (English SlimPajama, $K=16$ k)	1.379	125.0	-2.3%
	Deployment-matched ($K=16$ k)	1.591	144.0	$+12.5\%$

Table 9: **(a) Pruning the drafter helps only if calibrated to the deployment.** Llama-3.2-1B-Instruct with its shipped 1-layer EAGLE-2 drafter, pruning only the drafter’s vocabulary. FR-Spec’s English-derived index speeds up English ($+6.2\%$, reproducing their result) but *slows* multilingual decoding (-2.3%). Re-selecting the same-size index on the true deployment recovers $+12.5\%$ over full and $+15\%$ over FR-Spec. At this large K greedy and frequency coincide (144.0 vs. 144.2 tok/s), so this panel isolates the calibration corpus, not the rule.

Setting	$\bar{\ell}_{\text{freq}}$	$\bar{\ell}_{\text{greedy}}$	tok/s (freq)	tok/s (greedy)	greedy win/tie/lose
<i>Vocabulary budget K (Arabic eval, en 0.99, $n=80$ prompts)</i>					
$K=500$	1.084	1.136	107.1	111.9	57 / 22 / 1
$K=1000$	1.145	1.253	111.1	117.9	63 / 15 / 2
$K=2000$	1.347	1.400	127.3	130.5	47 / 19 / 14
<i>Cross-model: Qwen2-7B ($V=152k$), Arabic, $K=1,000$ ($n=40$ prompts)</i>					
Qwen2-7B	1.285	1.338	28.8	29.7	34 / 2 / 4

Table 10: **(b) Budget and model sweeps at equal K .** Same pipeline as (a), both columns pruned to identical size and differing only in which tokens are kept. The per-language rows are in Table 3. Win/tie/lose counts how many of the n paired prompts greedy accepted more, equal, or fewer drafted tokens on. The advantage is largest at the tightest budget and holds on a second family at 7B. Direction is order-invariant, and a frequency-vs-frequency control gives 0.04%.

Component	Params (M)	BF16 (GB)
Embedding, full ($ V =248,320$)	254.3	0.509
Embedding, pruned ($K=128k$)	131.1	0.262
Backbone	498.0	0.996
Total, full vocab	752.3	1.505
Total, pruned $K=128k$	629.1	1.258

Table 11: **Memory accounting for Qwen3.5-0.8B-Base.** Pruning shrinks embedding parameters, while quantization shrinks bytes-per-parameter. Because bitsandbytes skips the tied embedding, the 0.509 GB row is a hard floor for INT8/NF4 however hard the backbone is quantized.

H Vocabulary Pruning Under a Fixed Weight-Memory Budget

The experiments in Section 4 vary the token budget K directly, but a deployment engineer typically budgets *weight memory*, not token count, and asks a different question than Section 3: given N gigabytes, is it better to prune the vocabulary, quantize the weights, or combine both? We report a follow-up study on the same model (Qwen3.5-0.8B-Base), corpus (FLORES-200), calibration protocol, and greedy-versus-frequency comparison as Appendix E, reparametrized along a weight-memory axis and crossed with weight quantization (bitsandbytes INT8/NF4, and 21 llama.cpp/GGUF quantization tiers). Reverse KL is computed from the softmax of the actually deployed model, so a full-vocabulary quantized arm has nonzero reverse KL from quantization distortion alone even though no tokens are dropped. chrF is macro-averaged over the six evaluation languages of Appendix C.1 unless marked as a ten-language macro.

H.1 Pruning and quantization act on disjoint terms of the memory budget

Qwen3.5-0.8B-Base ties its input embedding and output head: of its 752.3M parameters, 254.3M sit in the embedding ($|V| \times d = 248,320 \times 1,024$) and 498.0M in the backbone. Weight-only quantization quantizes the backbone but, because the embedding is tied, bitsandbytes by design leaves it in BF16 (Table 11). The full-vocabulary embedding therefore imposes a 0.509 GB floor no amount of backbone quantization can cross. Only pruning shrinks this term, by physically removing embedding rows (Section 3’s S_A , realized as a smaller matrix rather than a masked one) rather than masking logits. llama.cpp/GGUF quantizes the embedding too, so a GGUF model can fall below the 0.509 GB floor before any pruning, and combining the two goes further still.

Two supporting facts come from a sweep we omit for space. Quality is stable from about four bits per weight upward and collapses below three, and layering greedy pruning on any tier at four bits or above shifts it roughly 0.1 GB left at a chrF cost between 0.01 and 0.11. Separately, the two composition orders deploy byte-identical weights and differ only in which distribution drives selection. Across 672 matched pairs spanning 21 GGUF tiers the mean gap is 0.006 nats and 0.03 chrF, worst case 0.071 nats at the near-degenerate 1-bit tier.

Lang	BF16 full 1.50 GB	Prune 128k 1.26 GB	NF4 full 0.76 GB	Q4_K_M full 0.53 GB	DEPLOY 0.43 GB
fr	16.76 (100%)	16.70 (100%)	16.22 (97%)	15.65 (93%)	15.45 (92%)
en	15.19 (100%)	15.18 (100%)	15.05 (99%)	15.07 (99%)	14.95 (98%)
ru	14.95 (100%)	14.92 (100%)	13.94 (93%)	14.01 (94%)	14.12 (95%)
ar	12.71 (100%)	12.72 (100%)	12.24 (96%)	12.34 (97%)	12.25 (96%)
ja	5.77 (100%)	5.67 (98%)	5.23 (91%)	5.66 (98%)	5.66 (98%)
zh	4.65 (100%)	4.34 (93%)	4.26 (92%)	4.64 (100%)	4.28 (92%)
Macro	11.67 (100%)	11.59 (99%)	11.15 (96%)	11.23 (96%)	11.12 (95%)

Table 12: **Per-language chrF at five operating points of increasing compression** (percent of BF16 retained in parentheses). Every language retains at least 92% of full-precision chrF at the DEPLOY point, a $3.5\times$ reduction in weight memory.

H.2 Per-language breakdown at the deployment point

Table 12 breaks the frontier down by language. Japanese and Chinese score low in absolute terms at *every* operating point, including full-precision BF16: that is the 0.8B base model’s own ceiling on those languages, not damage from compression.