MultiHashFormer: Hash-based Generative Language Models

Anonymous ACL submission

Abstract

Language models (LMs) represent tokens using embedding matrices that scale linearly with the vocabulary size. To constrain the parameter footprint, prior work proposes hashing many tokens into a single vector within encoder-only models. While this offers parameter efficiency, many-to-one collisions prevent its use in causal LMs. In this paper, we propose MULTIHASHFORMER, a new framework that allows hash-based autoregression. Each token is represented as a unique hash signature, a short sequence of discrete hash IDs, generated by multiple independent hash functions. A Hash Encoder compresses this signature into a single latent vector for processing by a Transformer decoder. Then, a Hash Decoder generates the hash signature of the next token, which is then mapped back to text. We evaluate our approach at the 100M, 1B and 3B parameter scales, demonstrating that MULTIHASHFORMER consistently outperforms standard Transformer LMs across multiple benchmarks. Furthermore, we show that our model handles multilingual vocabulary expansion with a constant parameter footprint without any modifications.[1]

1 Introduction

Language models (Ettinger et al., 2025; Team, 2025; Abdin et al., 2024, LMs) typically map tokens into high-dimensional vectors via learned embedding matrices. This allows each token in a discrete vocabulary to be represented by a unique, dense vector. However, this linear scaling creates a *vocabulary bottleneck*, locking the model into a fixed token capacity and restricting its ability to adapt seamlessly to new domains, or languages.

To mitigate this, previous research has explored token hashing (Prakash et al., 2020; Shu and Nakayama, 2018; Svenstrup et al., 2017; Ganchev and Dredze, 2008) to fix the parameter footprint of the embedding space. Hash-based models such as the Proformer (Sankar et al., 2021) and the HashFormer (Xue and Aletras, 2022) use many-to-one mappings, where multiple tokens (e.g., *cat*, *map*, and *physics*) may share the same hash index (e.g., 40). While parameter-efficient, these models are restricted to encoder-based architectures and discriminative training. In generative settings, if a decoder predicts a shared hash index, the model cannot deterministically recover the intended token from the set of colliding candidates. Consequently, this ambiguity has made training causal LMs on hashed spaces practically infeasible.

In this paper, we propose MULTIHASHFORMER, a new framework designed to bypass the vocabulary bottleneck for *decoder-based LMs*. Drawing inspiration from chaotic dynamic memory systems (Skarda and Freeman, 1987) which leverage distributed, overlapping state spaces for high-capacity pattern retrieval (Bricken and Pehlevan, 2021), we replace the traditional embedding matrix with a modular *hashing interface*. Specifically, each token is converted into a *unique hash signature* consisting of a short sequence of discrete hash IDs generated by multiple independent hash functions, similar to chaotic transitions that allow biological networks to chain together vast vocabularies of concepts (Tsuda, 2001). For example, *cat* is represented as $[12, 40, 56]$ and *map* as $[99, 40, 3]$. This multi-ID mapping eliminates the token collision problem of previous hash-based methods while allowing the model to *scale its vocabulary capacity with a strictly sub-linear parameter footprint*. For example, using four hash functions and 16,000 buckets per function, the MULTIHASHFORMER theoretically supports an upper bound of 16000^4 (approx. 65 quadrillion) unique signatures.

The interface is independent of the sequence processing backbone (e.g., a Transformer (Vaswani et al., 2017)) and manages the bidirectional translation between discrete tokens and the hash sig-

[1] Code is available at `https://anonymous.4open.science/r/MultiHashFormer-1B39/README.md`.

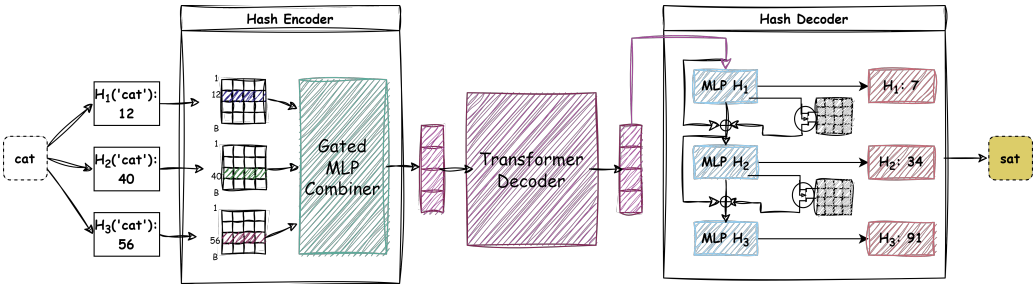

Figure 1: Overview of the MULTIHASHFORMER framework using three different hash functions (for illustration purposes) with B buckets to obtain a multi-ID hash signature.

natures. At the input, a *gated compositional embedding* compresses the multi-ID signature into a dense latent vector, providing the backbone with a unified representation for sequence processing (*Hash Encoder*). At the output, a *cascaded predictor* reconstructs the hash signature of the next token sequentially, which is then deterministically mapped back to a specific discrete text token in the vocabulary (*Hash Decoder*). Figure 1 illustrates the MULTIHASHFORMER architecture.

Our main contributions are as follows:

- We introduce the first hash-based framework that supports causal language modeling by preventing token collisions through multi-ID signature generation.
- MULTIHASHFORMER models consistently outperform standard Transformer LMs at 100M, 1B and 3B scales across 10 tasks, while offering better rare word representations.
- We show that our models maintain performance while expanding the vocabulary size from 32K to 48K without any structural changes, or parameter count increase.

2 Related Work

2.1 Token-Free Models

LMs rely on an embedding matrix that scales linearly with the vocabulary size. While subword tokenizers like BPE (Sennrich et al., 2016) and SentencePiece (Kudo and Richardson, 2018) mitigate vocabulary explosion, they remain constrained by a fixed, data-derived vocabulary that struggles with rare or out-of-domain words.

Token-free models such as AU-Net (Videau et al., 2025), Bolmo (Minixhofer et al., 2025), H-Net (Hwang et al., 2025), BLT (Pagnoni et al., 2025), CANINE (Clark et al., 2022), and ByT5 (Xue et al., 2022) bypass the vocabulary bottleneck entirely by operating directly on unicode or byte sequences. However, this approach dramatically increases the input sequence length. Alternatively, T-FREE (Deiseroth et al., 2024) avoids tokenization by embedding words via sparse activations over locality-based hashed character trigrams. While parameter-efficient, it relies on character-level morphological similarity, making it language-dependent. In contrast, MULTIHASHFORMER is orthogonal to these approaches. It resolves the vocabulary bottleneck while retaining the sequence compression advantages of subword tokenization. By decoupling the parameter matrix from the discrete vocabulary using combinatorial multi-hash mapping rather than sub-character heuristics, MULTIHASHFORMER remains language-agnostic and supports any arbitrary tokenization strategy.

2.2 Factorized and Hash-Based Embeddings

To compress the memory footprint of standard embedding matrices, prior work has explored parameter reduction techniques. For example, ALBERT (Lan et al., 2020) uses matrix factorization to decouple the embedding dimension from the hidden dimension. While this reduces the parameter footprint, it still allocates a localized, explicit vector per token, failing to break the linear scaling constraint.

A different approach is to use random hash embeddings (Svenstrup et al., 2017), which map discrete tokens into a highly compressed set of physical buckets. Architectural extensions like Proformer (Sankar et al., 2021) and HashFormer (Xue and Aletras, 2022) demonstrate that token hashing can be used to train Transformer-based models. However, compressing a vast vocabulary into a restricted physical bucket space inevitably forces multiple tokens to share the same index. This collision on a single hash function level prevents deterministic token recovery during

autoregressive generation, restricting these methods strictly to encoder-only architectures. MULTIHASHFORMER resolves this limitation via the multi-identifier framework detailed in §3.

3 MultiHashFormer

MULTIHASHFORMER comprises three modules shown in Figure 1: (1) a **Hash Encoder** that maps a discrete input token to a distributed multi-ID signature and compresses this signature into a single dense embedding; (2) a **Sequence Processing Backbone** that converts these embeddings into contextualized representations; and (3) a **Hash Decoder** that auto-regressively reconstructs the multi-ID signature of the next token.

3.1 Hash Encoder

Multi-Hash Indexing. To map each discrete token w into the hash signature space, we use H independent hash functions: $\mathcal{H}_1(w), \mathcal{H}_2(w), \ldots, \mathcal{H}_H(w)$. For any non-padding token, the i-th coordinate of the signature is computed using the non-cryptographic MurmurHash3 (MMH3) algorithm (Senuma, 2025). The algorithm uses iterative bitwise multiplication and rotation to diffuse the input, ensuring a single-bit change yields a uniform, randomized hash. We use a hash function-specific seed_i:

$$\mathcal{H}_i(w) = \text{MMH3}(w, \text{seed}_i) \pmod{B-1} + 1 \quad \text{if } w \neq \text{pad}.$$

B denotes the number of discrete hash buckets per function. $\mathcal{H}_i(\text{pad}) = 0$ is reserved across $\mathcal{H}$ to denote the padding token. To guarantee that every token in the vocabulary maps to a collision-free multi-hash ID, we use an iterative rehashing strategy. If a new token generates a signature that conflicts with an existing vocabulary entry, we incrementally modify the seed of the final hash function (seed_H) until an unused signature is found.

Gated Compositional Embedding. We allocate H separate embedding matrices $\mathbf{E}^{(i)} \in \mathbb{R}^{B \times d}$ (where $i \in \{1, \ldots, H\}$) to each hash coordinate $\mathcal{H}$, where d represents the backbone hidden dimension. Because hash buckets are shared globally per function, overlapping tokens that are semantically unrelated inevitably collide. Inspired by the multi-embedding approach of Guo et al. (2024), we resolve this ambiguity by compressing the coordinates into a unified token representation via a context-aware compositional gate. To determine the contribution of each bucket, the individual hash embeddings pass through a feed-forward bottleneck network with a compression dimension $d_z \ll d$, followed by softmax normalization. Finally, a linear adapter matrix $\mathbf{W}_s$ projects the combined representation into the latent space of the sequence processing backbone:

$$\mathbf{e} = \left(\sum_{i=1}^{H} \tilde{\alpha}_i \mathbf{E}^{(i)}_{[\mathcal{H}_i(w),:]} \right) \mathbf{W}_s,$$

$$\tilde{\alpha}_i = \frac{\exp^{\alpha_i}}{\sum_{j=1}^{H} \exp^{\alpha_j}}, \; \alpha_i = \sigma\left(\mathbf{E}^{(i)}_{[\mathcal{H}_i(w),:]} \mathbf{W}_1 \right) \mathbf{W}_2.$$

where $\mathbf{W}_s \in \mathbb{R}^{d \times d}$ represents the structural adapter, $\mathbf{W}_1 \in \mathbb{R}^{d \times d_z}$ and $\mathbf{W}_2 \in \mathbb{R}^{d_z \times 1}$ sequentially project the representations to an activation scaler with an non-linear activation above the intermediate bottleneck mainifold. Given an input sequence of tokens $(w_1, w_2, \ldots, w_n)$, the Hash Encoder processes each token w_i to its corresponding hash embedding $\mathbf{e}_i \in \mathbb{R}^d$.

3.2 Sequence Processing Backbone

$\mathbf{X} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n]^\top$ where $\mathbf{X} \in \mathbb{R}^{n \times d}$ passes through a standard stack of L Transformer layers. At each time-step t, the final layer of the backbone emits a contextualized latent vector $\mathbf{h}_t \in \mathbb{R}^d$: $\mathbf{h}_t = \text{Transformer}(\mathbf{X}_{[:t,:]})$. Subsequently, $\mathbf{h}_t$ is fed to the Hash Decoder.

3.3 Hash Decoder

To map the continuous hidden states back into the multi-hash ID signature space, we implement an auto-regressive *Cascaded Predictor*. This module iteratively refines its index predictions. It functions as a structured error-correcting system where preceding hashing choices directly constrain and contextualize subsequent token-signature inferences. To anchor this iterative cascade to the sequence context for predicting the next token w_{t+1}, the decoder initializes its prediction loop with the final hidden state of the backbone at the current time-step t. Thus, the root state $\mathbf{c}^{(1)} \in \mathbb{R}^d$ for the logit computation cascade is defined as $\mathbf{c}^{(1)} = \mathbf{h}_t$.

Logit Computation. For each sequential hash head $i \in \{1, \ldots, H\}$, the decoder projects the current latent hash state $\mathbf{c}^{(i)} \in \mathbb{R}^d$ to compute a logit distribution $\mathbf{o}^{(i)} \in \mathbb{R}^B$ over the physical bucket allocations. This is executed by slicing a dedicated

subspace of the hash head weights $\mathbf{W}_o^{(i)} \in \mathbb{R}^{d \times B}$:

$$\mathbf{o}^{(i)} = \mathbf{W}_o^{(i)\top} \mathbf{c}^{(i)},$$

where $\mathbf{W}_o^{(i)}$ defines the localized parameter weight matrix assigned exclusively to the i-th coordinate bucket array. Following Press and Wolf (2017), we tie the input and output embedding weights by setting $\mathbf{W}_o^{(i)\top} = \mathbf{E}^{(i)}$.

Soft Embedding Retrieval. For all non-terminal heads ($i < H$), the decoder extracts a continuous representation of its prediction to pass down the cascade. We first compute a normalized probability distribution $\mathbf{p}^{(i)} \in \mathbb{R}^B$ across the bucket candidates via a softmax operation:

$$\mathbf{p}^{(i)} = \mathrm{Softmax}(\mathbf{o}^{(i)}).$$

Using a hard discrete index during intermediate steps disrupts differentiability. Instead, the module computes a soft bucket embedding $\mathbf{e}^{(i)} \in \mathbb{R}^d$ as the expected value of the shared embedding matrix $\mathbf{E}^{(i)}$, weighted by the logit probabilities:

$$\mathbf{e}^{(i)} = \mathbf{E}^{(i)\top} \mathbf{p}^{(i)}$$

State Update via Cascade Mixer. To propagate the contextualized trajectory to the next signature head, the internal hash state requires an update. We achieve this using a recursive cascade mixer inspired by tree-structured recursive networks (Socher et al., 2011; Goller and Küchler, 1996). This block concatenates the existing state with the retrieved soft embedding and routes the joint representation through a bottleneck layer. Augmented with a structural residual connection, the subsequent state formulation $\mathbf{c}^{(i+1)}$ is defined as:

$$\mathbf{c}^{(i+1)} = \mathbf{c}^{(i)} + \mathbf{W}_{up}^{(i)\top} \sigma(\mathbf{W}_{dn}^{(i)\top} \begin{bmatrix} \mathbf{c}^{(i)} \\ \mathbf{e}^{(i)} \end{bmatrix}),$$

where σ is an element-wise activation, and $\mathbf{W}_{dn}^{(i)} \in \mathbb{R}^{2d \times b}$ and $\mathbf{W}_{up}^{(i)} \in \mathbb{R}^{b \times h}$ represent the low-rank projection weights of the bottleneck layer.

3.4 Probability Modeling

Standard autoregressive LMs decompose the joint probability of the input into a product of conditional probabilities. In contrast, MULTIHASHFORMER leverages the outputs of the Hash Decoder to factorize this token-level prediction. Specifically, the normalized distributions $(\mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(H)})$ produced iteratively by the cascaded predictor serve directly as the hash-specific conditional probabilities $P(\mathcal{H}_i(w_t) \mid \cdot)$ for each signature coordinate $i \in \{1, \ldots, H\}$.

Training Phase. The Hash Decoder maps tokens into a virtual vocabulary space $\mathcal{V}_{virt}$ defined by the total combinations of physical bucket allocations, where $|\mathcal{V}_{virt}| = B^H$. Because the actual language vocabulary is highly compact relative to this space, it forms a strict subset: $\mathcal{V}_{actl} \subset \mathcal{V}_{virt}$ and $|\mathcal{V}_{virt}| \gg |\mathcal{V}_{actl}|$. Consequently, we allow predictions over coordinate combinations that might not map to actual entries in the language vocabulary. This unconstrained formulation simplifies the optimization objective. The training probability distribution for a given token w is thus optimized via the product of its independent coordinate probabilities:

$$P_{train}(w \mid \cdot) = \prod_{i=1}^{H} P(\mathcal{H}_i(w) \mid \cdot).$$

Inference Phase. Allowing unconstrained decoding at inference could cause the model to generate coordinate combinations that do not map to actual semantic entries, leaking probability mass into invalid signatures. To guarantee the generation of valid multi-hash ID sequences, we explicitly exclude all unassigned signature during inference. This requires re-normalizing the probability distribution strictly over the true token vocabulary $\mathcal{V}_{actl}$:

$$P_{inf}(w \mid \cdot) = \frac{\prod_{i=1}^{H} P(\mathcal{H}_i(w) \mid \cdot)}{\sum_{w' \in \mathcal{V}_{actl}} \prod_{i=1}^{H} P(\mathcal{H}_i(w') \mid \cdot)}$$

By accumulating the log-probabilities of individual hash IDs, the final step is a standard softmax normalization over the valid token space $\mathcal{V}_{actl}$.

4 Experimental Setup

4.1 Models

We evaluate MULTIHASHFORMER models against standard autoregressive LMs across 100M, 1B and 3B parameter scales. All configurations use a decoder-only Transformer backbone based on the Qwen3 (Team, 2025) architecture. We use Mistral-7B-v0.3 (Jiang et al., 2023), a predominantly English BPE tokenizer with a 32K vocabulary.

Baselines. We use two baseline configurations: (1) A causal LM with a conventional embedding matrix $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}| \times d}$ and an equivalent LM head $\mathbf{W}_o \in \mathbb{R}^{d \times |\mathcal{V}|}$ (**Standard**); (2) a variant of Standard with k additional transformer layers to match the total parameter count of MULTIHASHFORMER (**Standard+kL**). This allows us to verify that

any performance improvements in MULTIHASHFORMER do not result from increased parameter capacity.[2] Embedding and LM head weights are tied similar to the MULTIHASHFORMER. We match the hidden dimension d and the number of attention heads to the MULTIHASHFORMER.

MultiHashFormer. We use the same core sequence processing backbone to the baseline configurations. We denote variants as HHBB, where H is the number of independent hash functions and B is the physical bucket allocation per head. We evaluate two configurations: **H3B10K** ($H = 3$, $B = 10,240$)[2], which directly matches the baseline parameter count, and **H4B16K** ($H = 4$, $B = 16,384$), our optimal configuration based on §6. For brevity, we fix the embedding matrices and projection layer sizes across all vocabulary settings. The bottleneck projection dimension for both the Gated Compositional Embedding and the Cascade Mixer is set to $d_z = 64$ (where $d_z \ll d$).

4.2 Pre-training Data and Hyperparameters

All models are pre-trained from scratch on a subset of English FineWeb-Edu (Penedo et al., 2024). Following Chinchilla scaling laws (Hoffmann et al., 2022), we adjust data volume based on model scale. The 100M models were trained on 10B tokens, and the 1B and 3B models on 100B tokens. We use a global batch size of 256 with a 2,048-token context window. See Table 7 in Appendix C for details.

4.3 Evaluation Benchmarks

Core capabilities. We include LAMBADA (Paperno et al., 2016) for Language Modeling; Commonsense Reasoning (CR) using ARC-Easy (Bhakthavatsalam et al., 2021), COPA (Roemmele et al., 2011), OBQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019); and Reading Comprehension (RC) with RACE (Lai et al., 2017), SciQ (Welbl et al., 2017), SIQA (Sap et al., 2019). We evaluate CR + RC using ReCoRD (Zhang et al., 2018).[3]

Rare Words. To investigate if the shared bucket mechanism of MULTIHASHFORMER improve rare word representations, we evaluate 1B models on the Card-660 dataset (Pilehvar et al., 2018). This dataset contains word pairs (rare-rare or rare-frequent) with human-annotated similarity scores. We compute the cosine similarity between the words using the final and second-to-last hidden states corresponding to the last token. Finally, we measure the correlation between these model-derived scores and the human annotations.

4.4 Vocabulary Expansion

As mentioned in §1, a key theoretical advantage of MULTIHASHFORMER is its ability to expand its vocabulary capacity without increasing its parameter footprint. To evaluate this capability, we test the models on their ability to integrate new tokens during multilingual vocabulary expansion.

Continual Pre-training Data. We continue pre-training (CPT) models on a multilingual corpus containing 6B tokens evenly split across Arabic, Chinese, and Hindi from FineWeb2 (Penedo et al., 2025). We also include an extra 2B English tokens from FineWeb-Edu (Penedo et al., 2024) to mitigate catastrophic forgetting. Following Yao et al. (2021), we add 5K tokens per language, yielding a final vocabulary of 48K tokens.

Model Configuration. For MULTIHASHFORMER, we expand the vocabulary size by registering new unique virtual (multi-hash ID) signatures *without adding new parameters*. For baselines, the weights of the new tokens within the embedding matrix are initialized using mean initialization, a standard protocol where each new token receives the average embedding of its corresponding source tokens from the original tokenizer (Mundra et al., 2024; Tejaswi et al., 2024; Yamaguchi et al., 2025, 2026). To reduce computational overhead, we only update the embedding layer, the LM heads for the Standard baselines, and the hash encoder and decoder for MULTIHASHFORMER. Following Remy et al. (2024); Owodunni and Kumar (2025); Nakash et al. (2025), the first two and last two Transformer layers are also updated. See Table 8 (Appendix C) for full details.

Evaluation. We use five multilingual tasks from MuBench (Han et al., 2025), which are HellaSwag, TruthfulQA (Lin et al., 2022), StoryCloze (Mostafazadeh et al., 2016), and

[2] Due to computational resource constraints, we consider this variant for 100M and 1B scales only.

[3] We report normalized accuracy for ARC-E, HellaSwag, SIQA, OBQA, and SciQ; F1 scores for ReCoRD; and standard accuracy for all remaining tasks. We use a five-shot setting for ARC-E, SIQA, and OBQA, while the rest are evaluated in a zero-shot setting.

		#Param		Lang. Model.	Commonsense Reasoning					Reading Comprehension			CR+RC
	Model	**Emb**	**Dec**	**LAMBADA**	**ARC-E**	**COPA**	**OBQA**	**PIQA**	**HellaSwag**	**RACE**	**SciQ**	**SIQA**	**ReCoRD**
	Rnd. Guess	-	-	-	25.00	50.00	25.00	50.00	25.00	25.00	25.00	25.00	19.10
100M	Standard	25M	87M	15.33	41.33	**64.00**	27.00	58.87	28.59	26.41	59.00	35.98	49.47
	Standard+4L	25M	116M	**19.64**	45.08	**64.00**	27.00	60.17	**29.99**	26.79	**62.60**	36.80	**53.08**
	MHF (H3B10K)	25M	87M	16.40	44.91	61.00	26.40	**60.23**	27.81	27.18	55.50	**37.00**	48.21
	MHF (H4B16K)	52M	87M	18.79	**45.24**	62.00	**27.60**	59.03	27.99	**28.42**	58.00	36.03	50.11
1B	Standard	65M	0.9B	30.41	59.85	64.00	31.60	65.78	38.99	29.38	69.90	39.10	63.20
	Standard+2L	65M	1.0B	30.66	60.23	65.00	31.20	65.02	39.85	29.28	**72.10**	38.89	63.78
	MHF (H3B10K)	65M	0.9B	**35.78**	61.95	65.00	**34.00**	**68.17**	40.45	29.00	68.30	40.48	64.51
	MHF (H4B16K)	138M	0.9B	35.34	**64.35**	**71.00**	29.80	66.49	**41.32**	**31.39**	70.20	**41.66**	**64.90**
3B	Standard	65M	2.8B	28.64	60.23	65.00	31.80	65.83	39.65	29.76	64.80	38.28	62.26
	MHF (H4B16K)	138M	2.8B	**37.26**	**66.88**	**68.00**	**34.60**	**68.39**	**42.89**	**30.91**	**70.60**	**40.74**	**64.13**

Table 1: Performance of standard and our MULTIHASHFORMER models. Bold denotes best performance.

MMLU (Hendrycks et al., 2021).[4] We also include the English benchmarks from §4.3 to track catastrophic forgetting.

5 Results

5.1 Core Capabilities

Table 1 presents the performance of the baselines alongside MULTIHASHFORMER (MHF) models across the 100M, 1B, and 3B parameter scales.

Efficacy while scaling parameter counts. We first observe that as the model capacity increases from 1B to 3B parameters, MHF (H4B16K) consistently outperforms the baseline on 9 out of 11 tasks. Specifically, on the LAMBADA task, MHF (H4B16K) offers substantial gains of 4.93% and 8.62% over the Standard baseline (30.41% and 28.64%) at the 1B and 3B scales, respectively. Although the standard LM baseline achieves higher scores on OBQA at the 1B scale, the overall results indicate the efficacy of MULTIHASHFORMER in capturing contextual dependencies.

Gains under strict parameter matching. When matching the parameter count at the 1B scale, MULTIHASHFORMER variants offer superior performance compared to baselines. Specifically, MHF (H3B10K) and MHF (H4B16K) outperform Standard and Standard+2L baselines on 8 out of 10 tasks, respectively. Notably, MHF (H4B16K) achieves 64.90 on ReCoRD, surpassing the 63.78 of Standard+2L. Similarly, MHF (H3B10K) outperforms Standard on COPA with 71.00 vs. 65.00. These results suggest that decoupling the vocabulary is effective. Allocating more parameters to the embeddings improves performance more than adding more layers to the model on larger scales.

Per-task performance. Looking at specific tasks, we note a performance trade-off based on the hash configuration. On OBQA, models with fewer hash buckets achieve higher accuracy. This reversal likely stems from the long-tail distribution of the dataset. A large bucket allocation fragments the embedding space for sparse vocabulary items, which degrades the representation of rare factual tokens. Conversely, reading comprehension requires high representational capacity to prevent coordinate-level collisions between tokens across complex contexts. At the 1B scale, the lower-capacity MHF (H3B10K) variant underperforms the Standard baseline on SciQ (68.30% vs. 69.90%) and RACE (29.00% vs. 29.38%), although it outperforms the baseline on SIQA (40.48% vs. 39.10%). Expanding the physical bucket allocation to the MHF (H4B16K) configuration improves these scores to 70.20% on SciQ and 31.39% on RACE, surpassing the Standard baseline. MULTIHASHFORMER's strong performance on LM and reasoning tasks stems primarily from its ability to mitigate the softmax bottleneck (Yang et al., 2018), which is further analyzed in Appendix A. This advantage is especially evident in LAMBADA, which requires predicting specific, context-dependent words, and HellaSwag, which tests narrative logic. Our model effectively distinguishes between high-probability candidates that Standard baselines often conflate.

Performance saturation at 100M. At the 100M scale, MULTIHASHFORMER variants do not consistently outperform baselines, especially on COPA, HellaSwag, and ReCoRD. Instead, increasing decoder depth by 4 layers yields higher accuracy gains across 7 tasks than expanding hash embeddings. This suggests that for smaller models, prior-

[4] We only include tasks where all Standard baselines outperform random guessing. For MMLU, we apply this filtering criterion at the subtask level and report the average over the retained subtasks. We report zero-shot accuracy for MMLU and normalized accuracy for the remaining tasks.

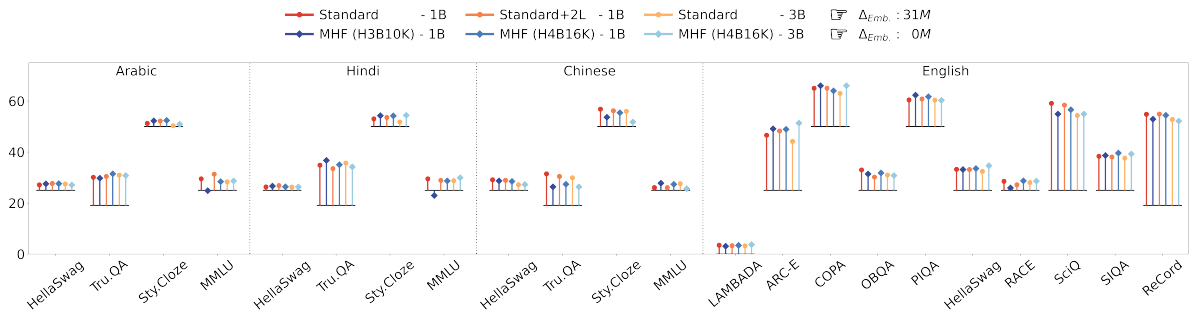

Figure 2: MULTIHASHFORMER (blue shades, diamond marker) are similar or better than Standard (orange shades, circle marker) at 1B and 3B scales after expanding the vocabulary from 32K to 47K without adding a single new parameter. Black horizontal lines denote random baseline performance per task.

Model	Last Hidden States		Second-to-Last Hidden States	
	Pearson $r \uparrow$	Spearman $\rho \uparrow$	Pearson $r \uparrow$	Spearman $\rho \uparrow$
Starndard	0.20	0.22	0.26	0.25
MHF (H3B10K)	**0.23**	**0.23**	**0.32**	**0.29**
Standard+2L	**0.22**	0.22	0.29	0.29
MHF (H4B16K)	0.20	**0.26**	**0.30**	**0.32**

Table 2: Correlations between human annotations and cosine similarities computed from the last two hidden states of the 1B models on the Card-660 dataset.

itizing decoder depth over embedding size is more effective. We attribute this to three constraints of smaller and shallower architectures: (1) restricted hidden dimensions conflate distinct tokens with similar hash signatures; (2) shallower depth prevents the model from contextualizing and disambiguating these embeddings; and (3) narrower hidden states restrict the rank upper bound (see our theoretical analysis in Appendix A), degrading log-probability estimation. Consequently, MULTIHASHFORMER scales more effectively in larger, billion-parameter decoders.

5.2 Rare Word Representations

Table 2 shows the correlation coefficients on the Card-660 dataset. The MULTIHASHFORMER variants consistently outperform Standard models while maintaining an equivalent parameter count. We find that MULTIHASHFORMER is better at identifying semantically equivalent word pairs (see examples in Appendix B). This performance gap is particularly evident when analyzing hidden states from the second-to-last decoder layer, where representations are less biased toward the specific training tasks. The MULTIHASHFORMER architecture effectively captures the semantic representations of rare vocabulary items by forcing sparse tokens to share representational capacity.

5.3 Vocabulary Expansion

Figure 2 illustrates the performance of the adapted baselines compared to the adapted MULTIHASHFORMER models at 1B and 3B parameter scales across Arabic, Hindi, Chinese, and English. Notably, both the 1B and 3B MHF (H4B16K) models consistently outperform Standard on 12 and 13 out of 22 multilingual tasks, respectively. They achieve this without the additional 31-million-parameters needed to accommodate 15K new tokens in the expanded vocabulary of the Standard baselines. We further find that MHF (H3B10K) and MHF (H4B16K) perform comparably (difference smaller than 1%) on 15 out of 22 tasks. This indicates that performance remains robust during vocabulary expansion. Furthermore, under a strict parameter-count matching constraint prior to vocabulary expansion, MHF (H3B10K) and MHF (H4B16K) achieve better or comparable performance to the Standard models on the majority of English tasks (6 and 9 out of 10, respectively). This suggests that MULTIHASHFORMER does not suffer from more catastrophic forgetting relative to Standard. It successfully supports a 46.9% larger multilingual vocabulary *without a single additional parameter* while preserving its core capabilities in English.

6 Analysis

We further analyze the core architectural components of MULTIHASHFORMER. Due to computational resource constraints, we conduct this analysis at the 100M parameter scale.

Single vs. Multi-hash ID. To demonstrate the necessity of Multi-ID signatures, we evaluate three different MULTIHASHFORMER configurations (H4B4K, H4B8K, and H4B16K) against

MHF	B4K	B8K	B16K
H1	2.17	5.69	9.72
H4	16.42	16.84	19.74

Table 3: LAMBADA accuracy across different MHF (H4BB) configurations against their corresponding variants H1BB using Single-ID signature.[5]

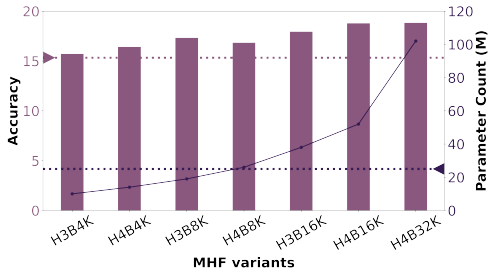

Figure 3: LAMBADA accuracy scores across MULTIHASHFORMER variants at the 100M scale against their parameter counts in embeddings. The horizontal dotted lines represent the Standard baseline.

their Single-ID counterparts (H1B4K, H1B8K, and H1B16K). Table 3 presents the results on LAMBADA. Models with Multi-ID signatures consistently outperform those using Single-ID signatures across all configurations (B4K, B8K, and B16K). Specifically, the performance difference is higher at lower bucket capacities (B4K), where H4B4K achieves 16.42% accuracy compared to the 2.17% of H1B4K. Expanding the capacity to B16K with Single-ID only improves accuracy to 9.72%. These results suggest that a signature collision severely degrades the core capability of the model. Employing Multi-ID signatures to prevent such collisions is more effective than increasing the number of hash buckets.

Varying Hash Functions and Bucket Size. To evaluate the impact of hash configuration on parameter efficiency and downstream performance, we test various combinations of hash functions (H) and bucket sizes (B). Figure 3 illustrates LAMBADA accuracy of MULTIHASHFORMER variants against their embedding parameter counts, across different combinations of H and B. All MULTIHASHFORMER configurations consistently outperform Standard, including variants that utilize strictly smaller embedding matrices, such as H3B4K, H4B4K, and H3B8K. Although scaling both H and B sharply increases the embedding parameter count, the corresponding accuracy scores remain stable. Specifically, the best-performing variant improves over the lowest by only 3%, despite a tenfold increase in parameter count from 10M to 102M. Consequently, the H4B16K variant achieves an optimal balance between parameter efficiency and predictive accuracy. We therefore adopt H4B16K as the primary configuration for the 1B and 3B scaling experiments presented in §5.1.

[5]H1BB use single value for 4 coordinates to strictly match the number of parameters in H4B for fair comparison.

LSH : MMH3	0 : 3	1 : 2	2 : 1	3 : 0
LAMBADA	17.95	18.75	18.20	16.79

Table 4: LAMBADA accuracy across different MHF(H3B16K) variants while replacing a different number of MMH3 functions to LSH.

MMH3 vs. LSH. We also investigate whether incorporating Locality-Sensitive Hashing (Datar et al., 2004, LSH) enhances the general capability of the model by capturing morphological similarity. Because LSH maps structurally similar input features to identical hash buckets with high probability, applying it to character-level trigrams tends to force subwords with similar spelling to share embedding representations. To evaluate this mechanism, we pre-train three additional MHF (H3B16K) variants from scratch at a 100M-parameter scale, keeping the dataset and hyperparameters identical. Table 4 presents the performance of these models on LAMBADA. Increasing the proportion of LSH functions while fixing the total number of hash functions does not yield a consistent accuracy improvement. This outcome suggests that explicitly injecting a morphological inductive bias is not necessary given the inherent computational complexity of LSH compared to MMH3. The standard deterministic random hashing of MMH3 provides sufficient representational flexibility for the network to learn subword relationships entirely end-to-end.

7 Conclusion

We introduced MULTIHASHFORMER, a generative framework that leverages multi-hash structures to bypass traditional vocabulary bottlenecks. Empirical evaluation demonstrates that it offers improvements both in core capabilities and rare word representations. Furthermore, the framework enables seamless vocabulary expansion without requiring additional parameters.

Limitations

Model size. Due to computational resource constraints, we restrict our evaluation to models at the 100M, 1B and 3B scales. Following the protocol of Cheng et al. (2024), 1B and 3B models were trained from scratch within a standard academic budget of 100B tokens. While MULTIHASHFROMER shows an upward performance trajectory with model size, exploring larger scales (e.g., 7B+ parameters) remains a valuable direction for institutions with access to the required computing resources.

Single seed. Due to the high computational cost of pre-training 1B and 3B parameter models from scratch for 100B tokens, conducting multiple training runs was prohibitive under our academic infrastructure constraints. Consequently, our empirical results are based on a single seed per configuration. However, the consistent performance gains observed across both the 1B and 3B scales demonstrate the robustness of the MULTIHASHFORMER framework. This consistency indicates that the improvements introduced by our Hash Encoder and Hash Decoder stem from inductive bias rather than random sampling variance. Furthermore, adopting the established hyperparameter protocols of Li et al. (2025) and Cheng et al. (2024), we ensure standard and reproducible training dynamics.

References

Marah I Abdin, Jyoti Aneja, Harkirat S. Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, and 8 others. 2024. Phi-4 technical report. *CoRR*, abs/2412.08905.

Sumithra Bhakthavatsalam, Daniel Khashabi, Tushar Khot, Bhavana Dalvi Mishra, Kyle Richardson, Ashish Sabharwal, Carissa Schoenick, Oyvind Tafjord, and Peter Clark. 2021. Think you have solved direct-answer question answering? try arc-da, the direct-answer AI2 reasoning challenge. *CoRR*, abs/2102.03315.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press.

Trenton Bricken and Cengiz Pehlevan. 2021. Attention approximates sparse distributed memory. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 15301–15315.

Daixuan Cheng, Yuxian Gu, Shaohan Huang, Junyu Bi, Minlie Huang, and Furu Wei. 2024. Instruction pre-training: Language models are supervised multitask learners. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2529–2550, Miami, Florida, USA. Association for Computational Linguistics.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91.

Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 20th ACM Symposium on Computational Geometry, Brooklyn, New York, USA, June 8-11, 2004*, pages 253–262. ACM.

Björn Deiseroth, Manuel Brack, Patrick Schramowski, Kristian Kersting, and Samuel Weinbach. 2024. T-FREE: Subword tokenizer-free generative LLMs via sparse representations for memory-efficient embeddings. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21829–21851, Miami, Florida, USA. Association for Computational Linguistics.

Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee F. Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, and 48 others. 2025. Olmo 3. *CoRR*, abs/2512.13961.

Kuzman Ganchev and Mark Dredze. 2008. Small statistical models by random feature mixing. In *Proceedings of the ACL-08: HLT Workshop on Mobile Language Processing*, pages 19–20, Columbus, Ohio. Association for Computational Linguistics.

Christoph Goller and Andreas Küchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of International Conference on Neural Networks (ICNN'96), Washington, DC, USA, June 3-6, 1996*, pages 347–352. IEEE.

Xingzhuo Guo, Junwei Pan, Ximei Wang, Baixu Chen, Jie Jiang, and Mingsheng Long. 2024. On the embedding collapse when scaling up recommendation models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria,*

July 21-27, 2024, Proceedings of Machine Learning Research, pages 16891–16909. PMLR / OpenReview.net.

Wenhan Han, Yifan Zhang, Zhixun Chen, Binbin Li, Haobin Lin, Bingni Zhang, Taifeng Wang, Mykola Pechenizkiy, Meng Fang, and Yin Zheng. 2025. Mubench: Assessment of multilingual capabilities of large language models across 61 languages. *CoRR*, abs/2506.19468.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, and 3 others. 2022. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Sukjun Hwang, Brandon Wang, and Albert Gu. 2025. Dynamic chunking for end-to-end hierarchical sequence modeling. *CoRR*, abs/2507.07955.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Houyi Li, Wenzhen Zheng, Jingcheng Hu, Qiufeng Wang, Hanshan Zhang, Zili Wang, Shijie Xuyang, Yuantao Fan, Shuigeng Zhou, Xiangyu Zhang, and Daxin Jiang. 2025. Predictable scale: Part I - optimal hyperparameter scaling law in large language model pretraining. *CoRR*, abs/2503.04715.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Benjamin Minixhofer, Tyler Murray, Tomasz Limisiewicz, Anna Korhonen, Luke Zettlemoyer, Noah A. Smith, Edoardo M. Ponti, Luca Soldaini, and Valentin Hofmann. 2025. Bolmo: Byteifying the next generation of language models. *CoRR*, abs/2512.15586.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California. Association for Computational Linguistics.

Nandini Mundra, Aditya Nanda Kishore Khandavally, Raj Dabre, Ratish Puduppully, Anoop Kunchukuttan, and Mitesh M Khapra. 2024. An empirical comparison of vocabulary expansion and initialization approaches for language models. In *Proceedings of the 28th Conference on Computational Natural Language Learning*, pages 84–104, Miami, FL, USA. Association for Computational Linguistics.

Itay Nakash, Nitay Calderon, Eyal Ben-David, Elad Hoffer, and Roi Reichart. 2025. Adaptivocab: Enhancing LLM efficiency in focused domains through lightweight vocabulary adaptation. *CoRR*, abs/2503.19693.

Abraham Toluwase Owodunni and Sachin Kumar. 2025. Continually adding new languages to multilingual language models. *CoRR*, abs/2509.11414.

Artidoro Pagnoni, Ramakanth Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason E Weston, Luke

Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, and Srini Iyer. 2025. Byte latent transformer: Patches scale better than tokens. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9238–9258, Vienna, Austria. Association for Computational Linguistics.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.

Guilherme Penedo, Hynek Kydlícek, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin A. Raffel, Leandro von Werra, and Thomas Wolf. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Guilherme Penedo, Hynek Kydlícek, Vinko Sabolcec, Bettina Messmer, Negar Foroutan, Amir Hossein Kargaran, Colin Raffel, Martin Jaggi, Leandro von Werra, and Thomas Wolf. 2025. Fineweb2: One pipeline to scale them all - adapting pretraining data processing to every language. *CoRR*, abs/2506.20920.

Mohammad Taher Pilehvar, Dimitri Kartsaklis, Victor Prokhorov, and Nigel Collier. 2018. Card-660: Cambridge rare word dataset - a reliable benchmark for infrequent word representation models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1391–1401, Brussels, Belgium. Association for Computational Linguistics.

Prafull Prakash, Saurabh Kumar Shashidhar, Wenlong Zhao, Subendhu Rongali, Haidar Khan, and Michael Kayser. 2020. Compressing transformer-based semantic parsing models using compositional code embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4711–4717, Online. Association for Computational Linguistics.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.

François Remy, Pieter Delobelle, Hayastan Avetisyan, Alfiya Khabibullina, Miryam de Lhoneux, and Thomas Demeester. 2024. Trans-tokenization and cross-lingual vocabulary transfers: Language adaptation of llms for low-resource NLP. *CoRR*, abs/2408.04303.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *Logical Formalizations of Commonsense Reasoning, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-06, Stanford, California, USA, March 21-23, 2011*. AAAI.

Chinnadhurai Sankar, Sujith Ravi, and Zornitsa Kozareva. 2021. ProFormer: Towards on-device LSH projection based transformers. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2823–2828, Online. Association for Computational Linguistics.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Hajime Senuma. 2025. mmh3: A python extension for murmurhash3. *J. Open Source Softw.*, 10(105):6124.

Raphael Shu and Hideki Nakayama. 2018. Compressing word embeddings via deep compositional code learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Christine A. Skarda and Walter J. Freeman. 1987. How brains make chaos in order to make sense of the world. *Behavioral and Brain Sciences*, 10(2):161–173.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 129–136. Omnipress.

Dan Svenstrup, Jonas Meinertz Hansen, and Ole Winther. 2017. Hash embeddings for efficient word representations. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4928–4936.

Qwen Team. 2025. Qwen3 technical report. *CoRR*, abs/2505.09388.

Atula Tejaswi, Nilesh Gupta, and Eunsol Choi. 2024. Exploring design choices for building language-specific LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10485–10500, Miami, Florida, USA. Association for Computational Linguistics.

Ichiro Tsuda. 2001. Toward an interpretation of dynamic neural activity in terms of chaotic dynamical systems. *Behavioral and Brain Sciences*, 24(5):793–810.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Mathurin Videau, Badr Youbi Idrissi, Alessandro Ferreira Leite, Marc Schoenauer, Olivier Teytaud, and David Lopez-Paz. 2025. From bytes to ideas: Language modeling with autoregressive u-nets. *CoRR*, abs/2506.14761.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, Copenhagen, Denmark. Association for Computational Linguistics.

Huiyin Xue and Nikolaos Aletras. 2022. HashFormers: Towards vocabulary-independent pre-trained transformers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7862–7874, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

Atsuki Yamaguchi, Terufumi Morishita, Aline Villavicencio, and Nikolaos Aletras. 2025. Adapting chat language models using only target unlabeled language data. *Trans. Mach. Learn. Res.*, 2025.

Atsuki Yamaguchi, Aline Villavicencio, and Nikolaos Aletras. 2026. How can we effectively expand the vocabulary of LLMs with 0.01GB of target language text? *Computational Linguistics*, 52(1):295–330.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Yunzhi Yao, Shaohan Huang, Wenhui Wang, Li Dong, and Furu Wei. 2021. Adapt-and-distill: Developing small, fast and effective pretrained language models for domains. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 460–470, Online. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *CoRR*, abs/1810.12885.

A Softmax Bottleneck

A.1 Preliminaries

Language models (LMs) define the conditional distribution $P_\theta(w|c)$ by applying a softmax function to a linear projection of the hidden state $\mathbf{h}_t$:

$$P_\theta(w|c) = \frac{\exp \mathbf{h}_t^\top \mathbf{w}_w}{\sum_{w'} \exp \mathbf{h}_t^\top \mathbf{w}_{w'}}$$

where both the context vector $\mathbf{h}_t(c_t;\theta)$ and the word embedding $\mathbf{w}_{w_t}(w_t;\theta)$ are on a d-dimensional space. Their inner product, $\mathbf{h}_t^\top \mathbf{w}_{w_t}$, defines the logit. To analyze the expressive capacity of the softmax function, we define the following three matrices and their log-probabilities:

$$\mathbf{H}_\theta = [\mathbf{h}_1, \ldots, \mathbf{h}_n]^\top, \quad \mathbf{W}_\theta = [\mathbf{w}_1, \ldots, \mathbf{w}_{|\mathcal{V}|}]^\top$$
$$\mathbf{A} = [a_{i,j}]_{n\times|\mathcal{V}|} \in \mathbb{R}^{n\times|\mathcal{V}|}, \text{ where } a_{i,j} = \log p^*(w_j|c_i)$$

$$\begin{aligned}\log p^*(w|c) &= \log\left(\frac{\exp(\mathbf{h}_t^\top \mathbf{w}_w)}{\sum_{w'} \exp(\mathbf{h}_t^\top \mathbf{w}_{w'})}\right)\\ &= \mathbf{h}_t^\top \mathbf{w}_w - \log \sum_{w'} \exp(\mathbf{h}_t^\top \mathbf{w}_{w'})\end{aligned}$$

$\mathbf{H}_\theta \in \mathbb{R}^{n\times d}$ denote the matrix of $\mathbf{h}_t$ and $\mathbf{W}_\theta \in \mathbb{R}^{|\mathcal{V}|\times d}$ is the projection matrix for all tokens in the vocabulary. The matrix $\mathbf{A} \in \mathbb{R}^{n\times|\mathcal{V}|}$ contains the log probabilities of the true data distribution. We index $\mathbf{H}$ and $\mathbf{W}$ by θ to indicate that both represent functions within a joint family $\mathcal{U}$ parameterized by θ. In practice, $\mathbf{H}_\theta$ is realized via a Transformer-based LM, while $\mathbf{W}_\theta$ is instantiated as a learned token embedding lookup table.

We further define a set of matrices, $F(\mathbf{A})$, generated by applying a row-wise shift to $\mathbf{A}$:

$$F(\mathbf{A}) = \{\mathbf{A} + \Lambda \mathbf{J}_{n,|\mathcal{V}|} \mid \Lambda \in \mathbb{R}^{n\times n} \text{ is a diagonal matrix}\}$$

where $\mathbf{J}_{n,|\mathcal{V}|} \in \mathbb{R}^{n\times|\mathcal{V}|}$ denotes the all-ones matrix. This row-wise shift operation effectively adds a unique scalar constant to each element of $\mathbf{A}$'s rows. Since the diagonal entries of Λ can be any real values, $F(\mathbf{A})$ constitutes an infinite set characterized by two key properties:

Property 1. *For any matrix $\mathbf{A}'$, $\mathbf{A}' \in F(\mathbf{A})$ if and only if Softmax$(\mathbf{A}') = P^*$. Thus, $F(\mathbf{A})$ represents the set of all logit matrices that recover the true data distribution.*

Property 2. *For any $\mathbf{A}_1, \mathbf{A}_2 \in F(\mathbf{A})$ such that $\mathbf{A}_1 \neq \mathbf{A}_2$, it holds that $|rank(\mathbf{A}_1) - rank(\mathbf{A}_2)| \leq 1$. Hence, the matrices in $F(\mathbf{A})$ maintain consistent ranks, with a maximum discrepancy of one.*

Given Property A.1, we establish:

Lemma 1. *Given a model parameter θ, $\mathbf{H}_\theta \mathbf{W}_\theta^\top \in F(\mathbf{A})$ if and only if $P_\theta(X|c)$ holds for all $c \in \mathcal{L}$.*

Following this lemma, expressiveness is framed as follows: does there exist a parameter θ and a matrix $\mathbf{A}' \in F(\mathbf{A})$ satisfying $\mathbf{H}_\theta \mathbf{W}_\theta^\top = \mathbf{A}'$. LMs learn matrices $\mathbf{H}_\theta$ and $\mathbf{W}_\theta$ to factorize a target matrix $\mathbf{A}' \in F(\mathbf{A})$. For a valid factorization, the rank of the product $\mathbf{H}_\theta \mathbf{W}_\theta^\top$ must be at least equal to the rank of $\mathbf{A}'$. Because $\mathbf{H}_\theta \in \mathbb{R}^{n\times d}$ and $\mathbf{W}_\theta \in \mathbb{R}^{|\mathcal{V}|\times d}$, the rank of this product is strictly upper-bounded by the embedding dimension d. Consequently, if $d \geq \text{rank}(\mathbf{A}')$, a universal approximator can recover $\mathbf{A}'$. Conversely, if $d < \text{rank}(\mathbf{A}')$, no pair $(\mathbf{H}_\theta, \mathbf{W}_\theta)$ can recover $\mathbf{A}'$, regardless of the expressiveness of $\mathcal{U}$.

Proposition 1. *Assuming the function family $\mathcal{U}$ is a universal approximator, there exists a parameter θ such that $P_\theta(X|c) = P^*(X|c)$ for all $c \in \mathcal{L}$ if and only if $d \geq \min_{\mathbf{A}' \in F(\mathbf{A})} rank(\mathbf{A}')$.*

By combining Proposition A.1 with the properties of $F(\mathbf{A})$ outlined in Property A.1, Yang et al. (2018) define the Softmax Bottleneck:

Corollary 1. ***(Softmax Bottleneck).*** *If $d < rank(\mathbf{A}) - 1$, then for any function family $\mathcal{U}$ that acts as a universal approximator, there is no parameter θ such that $P_\theta(X|c) = P^*(X|c)$ for all $c \in \mathcal{L}$. That is, $P_\theta(X|c) \neq P^*(X|c)$ must hold for at least some contexts.*

This demonstrates that an insufficient dimension d restricts Softmax from expressing the true data distribution. The conclusion is not restricted to a finite language $\mathcal{L}$. Because when $\mathcal{L}$ is infinite, one can always take a finite subset and the softmax bottleneck still exists.

A.2 MULTIHASHFORMER as a Mitigation Strategy

While training, MULTIHASHFORMER approximates elements in $\mathbf{A}$ by

$$\begin{aligned}\log p^*(w|c) &= \log \prod_i^H p^*(\mathcal{H}_i(w)|c)\\ &= \sum_{i=1}^H \log p^*(\mathcal{H}_i(w)|c)\\ &= \sum_{i=1}^H \log\left(\frac{\exp(\mathbf{c}^{(i)\top}\mathbf{w}^{(i)}_{\mathcal{H}_i(w)})\pi^{(i)}_{\mathcal{H}_i(w)}}{\sum_{j=1}^B \exp(\mathbf{c}^{(i)\top}\mathbf{w}^{(i)})}\right)\\ &= \sum_{i=1}^H \mathbf{c}^{(i)\top}\mathbf{w}^{(i)}_{\mathcal{H}_i(w)}\pi^{(i)}_{\mathcal{H}_i(w)}\\ &\quad - \sum_{i=1}^H \log \sum_{j=1}^B \exp(\mathbf{c}^{(i)\top}\mathbf{w}^{(i)}_j)\end{aligned}$$

The model projects representations to coordinates of the hash signature, using H localized parameter weight matrices assigned exclusively to i-th coordinate bucket array, $\mathbf{W}_o^{(i)} \in \mathbb{R}^{d \times B}$. These coordinates are then mapped to the true vocabulary space via transition matrices $\pi^{(i)} \in \{0,1\}^{B \times |\mathcal{V}|}$. Here, each column of $\pi^{(i)}$ is a one-hot vector, satisfying $\sum_{j=1}^{B} \pi_{j,k}^{(i)} = 1$ for every column k.
The estimation could be further written in a matrix form:

$$\hat{\mathbf{A}}_{MHF} = \sum_{i=1}^{H} \mathbf{H}_\theta^{(i)} \mathbf{W}_o^{(i)\top} \pi^{(i)} - \sum_{i=1}^{H} \log \left(\sum_{j=1}^{B} \exp(\mathbf{H}_\theta^{(i)} \mathbf{W}_o^{(i)\top}) \right)$$

Consequently, $\text{rank}(\hat{\mathbf{A}}_{\text{MHF}}) \leq \min(n, B, |\mathcal{V}_{actl.}|, H \times \min(d, B)) = \min(B, H \times d)$, where H denotes the number of hash functions configured by MULTIHASHFORMER, B denotes the number of discrete hash buckets per function, and d is the dimensionality of the latent hash state vectors $\mathbf{c}^{(i)}$. Given that the rank of Standard baselines is bounded by $\text{rank}(\hat{\mathbf{A}}_{\text{Standard}}) \leq \min(d, n, |\mathcal{V}_{\text{actl.}}|) = d$, MULTIHASHFORMER elevates the upper bound of this rank to enhance model expressiveness, while employing multi hash functions and non-linear transformation between latent hash state vectors. This aligns with the intuition that aggregating multiple distinct observations inherently improves estimation accuracy.

During inference, although we exclude all unassigned signatures, the rank upper bound remains invariant between training and inference. This stability stems from the fact that the row-wise partition function does not contribute to the final upper bound of the rank. While MultiHashFormer does not produce an arbitrary rank, it elevates the rank of the conditional distribution matrix. This improvement drives performance gains in high-entropy reasoning tasks, including LAMBADA and HellaSwag.

B Rare Word Examples from Card-660

We first look to word pairs with high human-annotated similarity to serve as hard examples for our illustration. Table 5 presents examples of high-similarity word pairs from the Card-660 dataset. We compare normalized human annotation scores against cosine similarities derived from the second-to-last decoder layers of the 1B Standard baseline and MULTIHASHFORMER. Compared to the Standard baseline, MULTIHASHFORMER consistently assigns higher similarity scores to semantically equivalent word pairs, such as abbreviations, aliases, misspellings, and synonyms. Examples from word pairs with low similarities and medium similarities are shown in Table 6.

Word Pair	Human	Stnd.	H3B10K	Stnd.+2L	H4B16K
Abbrevation					
retweeting ↔ RTing	1.00	0.56	0.65	0.47	0.71
science-fiction ↔ sci-fi	1.00	0.77	0.90	0.77	0.86
Brooklyn ↔ Brklyn	1.00	0.45	0.53	0.46	0.61
Internet Explorer 4 ↔ IE4	1.00	0.51	0.61	0.53	0.61
ITV2 ↔ ITV Two	1.00	0.51	0.60	0.43	0.52
electrolytic polishing ↔ electropolishing	0.99	0.74	0.83	0.62	0.87
Alias					
full-HD ↔ 1080p	1.00	0.55	0.72	0.40	0.67
Malva parviflora ↔ cheeseweed	1.00	0.52	0.58	0.57	0.61
first milk ↔ colostrum	1.00	0.54	0.64	0.53	0.69
little bee-eater ↔ Merops pusillus	1.00	0.48	0.60	0.51	0.64
Misspelling					
leggin ↔ legging	1.00	0.59	0.57	0.39	0.60
tariqa ↔ tariqah	1.00	0.77	0.82	0.72	0.82
shit ↔ shxt	0.99	0.17	0.53	0.22	0.52
sweeeet ↔ sweet	0.99	0.36	0.45	0.37	0.52
Synonym					
25 ↔ twenty-five	1.00	0.75	0.81	0.75	0.84
shapeless ↔ amorphous	0.99	0.38	0.50	0.51	0.63
unforeseen ↔ unanticipated	0.97	0.82	0.91	0.83	0.93

Table 5: Examples of word pairs from Card-660 with high human-annotated similarity scores (normalized to [0, 1]), compared against cosine similarities computed from the second-to-last decoder layer of the 1B Standard baselines and MULTIHASHFORMER models.

Word Pair	Human	Stnd.	H3B10K	Stnd.+2L	H4B16K
		Low			
NetMeeting ↔ Marwar Hall	0.00	0.37	0.41	0.39	0.34
Park Ji-sung ↔ Yosemite Park	0.00	0.36	0.49	0.36	0.43
Pizza Hut ↔ Pizzle rot	0.02	0.29	0.25	0.31	0.27
cheddah ↔ cheddar	0.06	0.44	0.23	0.47	0.51
cheddah ↔ cheddar	0.06	0.44	0.23	0.47	0.51
Apple ↔ Applebees	0.10	0.49	0.56	0.49	0.65
		Medium			
Ben-Hur ↔ Titanic	0.39	0.40	0.39	0.27	0.50
radionavigation ↔ frequency band	0.44	0.37	0.44	0.34	0.42
night sky ↔ skyglow	0.49	0.60	0.65	0.60	0.64
circus ↔ ropedancer	0.50	0.46	0.56	0.51	0.47
transmigration ↔ residence permit	0.52	0.44	0.64	0.50	0.57
Head tilt ↔ cervix	0.52	0.35	0.49	0.34	0.49

Table 6: Examples of word pairs from Card-660 with low and medium human-annotated similarity scores (normalized to [0, 1]), compared against cosine similarities computed from the second-to-last decoder layer of the 1B Standard baselines and MULTIHASHFORMER models. A human annotation score of 1.00 indicates semantic equivalence.

C Hyperparameters

C.1 Hyperparameters for Pre-training

Hyperparameters	100M	1B	3B
Hidden size	768	2048	2048
Intermediate size	2560	6144	11008
Max window layers	12	20	36
Added layers (if scaling the depth)	4	2	1
Number of attention heads	12	16	16
Number of hidden layers	12	20	36
Number of key value heads	2	8	2
Rope theta	1M	1M	1M
RMS norm eps	1e-06	1e-06	1e-06
Attention dropout	0.0	0.0	0.0
Tie word embeddings	True	True	True
Hidden activation	SiLU	SiLU	SiLU
Initializer range	0.02	0.02	0.02
Vocabulary size	32,768	32,768	32,768
Tokenizer	Mistral	Mistral	Mistral
Batch size	256	256	256
Train steps	20K	200K	200K
Sequence length	2,048	2,048	2,048
Maximum Learning Rate	5e-4	3e-4	2e-4
Learning rate scheduler	cosine	cosine	cosine
Warmup steps	2000	2000	2000
Optimizer	AdamW	AdamW	AdamW
Adam ϵ	1e-8	1e-8	1e-8
Adam β_1	0.9	0.9	0.9
Adam β_2	0.999	0.999	0.999
Gradient clipping	1.0	1.0	1.0
Weight decay	0.1	0.1	0.1
Training precision	BF16	BF16	BF16

Table 7: Hyperparameters of pre-training at each model scale.

C.2 Hyperparameters for VE Continual-pretraining

Hyperparameters	1B	3B
Adaptive Decoder Layer Indices	[1,2,19,20]	[1,2,35,36]
Attention dropout	0.0	0.0
Tie word embeddings	True	True
Vocabulary size	48,122	48,122
Tokenizer	expanded Mistral	expanded Mistral
Batch size	256	256
Train steps	16K	16K
Sequence length	2,048	2,048
Maximum Learning Rate	3e-4	2e-4
Learning rate scheduler	cosine	cosine
Warmup steps	800	800
Optimizer	AdamW	AdamW
Adam ϵ	1e-8	1e-8
Adam β_1	0.9	0.9
Adam β_2	0.999	0.999
Gradient clipping	1.0	1.0
Weight decay	0.1	0.1
Training precision	BF16	BF16

Table 8: Additional hyperparameters of continual-pretraining at each model scale.

D MultiHashFormer Abbrevations and Configurations

Table 9 presents the detailed configurations of MULTIHASHFORMER variants along with their abbreviations.

MHF abbr.	H	B
H3B4K	3	4,096
H3B8K	3	8,192
H3B10K	3	10,624
H3B16K	3	16,384
H4B4K	4	4,096
H4B8K	4	8,192
H4B16K	4	16,384
H4B32K	3	32,768

Table 9: Detailed configurations of MULTIHASHFORMER variants along with their abbreviations.