THE OUTSIZED VOCABULARY TAX: SELECTIVE-VOCABULARY DECODING FOR EDGE LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern small language models pair a compact backbone with a large vocabulary which incur an output projection cost of $\mathcal{O}(Vd)$ per decoded token, reaching up to 28% of total parameters and $\sim$20% of decode latency on edge hardware. Recent cluster-routed decoding reduces this to $\mathcal{O}(Cd + |\mathcal{S}|d)$ without modifying any weights, by scoring only the tokens $\mathcal{S}$ in the few of C clusters whose centroids best match a given hidden state. Existing methods cluster LM-head rows directly, but we show that tokens close in weight space are often not the tokens that similar hidden states select. We introduce CHES (**C**alibrated **H**idden-state **E**mbedding **S**election), a training-free method that represents each token by a count-adaptive shrinkage estimate on the unit sphere. This estimate interpolates between the mean direction of the hidden states that select the token and a prior given by its LM-head row. We further propose AVQ, a closed-form anisotropic centroid update that aligns clustering with inference-time inner-product routing. On Qwen2.5 and Gemma2 models, CHES+AVQ reaches 95% top-1 routing accuracy while scoring only 6-8% of the vocabulary and yields up to 3.2$\times$ `LM-Head` speedup on Apple M4 and Intel Xeon CPUs. Across multiple downstream benchmarks (*e.g.,* GSM8K, MATH, TriviaQA), CHES+AVQ can closely match the full `LM-Head` performance while utilizing only $\sim$18-20% of total vocabulary.

1 INTRODUCTION

Scaling modern large language models (LLMs) has largely been centered around enlarging the transformer backbone with deeper stacks, wider feed-forward modules, and more attention heads. On the other hand, a key LLM component that has grown just as quickly but drawn far less scrutiny is the output projection that maps hidden states to vocabulary logits. As modern vocabularies expand to accommodate multilingual tokenization, code, and multimodal codebooks, this component has become one of the dominant parameter blocks in the model. For each decoded token, it executes a dense matrix multiplication against the full vocabulary. On edge hardware, where memory bandwidth dominates wall-clock time and power requirements, this single layer can compete with the cost of the entire transformer backbone.

The situation deteriorates in the regime that matters most for on-device inference: *large-vocabulary small models*. Recent models with multilingual tokenizers can push vocabulary into the 150K–300K range (Team et al., 2024; 2025; Xu et al., 2025; Yang et al., 2025; Tao et al., 2024), and a unified multimodal pipeline can further stack the text vocabulary with one or more visual codebooks within a single head (Wu et al., 2025a;b; Chameleon, 2024; Sun et al., 2024). Note that these trends scale vocabulary independently of the hidden dimension and the output projection can consume 30-40% of total parameters. In this work, we call this **Vocabulary Tax**: an inference cost that is paid at every generated token, scales with vocabulary size independently of model size, and grows more severe for small-model deployment. Figure 1 shows this cost for several models.

Pruning as a Remedy for Vocabulary Tax: A de-facto strategy to address this issue can be static vocabulary pruning by estimating token importance based on frequency on a calibration corpus, dropping the cold tail, and shipping a smaller head. Despite having significant potential for storage and compute savings, this straightforward technique has two limitations: ❶ pruned vocabulary performance is only as good as the calibration corpus with the risk that a long-tail corpus can permanently remove image codebooks and multi-lingual tokens; ❷ pruning permanently deletes

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

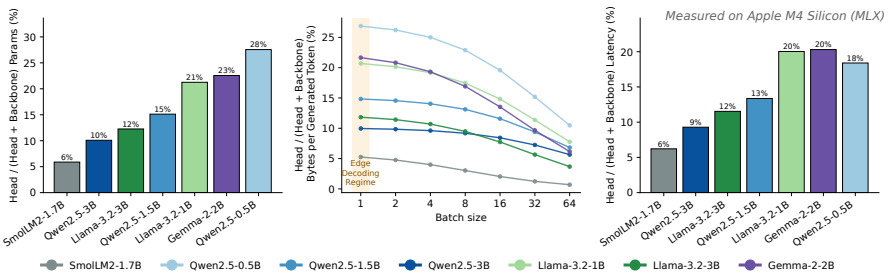


Figure 1: **The Vocabulary Tax on Edge Inference.** (a) *Parameter Share*: LM head parameter share across seven public models: the head consumes 6-28% of total parameters; (b) *Bandwidth Share*: An analytical estimate of head share of bytes moved per generated token vs. batch size (with S=2048). At batch=1 in conventional *edge decoding regime*, the head accounts for up to *27%* of memory traffic; (c) *Latency Share*: Measured decode-step latency share on Apple M4 Silicon (MLX, fp16, batch=1), the head consumes 6-20% of wall-clock time per token decoded.

vocabulary rows, which reshapes the input-output distribution and typically requires post-pruning finetuning with the added risk of unexpected regressing capabilities.

To address these limitations, some works (Shim et al., 2017; Chen et al., 2019a; Liu et al., 2025; Zhang et al., 2025) explore *sub-vocabulary decoding* such as candidate previewing and cluster-routed decoding. These methods partition the vocabulary into C clusters *offline*, and at every generated token open only the top-K clusters whose centroids best match the current hidden state. In this case, the dense $O(Vd)$ matrix multiplication collapses to $O(Cd + |\mathcal{S}|d)$, where $|\mathcal{S}|$ is the number of vocabulary tokens that fall into opened clusters. The vocabulary is preserved intact and no model weights are touched, so neither the calibration-mixture failure nor the retraining obligation of pruning applies. Existing methods primarily cluster the raw `LM-Head` rows (W) directly with Top-K Cluster Selection (TKCS), which we argue requires further exploration. The inference-time routing decision is made by an inner product $\langle c_k, h \rangle$ between a cluster centroid and the current hidden state h; it has nothing directly to do with the angular geometry of the W-rows. This motivates the central question of our work: *Prior works aim to answer "which K groups of tokens should I open?" But to answer that, one first has to decide:* ***in what space are the tokens grouped?***

In this work, we investigate two structural failure modes that follow from clustering in default W-space or h-space: ❶ *static-geometry mismatch* where two W-rows (W_i & W_j) that look alike as weight vectors need not be selected by similar hidden states, so W-neighbours are frequently *not* the tokens that compete at decode time; ❷ *finite-sample obstacle* in estimating each row's hidden-state direction from calibration which suffers starvation due to the Zipfian token distribution (Section 3). To address these challenges, we propose **CHES** (**C**alibrated **H**idden-state **E**mbedding **S**election) that aggregates per-token hidden-state representatives via a streaming memory-bounded calibration pipeline and blends each representative toward its corresponding W-row prior using *direction-space shrinkage on the unit sphere*. We additionally introduce AVQ, an anisotropic within-cluster aggregation technique in standard direction-symmetric euclidean k-means to match asymmetric inner-product $\langle c_k, h \rangle$ based inference-time cluster scoring. Our key contributions are summarized as:

- **Diagnostic Foundation:** We formalize the *Vocabulary Tax* as a key inference bottleneck of large-vocabulary small models and empirically investigate the overlooked issues of static-geometry mismatch and long-tail calibration noise while grouping vocabulary into clusters.
- **Method Innovation:** We introduce CHES, a direction-space shrinkage framework for cluster-routed decoding that resolves issues associated with naive usage of W-space or h-space through blending them on unit sphere controlled by shrinkage strength (λ).
- **Theoretical Innovation:** We derive AVQ as a plug-n-play technique within conventional k-means euclidean residual minimization to align with the requirement of asymmetric residual orientation for inner-product at inference. We additionally show that CHES asymptotically reduces to TKCS and pure h-space clustering in the cold and hot-row limits.
- **Empirical Evidence:** Across three model families (Qwen, Gemma, Phi), we empirically establish that: ❶ CHES consistently improves Top-1 routing accuracy over W-space and h-space baselines at matched vocabulary coverage; ❷ AVQ provides further notable coverage-efficiency gains over CHES; ❸ CHES-AVQ can achieve nearly lossless performance across multiple downstream (*e.g.,* GSM8K, MATH-500) with merely 10-15% vocabulary usage.

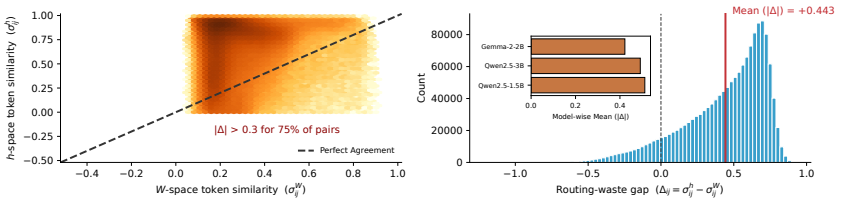


Figure 2: **Empirical Evidence of W-space vs h-space Geometric Mismatch (Qwen-1.5B).** (a) *Per-pair scatter of W-space similarity (σ_{ij}^W) against h-space similarity (σ_{ij}^h)*: We found the existence of a sharp divergence in which 75% of sampled pairs disagree by more than 0.3 cosine units with the mass concentrated in the *upper-left quadrant* where rows look unrelated in W-space yet share nearly identical hidden states. (b) *Distribution of the routing-waste gap ($\Delta_{ij} = \sigma_{ij}^h - \sigma_{ij}^W$)*: The distribution is strongly right-skewed with *aggregated mean* $|\Delta| = 0.443$ indicating significant mismatch between what TKCS optimises (W-similarity) and what routing requires (h-similarity).

2 Background and Notation

Output projection in LLMs: A causal LM with vocabulary $[V] = \{1, \ldots, V\}$ and hidden dimension d produces logits:

$$\ell_i = \langle W_i, h \rangle + b_i, \quad i \in [V], \tag{1}$$

where $W_i \in \mathbb{R}^d$ is the i-th row of $W \in \mathbb{R}^{V \times d}$. The argmax token is $i^\star(h) = \arg\max_i \ell_i$.

Cluster index: We define the cluster index Ω as a tuple:

$$\Omega = \Big(\{c_k\}_{k=1}^C, \{\mathcal{M}_k\}_{k=1}^C, \{N_k\}_{k=1}^C, \{B_k\}_{k=1}^C \Big), \tag{2}$$

where $c_k \in \mathbb{R}^d$ is the k-th centroid, $\{\mathcal{M}_k\}$ is a partition of $[V]$, and N_k, B_k are scalar bookkeeping derived from W and b:

$$\mathcal{M}_k \subseteq [V], \quad \bigsqcup_{k=1}^C \mathcal{M}_k = [V], \quad N_k = \max_{i \in \mathcal{M}_k} \|W_i\|, \quad B_k = \max_{i \in \mathcal{M}_k} b_i. \tag{3}$$

Cluster-routed Decode in LLMs: Given Ω and a budget $K \in [C]$, cluster-routed decode of h is:

$$s_k(h) = N_k \cdot \langle \hat{c}_k, h \rangle + B_k, \qquad \hat{c}_k = c_k / \|c_k\|, \tag{4}$$

$$\mathcal{T}_K(h) = \arg\text{top-}K_{k \in [C]} s_k(h), \qquad \mathcal{S}(h, K) = \bigcup_{k \in \mathcal{T}_K(h)} \mathcal{M}_k, \tag{5}$$

$$\tilde{\ell}_i(h) = \begin{cases} \langle W_i, h \rangle + b_i & i \in \mathcal{S}(h, K), \\ -\infty & i \notin \mathcal{S}(h, K). \end{cases} \tag{6}$$

The norm-max scoring bound: The score $\mathcal{S}(h, K)$ is motivated by the following inequality:

$$\max_{i \in \mathcal{M}_k} \langle W_i, h \rangle = \max_{i \in \mathcal{M}_k} \|W_i\| \cdot \langle \hat{W}_i, h \rangle \le N_k \cdot \max_{i \in \mathcal{M}_k} \langle \hat{W}_i, h \rangle, \tag{7}$$

which is tight when cluster members align in direction. The score replaces row-wise alignment with centroid alignment at $O(Cd)$ per token. A well-formed partition makes $s_k(h)$ a reliable substitute for true intra-cluster maximum, while a poorly formed opens clusters that do not contain the argmax.

Top-K Cluster Selection (TKCS) Method as Baseline: As per the cluster-routed decoding methods (Shim et al., 2017; Chen et al., 2019a; Liu et al., 2025; Zhang et al., 2025) introduced in Section 1, we take TKCS as our reference baseline throughout the paper. TKCS builds Ω by clustering the unit-normalised LM head rows $\hat{W}_i = W_i / \|W_i\|$ directly and solving k-means problem as:

$$\min_{c_1, \ldots, c_C,\ \pi : [V] \to [C]} \sum_{i=1}^V \big\| \hat{W}_i - c_{\pi(i)} \big\|^2 \tag{8}$$

via Lloyd's algorithm with k-means++ initialisation. The partition $\mathcal{M}_k = \{i : \pi(i) = k\}$ together with the bookkeeping (N_k, B_k) from Eq. 3 forms the cluster index. We would like to highlight that TKCS requires no calibration data and relies on a single hyperparameter C, it becomes the strongest offline baseline for comparison against our proposed technique.

3 INVESTIGATING WHEN W-SPACE DEFAULT IS NOT ENOUGH

In Section 1, we pose a question: *In what space are the tokens grouped?* Prior work TKCS $\{\hat{W}_i\}$ (Eq. equation 8) and its inheritors cluster directly with `LM-Head` rows W. We investigate Qwen2.5-1.5B checkpoint and identified three obstacles: ❶ a *geometric mismatch* between the space clustering optimises and the space in which routing decides (Section 3.1) ❷ *finite-sample obstacles* with Zipfian starvation of the warm tail and a norm-scale gap between h and W that block the naive h-space usage (Section 3.2).

3.1 THE STATIC-GEOMETRY MISMATCH PROBLEM

➤ *When two head rows are opened by the same cluster, are they the rows that competing hidden states would actually confuse?*

At inference, two LM head rows W_i and W_j end up in the same opened cluster if and only if their shared centroid c_k scores in the top-K for the current h. The relevant geometric question is therefore whether W_i and W_j tend to be selected as argmax by similar hidden states. On the other hand, clustering W directly answers a different question that whether W_i and W_j *look* similar in W-space. We formalize the gap as follows:

Definition 1 (h-space and W-space of row i)**.** *For a row $i \in V$, let $\mathcal{T}_i = \{t : i^\star(h_t) = i\}$ be its* hit-set*—the temporal positions at which i is the argmax. With row i, we associate two unit directions: (i) its W-*direction *($\hat{W}_i = W_i/\|W_i\|$); and (ii) its calibration-based h-*direction *($\mu_i^\star$) as mean direction of the unit hidden states $\{h_t/\|h_t\| : t \in \mathcal{T}_i\}$. The W-space and h-space similarity of a row pair (i, j) are defined as:*

$$\sigma_{ij}^W = \langle \hat{W}_i, \hat{W}_j \rangle \qquad \textit{and} \qquad \sigma_{ij}^h = \langle \mu_i^\star, \mu_j^\star \rangle.$$

TKCS groups rows with high σ_{ij}^W; correct routing requires grouping rows with high σ_{ij}^h. The per-pair *routing-waste gap* $\Delta_{ij} = \sigma_{ij}^h - \sigma_{ij}^W$ quantifies the discrepancy between what W-space clustering optimises and what routing requires.

★ **Empirical Finding:** Figure 2 measures this gap directly on Qwen2.5-1.5B. Plotting σ_{ij}^W against σ_{ij}^h for 500,000 sampled row pairs (Figure 2a), we found that the two similarities diverge notably from the diagonal where $\sim 75\%$ of sampled pairs disagree by more than 0.3 cosine units, with the mass sitting off the perfect-agreement line. The routing-waste gap is significantly large with an *aggregated mean* $|\Delta| = 0.443$ with a right-skewed distribution (Figure 2b). This indicates that many pairs are far more alike in h-space than in W-space, so any W-space partition can separate the rows that competing hidden states aligns. We found similar observation with a comparable mean gap across Qwen2.5-1.5B, Qwen2.5-3B, and Gemma-2-2B checkpoints, suggesting that optimization of W-space proximity is therefore misaligned with objective for routing.

3.2 THE CALIBRATION-NOISE AND NORM-MISMATCH PROBLEMS

➤ *If W-space is misaligned, why not simply cluster the hidden states that select each row?*

The natural fix is to cluster in the space where the routing decision actually resides by running calibration text through the model and, for each row i, average the hidden states that select it into $\bar{\mu}_i = \text{mean}\{h_t : t \in \mathcal{T}_i\}$, then cluster $\{\bar{\mu}_i\}$. Write $n_i = |\mathcal{T}_i|$ for the number of hidden states h_t (row hits) accumulated for row i; a row with $n_i = 0$ is *cold* (never selected during calibration). A naive implementation of this fix runs into two compounding problems shown in Figure 3.

Zipfian starvation of the warm tail: Token usage is Zipfian, so non-empty rows ($n_i > 0$) run from a few *hot* head of high-count rows into a long *warm* tail whose $\bar{\mu}_i$ are averaged over very few hits and stay noisy under a given calibration budget. The fluctuations drag k-means centroids around and because the noise is spread across the long warm tail it does not average out. The situation worsens for $n_i = 0$ with *cold* rows which fail to receive any hit (h_t) to participate in clustering. The median hit count over non-empty rows grows only sublinearly, reaching ~ 30 per row even at 30M tokens (Figure 3b) and scaling the budget therefore mainly enriches the already-hot set and can hurt top-1 accuracy as more rows activate with unreliable $\bar{\mu}_i$.

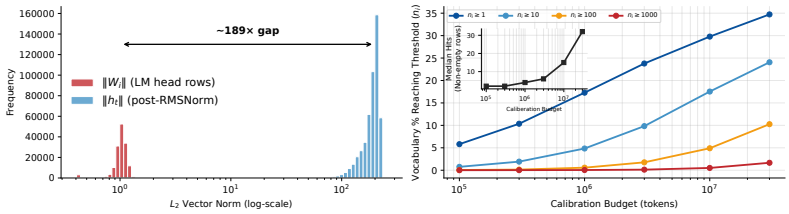


Figure 3: **Empirical Evidence of Calibration-Noise and Norm-Mismatch Obstacles (Qwen-1.5B).** (a) *Norm-scale gap*: $\|h_t\|$ (post-RMSNorm) and $\|W_i\|$ (LM head rows) remain at scales differing by $\sim$189×. (b) *Warm-tail growth vs. calibration budget:* Percentage of vocabulary reaching each hit-count threshold $n_i \in \{1,10,100,1000\}$ as budget scales from 100K to 30M tokens illustrating brute-forcing calibration data primarily benefits hot-set only. In addition, median hit count over non-empty rows grows sublinearly, reaching only 30 hits per row even at 30M tokens.

Norm mismatch between h-space and W-space: To address issue with cold rows, a naive solution can be falling back to W_i if $n_i = 0$ with appropriate regularization. However, regularising each noisy $\bar{\mu}_i$ toward its prior W_i is blocked by scale: the final RMSNorm (Zhang & Sennrich, 2019) pins $\|h_t\| \approx \sqrt{d}\,\gamma$ for a learned γ (on Qwen2.5-1.5B, we empirically found $\|h_t\| \in [100, 200]$), whereas $\|W_i\|$ lives far below ($\sim 189\times$) than h_t (Figure 3a). A naive blend estimated as:

$$\mu_i^{\text{naive}} = \frac{n_i}{n_i + \lambda}\,\bar{\mu}_i + \frac{\lambda}{n_i + \lambda}\,W_i \tag{9}$$

suffers because even at moderate λ the $\bar{\mu}_i$ term outweighs W_i by that same $\sim 189\times$, leaving W_i as rounding error rather than a precision-weighted prior. Pre-rescaling W_i to h-scale only injects an arbitrary scalar into the geometry and its noise can leak into the cluster structure. CHES (Section 4) resolves all three at once by performing the estimate-and-regularise step on the unit sphere.

4 CHES: Clustering by Calibrated Hidden-State Geometry

In section 3, we argued that the routing decision lives in h-space and investigated the challenges in estimating that geometry directly. Motivated by our findings, our proposed method CHES studies which hidden states h_t select which row within the `LM-Head` and proposes a four-stage *offline* pipeline: ❶ streaming calibration with memory-bounded incremental aggregation (Section 4.1); ❷ direction-space shrinkage that builds row-wise representatives μ_i with robustness to low calibration counts (Section 4.2); ❸ spherical k-means on $\{\mu_i\}$ to form the cluster index metadata (Section 4.3); ❹ a plug-n-play anisotropic refinement (AVQ) of within-cluster aggregation that sharpens centroids for inner-product routing at inference. Note that CHES is offline and doesn't influence inference pipeline from Section 2, so decode cost still remains $O(Cd+|\mathcal{S}|d)$ similar to TKCS baseline.

4.1 Stage 1: Streaming Calibration

In this stage, we pass the calibration corpus $\mathcal{D}$ through the model backbone (transformer layers plus final norm, excluding the `LM-Head`) to obtain a stream of hidden states $\{h_t\}_{t=1}^{T}$ and their full-vocabulary argmaxes as follows:

$$i_t = \arg\max_{i \in V} \langle W_i, h_t \rangle + b_i. \tag{10}$$

For each row $i \in V$, let $\mathcal{T}_i = \{t : i_t = i\}$ collect the positions where row i won, with hit count $n_i = |\mathcal{T}_i|$. We define the aggregation of hidden states that choose i as:

$$S_i = \sum_{t \in \mathcal{T}_i} h_t \in \mathbb{R}^d, \qquad \bar{\mu}_i = \frac{S_i}{n_i} \ \text{(when } n_i > 0\text{)}. \tag{11}$$

Here, a naive implementation which accumulates every (h_t, i_t) pair and forms $\bar{\mu}_i$ in a second pass will cost $O(T \cdot d)$ memory and enforces a permitted calibration budget based on whatever fits in the device RAM. We would like to highlight that streaming removes such restriction where we store $S \in \mathbb{R}^{V \times d}$ and $n \in \mathbb{N}^V$ in RAM bounded by $O(V \cdot d)$ and independent of T. In Figure 3, we empirically show that a Qwen-1.5B model with 152K-row vocabulary needs tens of millions of tokens before the warm Zipfian tail can win any hidden state. Detailed pseudocode in Appendix **??**.

4.2 STAGE 2: DIRECTION-SPACE SHRINKAGE

In section 3.2, we discussed two issues: warm-tail rows carrying noisy sample means $\bar{\mu}_i$ (Zipfian starvation), and significantly high ($\sim 189\times$ for Qwen-1.5B) difference in norm between $\bar{\mu}_i$ and W_i causing a naive convex blend (Equation 9) to collapse into $\bar{\mu}_i$ with W_i as negligible rounding error. We ask: *How do we blend W-direction prior into a noisy h-space estimate when the two remain at incompatible scales?* We propose to blend in unit sphere, where the two h-space and W-space are pure directions and the mixing weight remain meaningful. We define the hot-row ($n_i > 0$) and `LM-Head`-row unit directions as follows:

$$\hat{\bar{\mu}}_i = \frac{S_i}{\|S_i\|} \;\; (n_i > 0), \qquad \hat{W}_i = \frac{W_i}{\|W_i\|}, \tag{12}$$

and, we blend $\hat{\bar{\mu}}_i$ and $\hat{W}_i$ with a tunable hyperparameter *shrinkage strength* ($\lambda_{shrink} \geq 0$) as:

$$\mu_i = \frac{n_i \hat{\bar{\mu}}_i + \lambda_{shrink} \hat{W}_i}{\left\| n_i \hat{\bar{\mu}}_i + \lambda_{shrink} \hat{W}_i \right\|}. \tag{13}$$

Since both $\hat{\bar{\mu}}_i$ and $\hat{W}_i$ are unit vectors, the blend can be interpreted as a precision-weighted average of *directions* such that the W-space prior have weight at row i as $w_W(i) = \lambda_{shrink}/(n_i + \lambda_{shrink})$, which is ≈ 1 for cold rows ($n_i = 0$) and $\to 0$ as $n_i \to \infty$. Note that by definition, $\hat{\bar{\mu}}_i = 0$ for cold rows, and Eq. equation 13 reduces cleanly to $\mu_i = \hat{W}_i$ (degrades to TKCS signal) on rows which couldn't accumulate any calibration signal.

4.3 STAGE 3: SPHERICAL CLUSTERING AND BOOKKEEPING

CHES perform clustering on the row-wise unit-norm representatives (μ_i) directly:

$$\min_{c_1,\ldots,c_C,\,\pi} \sum_{i=1}^{V} \left\| \mu_i - c_{\pi(i)} \right\|^2, \qquad \|\mu_i\| = 1, \tag{14}$$

which is a spherical k-means by construction, since the inputs lie on $\mathbb{S}^{d-1}$. We use Lloyd's algorithm with k-means++ initialization (Arthur et al., 2007) with empty-cluster reseeding with size cap $|\mathcal{M}_k| \leq 3V/C$ induced by recursively splitting an oversized cluster. The cap prevents the abnormal imbalance that would otherwise inflate the worst-case sub-vocabulary size $|\mathcal{S}|$ at fixed K. Note that the metadata bookkeeping Ω have exactly the same overhead as in TKCS and computed from W and b, *not* from μ. The inner-product based inference-time logit $\langle W_i, h \rangle + b_i$ lives in W-space, and the cluster-scoring bound of Equation 7 is based on W-row alignment. In simple words, CHES decides *which* rows are grouped while TKCS-style bookkeeping decides *how* the resulting groups are ranked.

4.4 STAGE 4: PLUG-N-PLAY AVQ(ANISOTROPIC WITHIN-CLUSTER AGGREGATION)

The Lloyd update in Equation 14 minimizes a direction-symmetric cost $\sum_{i \in \mathcal{M}_k} \|\mu_i - c\|^2$ where a residual aligned with μ_i costs exactly as much as one orthogonal to it. In this section, we ask: *Does the Lloyd update minimizer that aligns well for l_2-based retrieval, also produces right centroid for inner-product routing?*

Cluster-decoding method's inference-time cluster score $s_k(h) = N_k \langle \hat{c}_k, h \rangle + B_k$ is an inner product, and inner-product retrieval is asymmetric in residual orientation. The error in the centroid *along* the data direction degrades the score directly, whereas error *orthogonal* to it is invisible to a query aligned with that direction (Guo et al., 2020). A cluster centroid that aligns well with all members in direction routes hidden states during inference correctly to that cluster, while a centroid that is pulled away from cluster members along their own directions misses on routing. The arithmetic mean therefore optimises the wrong notion of closeness for routing. Our proposed plug-n-play module AVQ corrects this by penalizing more for parallel residual than for orthogonal.

Anisotropic Objective: Let $\lambda_{\text{aniso}} \geq 0$ be an anisotropy strength. For the unit-norm inputs produced by Stage 2, we define the per-pair loss as:

$$L_{\text{AVQ}}(\mu_i, c_k) = \|\mu_i - c_k\|^2 + \lambda_{\text{aniso}} \langle \mu_i - c_k, \mu_i \rangle^2. \tag{15}$$

The new second term penalizes the projection of residual $(\mu_i - c_k)$ onto μ_i's own direction and leaves the orthogonal cost unchanged. At $\lambda_{\text{aniso}} = 0$, it vanishes and L_{AVQ} reduces to the conventional k-means euclidean loss of Equation 14.

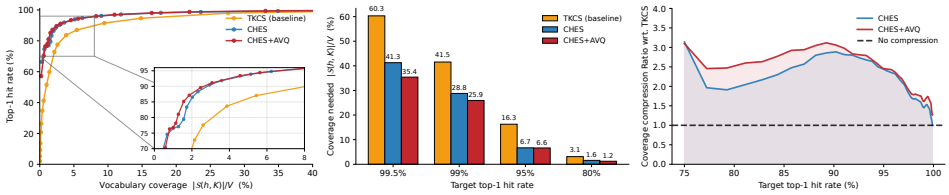


Figure 4: **Coverage–Efficiency of CHES and CHES+AVQ against the TKCS baseline (Qwen2.5-1.5B).** (a) *Hit rate vs. vocabulary coverage*: Top-1 routing hit rate as a function of the used vocabulary fraction $|S(h,K)|/V$ where both CHES and CHES+AVQ dominate TKCS reaching significantly high hit rates at a fraction of the coverage especially in the low-coverage regime. (b) *Coverage needed to reach a target accuracy*: To achieve 99.5% Top-1 Hit Rate, CHES+AVQ requires only 35.4% of the vocabulary versus 60.3% for TKCS. (c) *Compression ratio wrt.* TKCS: Coverage compression consistently stay above full `LM-Head` baseline with CHES+AVQ sustaining largest savings.

Closed-form update. For unit-norm inputs and a fixed partition π, the per-cluster objective $\mathcal{L}_k(c) = \sum_{i\in\mathcal{M}_k} \mathcal{L}_{\text{AVQ}}(\mu_i, c)$ is strictly convex, and its unique minimiser is the solution of the linear system given as:

$$\underbrace{\left(n_k\, I_d + \lambda_{\text{aniso}} \sum_{i\in\mathcal{M}_k} \mu_i \mu_i^\top\right)}_{A_k} c_k \;=\; \underbrace{(1+\lambda_{\text{aniso}}) \sum_{i\in\mathcal{M}_k} \mu_i}_{b_k}, \qquad c_k = A_k^{-1} b_k, \tag{16}$$

with $n_k = |\mathcal{M}_k|$ (full derivation in Appendix C.1). A_k is symmetric positive definite (a positive multiple of the identity plus a symmetric positive semi-definite outer-product sum) and thereby there exist a unique solution. At λ_{aniso}=0 it collapses to $c_k = \frac{1}{n_k}\sum_{i\in\mathcal{M}_k} \mu_i$, confirming that AVQ is a strict generalisation of the standard update.

Geometric Interpretation of AVQ: If we decompose any centroid $c = c_\parallel + c_\perp$ into components parallel and orthogonal to μ_i, since $\|\mu_i\|$=1 we will have $\left(I_d + \lambda_{\text{aniso}} \mu_i \mu_i^\top\right) c = (1+\lambda_{\text{aniso}})\, c_\parallel + c_\perp$. AVQ thus acts as an *anisotropic spring* between c and each μ_i stretching the parallel part by $(1+\lambda_{\text{aniso}})$ and leaving the orthogonal part untouched. In other words, it enables $(1+\lambda_{\text{aniso}})\times$ stiffness along each member's own direction than across it, which tilts the centroid toward the minority-direction members that the arithmetic mean would diminish. Hidden states aligned with those rare members can then score higher against the cluster and route correctly. The gain from AVQ plugged within CHES should therefore be largest in the high-accuracy regime, where the remaining routing errors are dominated by long-tail tokens and we will confirm this in Section 5.

Cost Overhead of AVQ: Solving $A_k c_k = b_k$ as a $d \times d$ system per cluster per Lloyd iteration would be computationally expensive. Note that in Equation 16, A_k deviates from a scaled identity by a matrix of rank at most $n_k \ll d$. In our case, cluster sizes are small ($\bar{n}_k = V/C$), so we can use Sherman-Morrison-Woodbury identity to reduce each update to an $n_k \times n_k$ solve. Hence, the per-iteration overhead becomes a fraction of the naive $O(VCd)$ cost. Appendix C.2 gives the gradient derivation, the Woodbury form, and a reference implementation.

5 Experiments and Ablation Studies

5.1 Performance Benchmarking of CHES and the W-space Baseline

➤ **Coverage-vs-Accuracy Curves.** Figure 4 reports top-1 hit rate against coverage on Qwen2.5-1.5B with 100K calibration tokens. To reach a 99% hit rate, TKCS opens 41.5% of the vocabulary, CHES 28.8% (1.44× compression), and CHES+AVQ 25.9% (1.60×). The margin widens in the mid-coverage regime that reflects practical edge budgets where at 95% target CHES+AVQ needs 6.6% coverage against 16.3% for TKCS (2.47×), and at 80% needs 1.2% against 3.1% (2.58×) for TKCS.

➤ **Cluster-size Distributions and the high-accuracy tail.** In comparison to TKCS, CHES also produces far more uniform partitions. At a matched mean cluster size, the size standard deviation falls from 355 for TKCS to 194 for CHES and 171 with AVQ (Figure 7a). Since each opened cluster adds $|\mathcal{M}_k|$ rows to $\mathcal{S}(h,K)$, the balanced cluster distribution keeps coverage and head cost proportional to K, making per-token latency predictable. It avoids TKCS's unpredictable opening of very large clusters, which can spend much of the budget on rows unrelated to the current hidden state h.

Task	Method	Qwen2.5-0.5B $\lvert\mathcal{S}\rvert/V{=}0.05$	Qwen2.5-0.5B $\lvert\mathcal{S}\rvert/V{=}0.10$	Qwen2.5-0.5B $\lvert\mathcal{S}\rvert/V{=}0.20$	Qwen2.5-1.5B $\lvert\mathcal{S}\rvert/V{=}0.05$	Qwen2.5-1.5B $\lvert\mathcal{S}\rvert/V{=}0.10$	Qwen2.5-1.5B $\lvert\mathcal{S}\rvert/V{=}0.20$	Gemma2-2B $\lvert\mathcal{S}\rvert/V{=}0.05$	Gemma2-2B $\lvert\mathcal{S}\rvert/V{=}0.10$	Gemma2-2B $\lvert\mathcal{S}\rvert/V{=}0.20$
GSM8K	Baseline		33.5			61.0			23.7	
	TKCS	31.9	32.1	32.4	39.5	55.6	58.2	19.1	21.5	22.9
	CSV-Decode (Liu et al., 2025)	32.7	32.8	33.0	41.2	57.3	59.4	20.2	22.4	23.4
	CHES [Ours]	32.0	33.1	33.4	59.9	60.7	60.6	21.6	22.8	23.3
	CHES+AVQ [Ours]	32.0	32.8	33.5	59.5	60.7	61.0	22.7	23.4	23.6
MATH	Baseline		10.1			27.4			15.2	
	TKCS	3.2	3.5	3.8	12.5	17.8	20.3	11.7	11.9	13.4
	CSV-Decode (Liu et al., 2025)	4.0	4.2	4.4	14.0	19.0	21.7	12.6	12.6	14.2
	CHES [Ours]	4.6	4.7	5.3	0.0	2.5	27.3	0.0	4.4	14.8
	CHES+AVQ [Ours]	4.4	9.8	10.0	0.0	19.0	27.3	0.0	14.9	15.2
MMLU	Baseline		47.9			59.7			51.2	
	TKCS	47.9	47.9	47.9	59.7	59.7	59.7	51.2	51.2	51.2
	CSV-Decode (Liu et al., 2025)	47.9	47.9	47.9	59.7	59.7	59.7	51.2	51.2	51.2
	CHES [Ours]	47.9	47.9	47.9	59.7	59.7	59.7	51.2	51.2	51.2
	CHES+AVQ [Ours]	47.9	47.9	47.9	59.7	59.7	59.7	51.2	51.2	51.2
TriviaQA	Baseline		12.3			29.3			50.6	
	TKCS	8.8	10.1	10.4	17.8	22.7	24.2	35.9	43.1	44.9
	CSV-Decode (Liu et al., 2025)	9.6	10.8	11.1	19.2	24.3	25.6	37.1	44.6	46.3
	CHES [Ours]	10.7	11.4	11.8	26.5	28.2	28.5	41.8	45.4	48.5
	CHES+AVQ [Ours]	10.6	11.4	12.0	26.5	28.0	29.3	42.5	46.6	48.8
NQ-Open	Baseline		4.7			7.8			15.7	
	TKCS	3.4	3.9	4.1	4.8	6.2	6.5	11.9	13.2	13.7
	CSV-Decode (Liu et al., 2025)	3.8	4.3	4.5	5.3	6.8	7.0	12.6	13.9	14.3
	CHES [Ours]	4.3	4.4	4.5	7.1	7.5	7.6	13.7	14.5	15.0
	CHES+AVQ [Ours]	4.3	4.3	4.6	7.2	7.7	7.8	13.7	14.6	15.7
HumanEval	Baseline		29.9			36.6			20.1	
	TKCS	17.7	21.3	22.6	20.7	27.4	30.2	10.4	11.0	11.9
	CSV-Decode (Liu et al., 2025)	19.5	23.2	24.4	22.6	29.3	31.7	12.2	12.8	13.4
	CHES [Ours]	15.9	22.6	25.6	20.1	29.3	32.9	15.9	15.9	17.1
	CHES+AVQ [Ours]	18.3	20.7	26.8	20.1	27.4	35.1	11.0	11.6	19.4
MBPP	Baseline		29.8			44.6			27.4	
	TKCS	18.2	21.6	22.1	21.8	31.9	34.4	7.4	12.8	14.8
	CSV-Decode (Liu et al., 2025)	20.0	23.2	23.4	23.4	33.8	36.2	8.6	14.2	16.2
	CHES [Ours]	17.0	23.0	26.4	25.8	36.4	38.4	0.2	0.2	20.6
	CHES+AVQ [Ours]	18.8	24.6	26.8	32.4	40.8	44.0	0.0	21.0	23.4

Table 1: Accuracy across 7 benchmarks and 3 model scales at pruned vocabulary fractions $|\mathcal{S}|/V \in \{5, 10, 20\}$. CSV-Decode (Liu et al., 2025) is the strongest prior dynamic sub-vocabulary baseline.

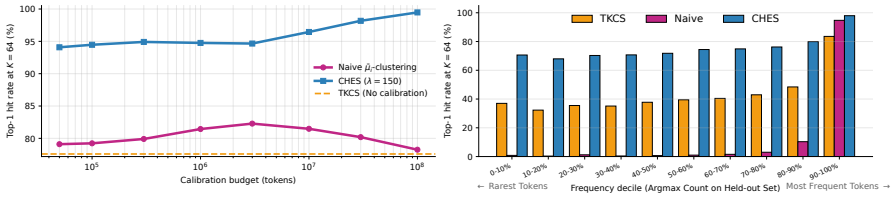


Figure 5: **CHES turns calibration budget into routing accuracy where naive aggregation fails (Qwen2.5-1.5B).** (a) *Top-1 hit rate vs. calibration budget ($K{=}64$)*: CHES improves monotonically as the budget scales from 100K to 100M tokens whereas naive $\bar{\mu}_i$-clustering regresses with increase in calibration tokens. (b) *Hit rate by token-frequency decile ($K{=}64$)*: Split by frequency decile estimated using held-out set, CHES benefits Top-1 Hit-rate across all deciles with sharpest performance on rarest tokens (left) where TKCS and naive aggregation are weakest.

➤ **Calibration budget on Qwen2.5-1.5B.** Direction-space shrinkage prevents the hit-rate collapse with growing calibration budget anticipated in Figure 3b. CHES ($\lambda_{\text{shrink}} = 150$) improves monotonically as the budget scales from 100K to 100M tokens, whereas naive $\bar{\mu}_i$-clustering regresses as more warm-tail rows are activated with unreliable estimates (Figure 5a). Even at 100K tokens, CHES exceeds naive aggregation at every budget up to 100M. Note that advantage is sharpest on the rarest-token deciles (Appendix B), exactly where TKCS and naive aggregation are weakest (Figure 5b).

➤ **Wall-clock decode latency on Hardware.** We investigate how does the coverage savings translate into per-token speedups in (Figure 6). On Apple M4 (MLX GPU, $B = 1$, $K = 64$ of 512), the head's per-token time drops by 1.6 to 3.2×, for example from 12.9 to 4.06 ms on Gemma-2-2B. On an Intel Xeon 8559C at 16 threads it drops 3.2× on Qwen2.5-1.5B and 2.4× on Qwen2.5-0.5B, but only 1.3× on Gemma-2-2B. Speedup grows as fewer clusters are opened, reaching ∼40× at the extreme, and holds between 2 and 4× in the low-thread edge regime.

➤ **Downstream task performance preservation.**[1] Table 1 investigates whether gains in routing accuracy influence on autoregressive generation, where a single missed argmax can hurt each subsequent token. At a 20% budget, CHES+AVQ matches or exceeds both baselines on every model and task, and closes most of the gap to the full head, with a mean deficit of 0.9 points compared with 4.9 for TKCS and 3.9 for CSV-Decode. We found that degradation concentrates on code generation, whose indentation and identifier tokens are rarely selected during calibration. The advantage is most pronounced where TKCS is most fragile. On Qwen2.5-1.5B at a 5% budget, CHES+AVQ retains 59.5 on GSM8K and 26.5 on TriviaQA (full head 61.0 and 29.3), whereas TKCS drops to 39.5 and 17.8.

[1]MMLU is scored by log-likelihood over fixed options (A/B/C/D) which are preserved and hence identical for every method and budget.

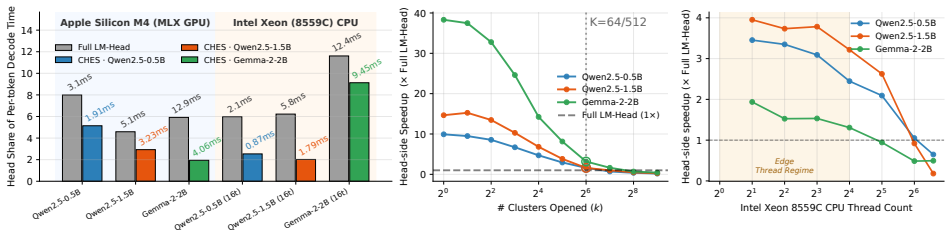


Figure 6: **Measured `LM-Head` Latency & Speedup:** (a) Per-token `LM-Head` decode time on Apple M4 (MLX GPU) and Intel CPU, swapping the Full `LM-Head` for CHES decoding (K=64/512) significantly reduces the head's share of per-token decode cost. (b) *Speedup vs. Clusters opened*: CHES speedup over the Full `LM-Head` rises sharply as fewer clusters K are opened (up to $\sim$40$\times$ in the extreme). (c) *Speedup vs. CPU thread count*: CHES can provide 2-4$\times$ speedup is lin the low-thread edge inference regime with highest vocabulary tax.

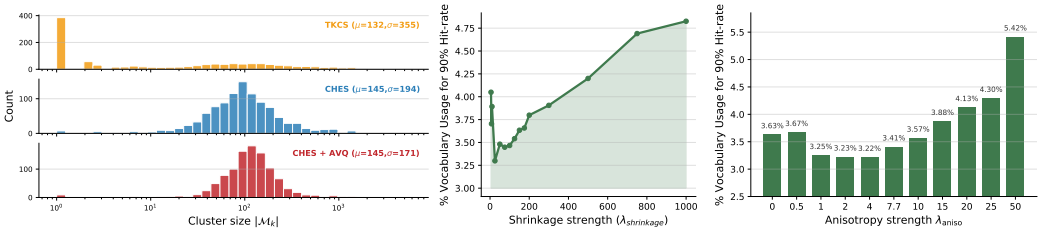


Figure 7: **Cluster balance and sensitivity (Qwen2.5-1.5B).** (a) *Cluster-size distributions* at matched mean size: CHES (σ=194) and CHES+AVQ (σ=171) are far more balanced than TKCS (σ=355). (b) *Shrinkage strength*: Total Vocabulary usage at a 90% hit rate is minimised near $\lambda_{\text{shrink}}\approx$75–125. (c) *Anisotropy*: Total Vocabulary usage is minimised at λ_{aniso}=4 (3.22%).

Consistent with the analysis in Section 4.4, AVQ contributes most when generation depends on long-tail tokens. At a 10% budget it improves MATH from 2.5 to 19.0 on Qwen2.5-1.5B and from 4.4 to 14.9 on Gemma-2-2B, leaving GSM8K, TriviaQA unchanged.

5.2 A Deeper Investigation of Shrinkage and Anisotropy

➤ **Sensitivity to λ_{shrink}.** Sweeping λ_{shrink} from 0 to ∞ isolates the weighting between calibration evidence and the W-row prior (Figure 7b). As $\lambda_{\text{shrink}} \to \infty$ the token representative collapses onto the W-row prior and CHES reduces to TKCS, while at $\lambda_{\text{shrink}} = 0$ (naive $\bar{\mu}_i$-clustering) the coverage curve becomes unstable and degrades on the warm tail. Vocabulary usage for a 90% top-1 hit rate shows a U-shaped curve that is minimised between 75 and 125 and rises gradually beyond it. In our work, all presented results use $\lambda_{\text{shrink}} = 150$, close to this minimum.

➤ **Sensitivity to λ_{aniso}.** Sweeping λ_{aniso} from 0 to 50 at fixed $\lambda_{\text{shrink}} = 150$ investigates the contribution of AVQ (Figure 7c) in our work. Setting $\lambda_{\text{aniso}} = 0$ reproduces CHES alone, and we found that every value between 1 and 7 improves CHES. Vocabulary usage for a 90% hit rate is again U-shaped and reaches its minimum of 3.22% at $\lambda_{\text{aniso}} = 4$. Too little anisotropy fails to correct the inner-product mismatch, whereas too much over-stiffens the centroid and degrades coverage, raising usage to 5.42% at $\lambda_{\text{aniso}} = 50$. We therefore adopt $\lambda_{\text{aniso}} = 4$ as the default, which weights parallel residuals five times more heavily than orthogonal ones in the per-pair loss.

6 Conclusion

In this work, we propose CHES, a training-free clustering method for cluster-routed decoding that addresses two empirical failure modes of grouping the vocabulary in weight space, namely the static-geometry mismatch between LM head rows and the hidden states that select them, and the calibration starvation of rare tokens under a Zipfian distribution. CHES represents each token by a count-adaptive shrinkage estimate on the unit sphere that interpolates between the mean direction of its selecting hidden states and its LM head row. We further introduce AVQ, a closed-form anisotropic centroid update that aligns clustering with inference-time inner-product routing. We theoretically motivate and empirically establish the benefits of CHES+AVQ in reducing the vocabulary tax of large-vocabulary small models while closely matching the performance with full LM-Head decoding.

AI USE STATEMENT

In this work, we did not use generative AI tools for any task that requires disclosure. We have not used generative AI tools to generate synthetic datasets, develop theoretical models, or propose and refine hypotheses, or interpret results. We used generative AI tools only to edit the paper for readability and grammatical correctness. We have reviewed all AI-assisted edits and verified that they do not alter the technical content, claims, or reported results. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Amine Abdaoui, Camille Pradel, and Grégoire Sigel. Load what you need: Smaller versions of mutililingual BERT. In Nafise Sadat Moosavi, Angela Fan, Vered Shwartz, Goran Glavaš, Shafiq Joty, Alex Wang, and Thomas Wolf (eds.), *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pp. 119–123, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.sustainlp-1.16. URL https://aclanthology.org/2020.sustainlp-1.16/.

David Arthur, Sergei Vassilvitskii, et al. k-means++: The advantages of careful seeding. In *Soda*, volume 7, pp. 1027–1035, 2007.

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 Shabnam Tafreshi, Arjun Akula, João Sedoc, Aleksandr Drozd, Anna Rogers, and Anna Rumshisky (eds.), *Proceedings of the Fifth Workshop on Insights from Negative Results in NLP*, pp. 148–153, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.insights-1.17. URL https://aclanthology.org/2024.insights-1.17/.

Chameleon. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*, 2024.

Patrick Chen, Si Si, Sanjiv Kumar, Yang Li, and Cho-Jui Hsieh. Learning to screen for fast softmax inference on large vocabulary neural networks. In *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=ByeMB3Act7.

Patrick Chen, Si Si, Sanjiv Kumar, Yang Li, and Cho-Jui Hsieh. Learning to screen for fast softmax inference on large vocabulary neural networks. In *ICLR*, 2019b.

Raghavv 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.

Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. Efficient softmax approximation for GPUs. In *International Conference on Machine Learning (ICML)*, 2017.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*, pp. 3887–3896. PMLR, 2020.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2015.

Dong Liu, Shu Wang, Yanxuan Yu, Haisheng Wang, and Ben Lengerich. Csv-decode: Certifiable sub-vocabulary decoding for efficient large language model inference. *arXiv preprint arXiv:2511.21702*, 2025.

Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In Robert G. Cowell and Zoubin Ghahramani (eds.), *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, volume R5 of *Proceedings of Machine Learning Research*, pp. 246–252. PMLR, 06–08 Jan 2005. URL https://proceedings.mlr.press/r5/morin05a.html. Reissued by PMLR on 30 March 2021.

Xing Shi and Kevin Knight. Speeding up neural machine translation decoding by shrinking runtime vocabulary. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 574–579, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-2091. URL https://aclanthology.org/P17-2091/.

Kyuhong Shim, Minjae Lee, Iksoo Choi, Yoonho Boo, and Wonyong Sung. Svd-softmax: Fast softmax approximation on large vocabulary neural networks. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Quan Sun, Qiying Yu, Yufeng Cui, Fan Zhang, Xiaosong Zhang, Yueze Wang, Hongcheng Gao, Jingjing Liu, Tiejun Huang, and Xinlong Wang. Emu: Generative pretraining in multimodality. In *International conference on learning representations*, volume 2024, pp. 12352–12380, 2024.

Chaofan Tao, Qian Liu, Longxu Dou, Niklas Muennighoff, Zhongwei Wan, Ping Luo, Min Lin, and Ngai Wong. Scaling laws with vocabulary: Larger models deserve larger vocabularies. *Advances in Neural Information Processing Systems*, 37:114147–114179, 2024.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Asahi Ushio, Yi Zhou, and Jose Camacho-Collados. Efficient multilingual language model compression through vocabulary trimming. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 14725–14739, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.981. URL https://aclanthology.org/2023.findings-emnlp.981/.

Chengyue Wu, Xiaokang Chen, Zhiyu Wu, Yiyang Ma, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, Chong Ruan, et al. Janus: Decoupling visual encoding for unified multimodal understanding and generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12966–12977. IEEE, 2025a.

Yecheng Wu, Zhuoyang Zhang, Junyu Chen, Haotian Tang, Dacheng Li, Yunhao Fang, Ligeng Zhu, Enze Xie, Hongxu Yin, Li Yi, et al. Vila-u: a unified foundation model integrating visual understanding and generation. In *International Conference on Learning Representations*, volume 2025, pp. 93620–93638, 2025b.

Jin Xu, Zhifang Guo, Hangrui Hu, Yunfei Chu, Xiong Wang, Jinzheng He, Yuxuan Wang, Xian Shi, Ting He, Xinfa Zhu, et al. Qwen3-omni technical report. *arXiv preprint arXiv:2509.17765*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

Jinbin Zhang, Nasib Ullah, Erik Schultheis, and Rohit Babbar. Dynaspec: Context-aware dynamic speculative sampling for large-vocabulary language models. *arXiv preprint arXiv:2510.13847*, 2025.

Minjia Zhang, Xiaodong Liu, Wenhan Wang, Jianfeng Gao, and Yuxiong He. Navigating with graph representations for fast and scalable decoding of neural language models. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 6311–6322, 2018.

Weilin Zhao, Tengyu Pan, Xu Han, Yudi Zhang, Ao Sun, Yuxiang Huang, Kaihuo Zhang, Weilun Zhao, Yuxuan Li, Jie Zhou, Hao Zhou, Jianyong Wang, Zhiyuan Liu, and Maosong Sun. FR-spec: Accelerating large-vocabulary language models via frequency-ranked speculative sampling. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3909–3921, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.198. URL https://aclanthology.org/2025.acl-long.198/.

A Related Works

Reducing the cost of the output projection has long been studied in neural language modeling. The problem has regained importance as small LLMs adopt large amount of vocabularies from 150K to 300K tokens (Tao et al., 2024). Existing approaches differ along two axes: whether the candidate set adapts to the current hidden state or is fixed in advance and whether the method requires training or architectural changes. Dynamic sub-vocabulary decoding for LLMs (Section A.1) lies closest to our work. It selects candidates for each hidden state and leaves the model weights intact. Most of methods in this line require no training. Classical output-layer acceleration (Section A.2) spans both axes. Hierarchical and adaptive softmax fix a structured output layer at training time, whereas screening and search methods adapt the candidate set to each query, some of them through learned components. Static vocabulary pruning (Section A.3) lies at the opposite end. It fixes a reduced vocabulary in advance and permanently removes the remaining rows, which often calls for fine-tuning to recover quality. CHES thus occupies the query-dependent, post hoc region of this design space, alongside prior dynamic sub-vocabulary methods. Its contribution is orthogonal to how clusters are routed or certified: it revisits the space in which tokens are grouped where prior methods implicitly fix to the geometry of the LM-head rows.

A.1 Dynamic Sub-vocabulary Decoding

The most common form of query-dependent decoding, which we refer to as TKCS, clusters the unit-normalized LM-head rows with k-means offline. At each decoding step, it computes exact logits only within the top-K clusters whose centroids score highest against the current hidden state (Section 2). CSV-Decode (Liu et al., 2025) clusters the output embeddings and uses centroid plus radius upper bounds on the logits within each cluster to skip clusters that provably cannot affect the top-k set, falling back to full-vocabulary computation when the certified sub-vocabulary grows too large. DynaSpec (Zhang et al., 2025) trains a lightweight meta-classifier that routes each context to a small set of token clusters and uses their union as the drafter's shortlist, while the target model still verifies drafts over the full vocabulary, so that the output distribution is preserved. These methods refine how clusters are sorted, certified, or selected, but the grouping itself is typically derived from the geometry of the LM-head rows. Our diagnostic in Section 3 shows that similarity between LM-head rows is a poor predictor of similarity between the hidden states that select them. CHES therefore leaves the inference path of TKCS unchanged and revisits the grouping instead. It clusters tokens by the hidden states that select them, using the LM-head rows only as a prior where calibration evidence is scarce.

A.2 Output Layer Acceleration

Hierarchical softmax (Morin & Bengio, 2005) factorizes the output distribution along a tree over the vocabulary. Adaptive softmax (Grave et al., 2017) groups tokens by frequency into a head and several reduced-dimension tail clusters. Both replace the softmax with a different output model that must be trained with it, so applying them to a pre-trained checkpoint requires retraining and changes its output distribution. Screening and search methods instead accelerate a pre-trained output layer at inference time, adapting the candidate set to each query. SVD-Softmax (Shim et al., 2017) previews all logits using the leading singular directions of the output matrix and recomputes exact logits only for the top-scoring candidates. Graph-based search Zhang et al. (2018) casts top-k decoding as maximum inner product search over an index built from the LM-head rows alone, without regard to the distribution of decoder hidden states. L2S (Chen et al., 2019b) is closest to CHES since it also lets the distribution of hidden states shape the candidates sets. It clusters context vectors and learns the cluster assignment with a gumbel-softmax relaxation (Jang et al., 2017) and a candidate set for each cluster. CHES differs in three aspects. First, it partitions the vocabulary rather than the context space, so each token belongs to exactly one cluster and the cluster-routed inference path is unchanged. Second, it computes token representatives in closed form from streamed calibration statistics rather than through gradient-based training. Third, it explicitly handles rare tokens through count-adaptive shrinkage toward the LM-head prior, which becomes critical at large vocabulary scales. In neural machine translation, decoding was commonly restricted to candidate lists predicted from the source sentence (Jean et al., 2015; Shi & Knight, 2017). Lacking such a source signal, decoder only LLMs must select candidates from the hidden state at every step.

A.3 Static Vocabulary Pruning

Static pruning typically selects tokens by their frequency in a reference corpus, for example to derive smaller versions of multilingual models restricted to target languages (Abdaoui et al., 2020; Ushio et al., 2023). Bogoychev et al. (2024) apply such heuristics to LLM decoding and find that the outputs can diverge from full-vocabulary decoding, especially for non-Latin scripts and code-mixed text. In speculative decoding, the same idea is applied to the drafter, restricting its LM-head to a fixed, frequency-ranked subset of the vocabulary (Zhao et al., 2025; Goel et al., 2025). Since the target model still verifies over the full vocabulary, the output distribution is preserved and the cost appears only as a lower acceptance rate.

Applied to the target model itself, static pruning offers no such safeguard. Removed tokens become permanently unreachable and output quality depends on how well the reference corpus represents the deployment distribution. CHES removes no rows. Rows never selected during calibration fall back to their LM-head direction and still participate in clustering and routing. Our frequency-decile analysis (Figure 5 (b)) shows that CHES gains the most over TKCS in the lowest-frequency deciles which are the rows that frequency-based pruning is most likely to discard.

B Details of Frequency Bins for Figure 7(c)

The Nemotron-CC-v2 High-Quality subset is streamed through Qwen2.5-1.5B with the Qwen2.5 BPE tokenizer ($|\mathcal{V}| = 151{,}936$, sequence length $T = 1{,}024$), producing a calibration stream of $\sim$20M tokens over $\sim$19,500 sequences. Corpus token counts are computed on this stream where tokens with zero observed frequency remain in the analysis because static head pruning is a *per-row* decision on the output matrix $W \in \mathbb{R}^{V \times d}$ and every row either observed or not remains a candidate for removal.

Table 2: Frequency decile bin reference for Figure 7(c). Nemotron-CC-v2 corpus, Qwen2.5 tokenizer ($|V| = 151{,}936$), $\sim$20M calibration tokens. Each bin contains $\approx$15,194 token *types* assigned by ascending corpus frequency (rank-based, ties broken arbitrarily). Frequency ranges shown are typical for this setup; exact boundaries depend on the specific stream sampled.

Bin	Freq. range	Types	Description	Role
1	0	15,194	Zero-frequency & unused BPE pieces	**Rare**
2	0–2	15,194	Byte fallbacks, rare multilingual fragments	**Rare**
3	2–8	15,193	Rare technical vocabulary, foreign compounds	**Rare**
4	8–25	15,194	Uncommon content words	**Mid**
5	25–70	15,194	Infrequent content words	**Mid**
6	70–180	15,193	Moderate-frequency content words	**Mid**
7	180–450	15,194	Fairly common content words	**Mid**
8	450–1,200	15,194	Common words & mid function words	**Common**
9	1,200–5,000	15,193	High-frequency & function words	**Common**
10	5,000–350,000	15,193	Ubiquitous function, punctuation, subwords	**Common**
Rare examples (Bins 1–3)			<|endoftext|>, byte fallbacks, rare CJK/Cyrillic pieces	
Mid examples (Bins 4–7)			_volcano, _diocese, _battalion, _sculptor	
Common examples (Bins 8–10)			_the, _of, _and, _to, _in, ., ,	

Decile assignment of token types. All $V = 151{,}936$ token types (no structural filtering; see rationale above) are sorted by ascending corpus frequency, breaking ties by token id, and partitioned into $B = 10$ equal-cardinality bins of $\approx$15,194 types each. The bin index of token v is assigned by

$$b(v) \;=\; \min\left(\left\lfloor \frac{\mathrm{rank}(v)}{|V|} \cdot B \right\rfloor,\; B-1\right) \;\in\; \{0,1,\ldots,B-1\}, \tag{17}$$

C THE AVQ CENTROID UPDATE: DERIVATION, WOODBURY REDUCTION, AND COST

This appendix collects the material deferred from Section **??**: the per-pair gradient (Appendix C.1), the Woodbury reduction that makes the closed-form update cheap (Appendix C.2), and the resulting per-cluster algorithm, cost accounting, and reference implementation (Appendix **??**). Throughout we use the unit-norm property $\|\mu_i\| = 1$ guaranteed by Stage 2's normalisation.

C.1 PER-PAIR GRADIENT

The per-pair AVQ loss (Eq. equation 15) is $\mathcal{L}_{\text{AVQ}}(\mu_i, c) = \|\mu_i - c\|^2 + \lambda_{\text{aniso}}\langle \mu_i - c,\ \mu_i\rangle^2$.

Expanding the euclidean term $\|\mu_i - c\|^2 = \mu_i^\top \mu_i - 2\mu_i^\top c + c^\top c$ and using $\nabla_c(a^\top c) = a$ as well as $\nabla_c(c^\top c) = 2c$,

$$\nabla_c \|\mu_i - c\|^2 = -2\mu_i + 2c = -2(\mu_i - c). \tag{18}$$

The unit-norm property of anisotropic term gives $\langle \mu_i - c, \mu_i\rangle = \|\mu_i\|^2 - c^\top \mu_i = 1 - c^\top \mu_i$. So the anisotropic term is the scalar $g(c) = (1 - c^\top \mu_i)^2$.

With $u(c) = 1 - c^\top \mu_i$ and $\nabla_c u = -\mu_i$, the chain rule gives

$$\nabla_c g(c) = 2\,u(c)\,\nabla_c u(c) = -2\,(1 - c^\top \mu_i)\,\mu_i = -2\mu_i + 2\,(\mu_i \mu_i^\top)\,c, \tag{19}$$

We combine Equation 18 and λ_{aniso} times Equation 19, and have:

$$\begin{aligned}
\nabla_c \mathcal{L}_{\text{AVQ}}(\mu_i, c) &= \big(-2\mu_i + 2c\big) + \lambda_{\text{aniso}}\big(-2\mu_i + 2\mu_i\mu_i^\top c\big) \\
&= -2\mu_i - 2\lambda_{\text{aniso}}\mu_i + 2c + 2\lambda_{\text{aniso}}\mu_i\mu_i^\top c \\
&= -2(1 + \lambda_{\text{aniso}})\,\mu_i + 2\big(I_d + \lambda_{\text{aniso}}\,\mu_i\mu_i^\top\big)\,c.
\end{aligned} \tag{20}$$

The final two terms provide a clean interpretation: ❶ the *pull* term $-2(1 + \lambda_{\text{aniso}})\mu_i$ attracts the centroid toward μ_i with a force $(1 + \lambda_{\text{aniso}})\times$ harder than standard k-means ($-2\mu_i$); ❷ while the *stiffness* matrix $(I_d + \lambda_{\text{aniso}}\mu_i\mu_i^\top)$ is a direction-dependent restoring force (the geometric decomposition is given in Section 4.4). Summing Eq. equation 20 over $i \in \mathcal{M}_k$ and setting the result to zero gives the linear system $A_k c_k = b_k$ of Eq. equation 16, with $A_k = n_k I_d + \lambda_{\text{aniso}} \sum_{i\in\mathcal{M}_k} \mu_i\mu_i^\top$ and $b_k = (1 + \lambda_{\text{aniso}}) \sum_{i\in\mathcal{M}_k} \mu_i$.

C.2 WOODBURY REDUCTION

Solving $A_k c_k = b_k$ as a $d \times d$ system per cluster per Lloyd iteration is expensive. A_k's deviation from the scaled identity $n_k I_d$ has rank at most n_k. Stacking the cluster's unit-norm members as the rows of $X_k \in \mathbb{R}^{n_k \times d}$, so that $\sum_{i\in\mathcal{M}_k} \mu_i\mu_i^\top = X_k^\top X_k$ and $A_k = n_k I_d + \lambda_{\text{aniso}} X_k^\top X_k$. The Sherman-Morrison-Woodbury identity states that:

$$(A + UCV^\top)^{-1} = A^{-1} - A^{-1}U\big(C^{-1} + V^\top A^{-1}U\big)^{-1}V^\top A^{-1}, \tag{21}$$

Mapping as following:

$$A \leftarrow n_k I_d, \qquad U \leftarrow X_k^\top, \qquad V \leftarrow X_k^\top, \qquad C \leftarrow \lambda_{\text{aniso}} I_{n_k},$$

reproduces A_k since $A + UCV^\top = n_k I_d + X_k^\top(\lambda_{\text{aniso}} I_{n_k})X_k = n_k I_d + \lambda_{\text{aniso}} X_k^\top X_k = A_k$.

Because A and C are scaled identities, their inverses are straightforward as: $A^{-1} = \frac{1}{n_k} I_d$ and $C^{-1} = \frac{1}{\lambda_{\text{aniso}}} I_{n_k}$. The remaining blocks of Eq. equation 21 evaluate to:

$$A^{-1}U = \tfrac{1}{n_k}\,X_k^\top \ \ (d \times n_k), \quad V^\top A^{-1} = \tfrac{1}{n_k}\,X_k \ \ (n_k \times d), \quad V^\top A^{-1}U = \tfrac{1}{n_k}\,X_k X_k^\top \ \ (n_k \times n_k).$$

The central factor is therefore the $n_k \times n_k$ matrix $C^{-1} + V^\top A^{-1}U = \frac{1}{\lambda_{\text{aniso}}} I_{n_k} + \frac{1}{n_k} X_k X_k^\top =: M_k$. Substituting these blocks back into Eq. equation 21, the two factors of $\frac{1}{n_k}$ contributed by $A^{-1}U$ and $V^\top A^{-1}$ combine to $\frac{1}{n_k^2}$, which turns the $d \times d$ inverse into an $n_k \times n_k$ one:

$$A_k^{-1} = \frac{1}{n_k} I_d - \frac{1}{n_k^2}\, X_k^\top M_k^{-1} X_k, \qquad M_k := \frac{1}{\lambda_{\text{aniso}}}\, I_{n_k} + \frac{1}{n_k}\, X_k X_k^\top. \tag{22}$$

Writing $v := M_k^{-1}(X_k b_k)$ as the solution of the small system $M_k v = X_k b_k$, with M_k as the cluster's regularised matrix defined in Eq. equation 22, the centroid takes the form:

$$c_k = A_k^{-1} b_k = \frac{1}{n_k}\Big(b_k - \frac{1}{n_k} X_k^\top v\Big), \qquad \text{where } M_k v = X_k b_k. \tag{23}$$

A PyTorch reference implementation of the per-cluster update following directly from Equation 23 can be given as:

```
def avq_centroid(X_k, b_k, n_k, lambda_aniso):
    """One-cluster AVQ centroid update (Eq. for c_k).
    X_k: (n_k, d) unit-normalised cluster members, stored as rows.
    b_k: (d,) right-hand side, (1 + lambda_aniso) * sum_i mu_i.
    """
    I = torch.eye(n_k, device=X_k.device, dtype=X_k.dtype)
    M = I / lambda_aniso + (X_k @ X_k.T) / n_k      # SPD, n_k x n_k
    v = torch.linalg.solve(M, X_k @ b_k)            # M v = X_k b_k
    return (b_k - (X_k.T @ v) / n_k) / n_k           # centroid, (d,)
```

Algorithm 1 places this update inside the Lloyd loop; the partition is unchanged by λ_{aniso}, and the resulting cluster index is consumed by the same inference path as TKCS.

The centroid update factors into four substeps, each confined to the n_k-dimensional reduced space; Table 3 lists them with their per-cluster and full-pass costs (mean cluster size $\bar{n} = V/C$).

Substep	**Computation**	**Cost**
Right-hand side	$u := X_k b_k \in \mathbb{R}^{n_k}$, entry i: $\langle \mu_i, b_k \rangle$	$O(n_k d)$
Reduced matrix	$M_k = \frac{1}{\lambda_{\text{aniso}}} I_{n_k} + \frac{1}{n_k} X_k X_k^\top$	$O(n_k^2 d)$
Solve	$v = M_k^{-1} u$ (dense SPD solve)	$O(n_k^3)$
Centroid	$c_k = \frac{1}{n_k}\big(b_k - \frac{1}{n_k} X_k^\top v\big) \in \mathbb{R}^d$	$O(n_k d)$
Per-cluster total		$O(n_k^2 d + n_k^3)$
Full pass over C clusters ($\bar{n} = V/C$)		$O(V \bar{n}\, d + V \bar{n}^2)$

Table 3: Per-cluster AVQ centroid update cost.

Algorithm 1 Build cluster index Ω with optional AVQ refinement

Require: Representatives $X \in \mathbb{R}^{V \times d}$ ($\{\hat{W}_i\}$ for TKCS, $\{\mu_i\}$ for CHES, both unit-norm); W, b for bookkeeping; target C; anisotropy $\lambda_{\text{aniso}} \geq 0$ (0 recovers spherical k-means, 4 recommended for AVQ); size-cap multiple ρ

Ensure: Cluster index $\Omega = (\{c_k\}, \{\mathcal{M}_k\}, \{N_k\}, \{B_k\})$

Initialise $\{c_k\}_{k=1}^{C}$ by k-means++ on X
repeat
  $\pi(i) \leftarrow \arg\min_k \|X_i - c_k\|$ for all i ▷ L_2 assignment
  for $k = 1$ to C **do**
    $n_k \leftarrow |\{i : \pi(i)=k\}|$
    if $\lambda_{\text{aniso}} > 0$ **and** $n_k > 1$ **then**
      $b_k \leftarrow (1 + \lambda_{\text{aniso}}) \sum_{\pi(i)=k} X_i; \quad X_k \leftarrow [\, X_i \,]_{\pi(i)=k}$
      $c_k \leftarrow \frac{1}{n_k}\big(b_k - \frac{1}{n_k} X_k^\top v\big), M_k v = X_k b_k$ ▷ Woodbury solve, Equation 23
    else
      $c_k \leftarrow \text{mean}_{\pi(i)=k}\, X_i$ ▷ arithmetic mean (λ_{aniso}=0 or singleton)
    end if
  end for
until assignment change rate $<$ tol
Reseed empty clusters from farthest points of the largest clusters
Split clusters with $|\mathcal{M}_k| > \rho\,(V/C)$ by recursive k-means under the same λ_{aniso}
$c_k \leftarrow c_k / \|c_k\|$ ▷ snap to unit sphere
$\mathcal{M}_k \leftarrow \{i : \pi(i)=k\}; \quad N_k \leftarrow \max_{i \in \mathcal{M}_k} \|W_i\|; \quad B_k \leftarrow \max_{i \in \mathcal{M}_k} b_i$
return $\Omega = (\{c_k\}, \{\mathcal{M}_k\}, \{N_k\}, \{B_k\})$