SINGLE-LAYER MEMO AS A RANDOMIZED HAMMING-KERNEL CLASSIFIER

Anonymous authors
Paper under double-blind review

ABSTRACT

MEMO (Zanzotto et al., 2025) is a recent language-model architecture that stores associations between token contexts and next tokens in a correlation matrix memory. In this work, we study its single-layer form and show that its ideal retrieval rule is a multiclass classifier based on the *positional Hamming kernel*. The MEMO architecture represents both the sequence features and the output labels with Gaussian random codes. Its score is therefore a doubly randomized sketch of the ideal classifier. Under independent input and output codebooks, we bound the errors introduced by context sketching and output decoding, characterize their dependence on model and data parameters, and give a margin-based guarantee for recovering the ideal prediction. Controlled simulations support the trends predicted by the analysis. On a restricted `WikiText-2` next-token task, we compare single-layer MEMO with classical baselines and show that it can offer a useful trade-off among predictive accuracy, memory, and throughput, particularly on a GPU, where its matrix operations can be parallelized.

1 INTRODUCTION

Language models can benefit from both general patterns and specific associations observed in data. In transformer models, these associations are typically encoded in learned parameters, making them less directly accessible for inspection or editing. This has motivated architectures that make memory an explicit operation (Wu et al., 2022; Berges et al., 2025; Behrouz et al., 2025).

Correlation matrix memories (CMMs) provide a classical mechanism for storing key-value associations in a distributed representation (Anderson, 1972; Kohonen, 1972). Given pairs of vectors (k_j, v_j) that represents keys and values, respectively, a CMM stores the matrix $\sum_j k_j v_j^\top$. Querying the memory with a key q returns a linear combination of the stored values, where the coefficient of v_j is the inner product $\langle q, k_j \rangle$. This simple algebra makes the storage rule transparent and permits local edits by adding or subtracting outer products.

MEMO applies this mechanism to language modeling (Zanzotto et al., 2025). A fixed-length sequence of tokens is encoded as a random vector and used as the key of a CMM, while the next token is represented as its value. At retrieval time, stored next-token vectors are weighted by their similarity to the query sequence and decoded against the token codebook. The original presentation observes that this operation behaves as an approximate counter and assigns partial credit to partially matching contexts. However, this description leaves three questions open. What is the precise learning problem solved by the single-layer model? What geometry does its notion of partial matching induce? Finally, which roles are played by the different dimensions of the random representation?

We answer the first two questions by showing that the single-layer model is a randomized implementation of a *positional Hamming-kernel classifier*. For two contexts of length h, the underlying kernel is $K_H(s, q) = \frac{1}{h} \sum_{r=1}^{h} \mathbf{1}\{s_r = q_r\}$, i.e., the fraction of positions at which the two contexts contain the same token. For every possible next token c, the ideal classifier sums this similarity over all stored contexts whose target is c. The prediction is the class with the largest accumulated similarity. Thus, single-layer MEMO can be understood as a *kernel voting rule*, or equivalently, as a linear classifier whose class weights are unnormalized sums of training features.

MEMO implements this classifier only approximately, using random codes for both context features and output labels. Our main technical contribution is an analysis of this approximation: we bound the errors due to context sketching and output decoding, then combine these bounds with the ideal classifier's margin

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.

to obtain an exact-recovery guarantee. The analysis shows how representation dimensions, context length, and repeated labels affect recovery, thus answering the third question.

Controlled simulations test the predicted behavior. On a restricted `WikiText-2` next-token task, we also compare MEMO with classical baselines to assess its fidelity, predictive accuracy, memory use, and inference throughput.

Related work. As mentioned in the introduction, correlation matrix memories store key-value associations as sums of outer products (Anderson, 1972; Kohonen, 1972), while MEMO applies them to next-token prediction (Zanzotto et al., 2025). We study the classifier implemented by its single-layer retrieval rule. Distributed symbolic codes (Plate, 1995), random projections (Johnson & Lindenstrauss, 1984), and random features for kernel machines (Rahimi & Recht, 2007) provide related compression tools. MEMO sketches both context features and output labels. Cabannes et al. (2024) study capacity and scaling laws for outer-product associative memories with independent random codes for whole inputs, whereas we analyze MEMO's token-based context encoding and its recovery of the positional Hamming-kernel classifier.

Kernel classification provides the general framework (Schölkopf & Smola, 2002). Position-aware weighted-degree kernels have been used for sequence classification (Sonnenburg et al., 2005), and their degree-one case on fixed-length sequences is the positional Hamming kernel up to normalization. Efficient approximations of mismatch string kernels have also been studied for sequence classification (Farhan et al., 2017), while recent work analyzes the expressive limits of Hamming kernels on discrete sequences (Amin et al., 2025). Our focus is different: we identify the count-weighted kernel vote inside MEMO and bound when its randomized implementation preserves the ideal prediction.

2 PRELIMINARIES

Notation and data model. Let $\mathcal{V} = \{1, \dots, n\}$ be a finite vocabulary of n tokens. A context of length h is an element $s = (s_1, \dots, s_h) \in \mathcal{V}^h$. We consider a multiset $\mathcal{D} = \{(s_j, y_j)\}_{j=1}^{M} \subseteq \mathcal{V}^h \times \mathcal{V}$, where y_j is the token observed after context s_j. Since $\mathcal{D}$ is a multiset, repetitions contribute with their multiplicity. Unless explicitly stated otherwise, $\mathcal{D}$ is fixed and all probabilities are taken only over the random initialization of the representations. The Euclidean inner product and norm are denoted by $\langle \cdot, \cdot \rangle$ and $\|\cdot\|$, respectively.

Gaussian codebooks and random projections. We use two token codebooks. The input codebook $U = [u_1, \dots, u_n] \in \mathbb{R}^{d \times n}$ represents tokens appearing in a context, and the output codebook $Z = [z_1, \dots, z_n] \in \mathbb{R}^{d \times n}$ represents next-token labels. In the untied model, $u_x, z_x \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_d/d) \ \ \forall x \in \mathcal{V}$, and U and Z are independent. The tied construction is obtained by setting $Z = U$. The algebraic description of MEMO is the same in both cases, but the independence of U and Z will be relevant when computing expectations.

Standard Gaussian concentration implies that, for every fixed $\varepsilon \in (0, 1)$, the events $\{|\, \|u_x\|^2 - 1| > \varepsilon\}$ and $\{|\, \langle u_x, u_{x'} \rangle \,| > \varepsilon\}$ (with $x' \neq x$) hold with probability at most $2e^{-cd\varepsilon^2}$, for an absolute constant $c > 0$ (analogous bounds hold for Z). A union bound therefore gives uniform approximate orthogonality of a codebook of size n when $d \gtrsim \varepsilon^{-2}\big(\log n + \log(1/\delta)\big)$.

Let d_h be the projection dimension and set $p = hd_h$. We draw a projection $W \in \mathbb{R}^{d_h \times d}$ independently of the codebooks, with entries independently distributed as $\mathcal{N}(0, 1/p)$. This normalization ensures that $\mathbb{E}_W \|Wu\|^2 = \frac{1}{h}\|u\|^2$ for every fixed $u \in \mathbb{R}^d$. Equivalently, $\sqrt{h}W$ is a standard Gaussian Johnson-Lindenstrauss map into $\mathbb{R}^{d_h}$ (Johnson & Lindenstrauss, 1984). Thus, for a fixed codebook and distortion $\zeta \in (0, 1)$, all pairwise distances among its n columns are simultaneously preserved with probability at least $1 - \delta$, provided that $d_h \gtrsim \zeta^{-2}\big(\log n + \log(1/\delta)\big)$.

The positional Hamming kernel. For every pair of sequences $s, q \in \mathcal{V}^h$ we define the *Positional Hamming Kernel* (or *Positional Hamming Similarity*) as

$$K_H(s, q) = \frac{1}{h} \sum_{r=1}^{h} \mathbf{1}\{s_r = q_r\} = 1 - \frac{d_H(s, q)}{h}, \tag{1}$$

where d_H is the Hamming distance. This is a positive-semidefinite kernel. To see this, let $e_x \in \mathbb{R}^n$ be the one-hot vector associated with token x, and define $\Psi(s) = \frac{1}{\sqrt{h}}\big(e_{s_1} \oplus \cdots \oplus e_{s_h}\big) \in \mathbb{R}^{hn}$, where $\oplus$

indicates the concatenation operator. Then $\langle \Psi(s), \Psi(q) \rangle = K_H(s, q)$. The feature map records the token appearing at each position separately. It does not contain joint features involving two or more positions.

3 Problem Formulation

For every candidate next token $c \in \mathcal{V}$ and query context $q \in \mathcal{V}^h$, we define the *class score*

$$S_c(q) = \sum_{j:y_j=c} K_H(q, s_j).$$

The multiplicity of a pair in $\mathcal{D}$ is retained in the sum. We study the following prediction problem.

Problem 1 (Hamming-kernel next-token classification)**.** Given a multiset $\mathcal{D} \subseteq \mathcal{V}^h \times \mathcal{V}$ and a query $q \in \mathcal{V}^h$, compute

$$c^*(q) \in \arg\max_{c \in \mathcal{V}} S_c(q). \tag{2}$$

If the maximum is not unique, we can break the tie arbitrarily.

The problem admits two complementary interpretations. First, it is a *kernel voting rule*: every stored example votes for its label with weight equal to its similarity to the query. Second, it is a *linear classification problem*. We have a class c for every token in $\mathcal{V}$, and for each class consider the unnormalized class prototype $\mu_c = \sum_{j:y_j=c} \Psi(s_j)$. Since $\langle \Psi(s), \Psi(q) \rangle = K_H(s, q)$, we have $S_c(q) = \sum_{j:y_j=c} \langle \Psi(q), \Psi(s_j) \rangle = \Psi(q)^\top \sum_{j:y_j=c} \Psi(s_j) = \langle \Psi(q), \mu_c \rangle$. The prototypes are unnormalized because MEMO is designed to accumulate occurrence counts. Dividing μ_c by the number of examples in class c would instead produce an average-similarity classifier and would remove the empirical class-frequency factor, that is a different prediction rule.

The additive structure of K_H gives an alternative expression. Let $N_{c,r}(x) = |\{j : y_j = c,\ (s_j)_r = x\}|$ be the number of training examples with label c and token x at position r. Then we can write

$$S_c(q) = \frac{1}{h} \sum_{r=1}^{h} N_{c,r}(q_r). \tag{3}$$

Consequently, the ideal classifier is completely determined by the token–position–label count tables. This identity will later provide both a geometric interpretation and a useful baseline against which the randomized MEMO implementation can be compared.

4 Single-Layer MeMo

Sequence encoding. The randomized encoding of a context $s = (s_1, \ldots, s_h)$ is the vertical concatenation $\phi(s) = [W u_{s_1} \cdots W u_{s_h}]^\top \in \mathbb{R}^p$, with $p = hd_h$. The block structure preserves position: the encoding of the first token and that of the second token occupy disjoint coordinates, even though the same projection W is applied to both.

Lemma 1. *For every $s, q \in \mathcal{V}^h$, we have $\mathbb{E} \langle \phi(s), \phi(q) \rangle = K_H(s, q)$.*

Proof. Using the block structure, and conditioning on the input codebook U, $\mathbb{E} \langle \phi(s), \phi(q) \rangle = \sum_{r=1}^{h} \mathbb{E} \langle W u_{s_r}, W u_{q_r} \rangle = \frac{1}{h} \sum_{r=1}^{h} \mathbb{E} \langle u_{s_r}, u_{q_r} \rangle$. The Gaussian codebook satisfies $\mathbb{E} \langle u_x, u_{x'} \rangle = \mathbf{1}\{x = x'\}$. The lemma follows by substituting the sum with the definition of Hamming Kernel from equation (1). □

Thus, the random map ϕ is a finite-dimensional randomized feature map for the positional Hamming kernel. Its approximation error is controlled jointly by the coherence of the input codebook U, and by the distortion introduced by W.

Memorization. Single-layer MEMO stores every association (s_j, y_j) as an outer product between its randomized context key and its output code. Therefore, the correlation matrix memory is $C = \sum_{j=1}^{M} \phi(s_j) z_{y_j}^\top \in \mathbb{R}^{p \times d}$. We can add a new observation (or removing an existing one) (s, y) simply by adding or subtracting one term $\phi(s) z_y^\top$. The CMM contains $pd = hd_h d$ scalar parameters. When $d_h = d/h$, one has $p = d$ and the memory is square (the definition above also covers the rectangular memories).

Retrieval and decoding. Given a query context $q \in \mathcal{V}^h$, the retrieved output vector is $r(q) = C^\top \phi(q) \in \mathbb{R}^d$. The score assigned to candidate token c is obtained by decoding against its output code, i.e.,

$$\widetilde{S}_c(q) = \langle z_c, r(q) \rangle = \sum_{j=1}^{M} \langle \phi(q), \phi(s_j) \rangle \left\langle z_c, z_{y_j} \right\rangle . \tag{4}$$

Finally, single-layer MEMO predicts $\widehat{c}(q) \in \arg\max_{c \in \mathcal{V}} \widetilde{S}_c(q)$.

Equation (4) displays two distinct sketches. The factor $\langle \phi(q), \phi(s_j) \rangle$ approximates positional Hamming similarity in the sequence space, while $\left\langle z_c, z_{y_j} \right\rangle$ approximates the indicator that the stored label is c. If both families of vectors were exactly orthonormal, then $\widetilde{S}_c(q) = S_c(q)$ and MEMO would solve equation (2) exactly. The next lemma proves that $\widetilde{S}_c(q)$ is an unbiased estimator of $S_c(q)$.

Lemma 2. *Suppose that the input codebook U, output codebook Z, and projection W are mutually independent. Then, for every fixed dataset $\mathcal{D}$, query q, and candidate token c, $\mathbb{E}\widetilde{S}_c(q) = S_c(q)$.*

Proof. The independence of Z from (U, W) and linearity of expectation give $\mathbb{E}\widetilde{S}_c(q) = \sum_{j=1}^{M} \mathbb{E} \langle \phi(q), \phi(s_j) \rangle \mathbb{E} \left\langle z_c, z_{y_j} \right\rangle = \sum_{j=1}^{M} K_H(q, s_j) \mathbf{1}\{c = y_j\} = S_c(q)$, where we used Lemma 1 and the identity $\mathbb{E} \langle z_c, z_y \rangle = \mathbf{1}\{c = y\}$. □

Remark 2 (Tied input and output representations). The tied construction $Z = U$ uses fewer random objects and corresponds to using the same token representation on both sides of the memory. In this case, equation (4) remains exact, but the two inner products in each summand are generally dependent. Hence Lemma 2 cannot be invoked directly: repeated occurrences of output tokens inside contexts can produce bias terms. Separating this bias from random cross-talk is an essential step in the analysis of the tied architecture.

The difference $\widetilde{S}_c(q) - S_c(q)$ is the retrieval error induced by the random sketch. Correct next-token prediction depends not only on its absolute size, but on the *margin* of the ideal classifier, i.e.,

$$\Delta(q) = S_{c^*(q)}(q) - \max_{c \neq c^*(q)} S_c(q).$$

A subsequent capacity analysis can therefore be organized in two steps: first obtain simultaneous bounds on the score errors, and then compare those bounds with $\Delta(q)$. This separates the intrinsic ambiguity of the ideal classification problem from the approximation error and cross-talk introduced by the randomized memory representation.

5 ANALYSIS

The model approximates the ideal score in two successive stages: the retrieval and the decoding. For every label c, we define the score after the context sketch but before output decoding,

$$G_c(q) = \sum_{j : y_j = c} \langle \phi(q), \phi(s_j) \rangle . \tag{5}$$

We can think of $G_c(q)$ as the overall contribution of class c with respect to query q. The passage from $S_c(q)$ to $G_c(q)$ contains only the approximation of the Hamming kernel. Grouping equation (4) by the stored label then gives

$$\widetilde{S}_c(q) = \sum_{y \in \mathcal{V}} G_y(q) \langle z_c, z_y \rangle . \tag{6}$$

Hence, the *total score error* can be decomposed in the following two elementary

$$\widetilde{S}_c(q) - S_c(q) = \big(G_c(q) - S_c(q)\big) + \big(\widetilde{S}_c(q) - G_c(q)\big). \tag{7}$$

We will refer to the first difference as *context-sketch error*, and to the second one as *output-code interference*. Intuitively $|G_c(q) - S_c(q)|$ measures how accurately are contexts represented by the random map ϕ, while $|S_c(q) - G_c(q)|$ indicates how much interference is introduced during decoding phase.

Thus, the analysis will be organized in four steps: in the remainder of this section we first determine how much score error the ideal prediction can tolerate, then bound the context-sketch error and

output-code interference separately (Sections 5.1 and 5.2), and finally combine the two bounds to obtain an exact-recovery guarantee (Section 5.3).

We now proceeds with the first point of the analysis. Without loss of generality, assume that the ideal classifier has a unique winner $c^* = c^*(q)$. For every competitor $c \neq c^*$, we define

$$\Delta_c(q) = S_{c^*}(q) - S_c(q), \qquad \Delta(q) = \min_{c \neq c^*} \Delta_c(q) = S_{c^*}(q) - \max_{c \neq c^*} S_c(q).$$

Thus $\Delta(q)$ is the gap between the largest and second-largest ideal scores.

Theorem 3. *Let $\varepsilon_{\text{score}}(q) = \max_{c \in \mathcal{V}} |\widetilde{S}_c(q) - S_c(q)|$. If $\varepsilon_{\text{score}}(q) < \Delta(q)/2$, then single-layer* MEMO *and the ideal Hamming-kernel classifier predict the same token, i.e., $\widehat{c}(q) = c^*(q)$.*

Proof. Fix a competitor $c \neq c^*$. By definition, we have $\widetilde{S}_{c^*}(q) \geq S_{c^*}(q) - \varepsilon_{\text{score}}(q)$, and $\widetilde{S}_c(q) \leq S_c(q) + \varepsilon_{\text{score}}(q)$. Subtracting the second inequality from the first yields $\widetilde{S}_{c^*}(q) - \widetilde{S}_c(q) \geq S_{c^*}(q) - S_c(q) - 2\varepsilon_{\text{score}}(q) = \Delta_c(q) - 2\varepsilon_{\text{score}}(q) \geq \Delta(q) - 2\varepsilon_{\text{score}}(q) > 0$. Thus the randomized score of c^* is strictly larger than the score of every competitor, and the two classifiers have the same argmax. □

The theorem separates two causes of prediction failure. A small $\Delta(q)$ is an ambiguity of the ideal Hamming-kernel problem, while a large score error is a limitation of the randomized representation. By equation (7), it remains to control the context sketch and the output decoding separately.

5.1 CONTEXT-SKETCH APPROXIMATION

We now control the first difference in equation (7). We first state the result for an arbitrary per-position dimension d_h, and then we specialize it to the standard square-memory choice $d_h = d/h$. The argument uses only one idea: the input codebook U and the projection W must jointly preserve every inner product between two token codes.

For each label c, let $M_c = |\{j : y_j = c\}|$ be the number of observation in $\mathcal{D}$ that has c as next token, and $M_{\max} = \max_c M_c$. Observe that $\sum_c M_c = M$, and $M_{\max} \leq M$.

Theorem 4 (Context-sketch approximation)**.** *For $\delta \in (0,1)$, we define $L_\delta = \ln\left(\frac{2n}{\sqrt{\delta}}\right)$. There exist universal constants $c_0, C_0 > 0$ such that, if $d \geq c_0 L_\delta$, then with probability at least $1 - \delta$ over (U, W), simultaneously for every query $q \in \mathcal{V}^h$,*

$$\max_{c \in \mathcal{V}} |G_c(q) - S_c(q)| \leq \mathcal{B}_{\text{ctx}}(\delta) := C_0 M_{\max} \left(\sqrt{\frac{L_\delta}{d}} + \frac{L_\delta}{d} + \sqrt{\frac{L_\delta}{d_h}} + \frac{L_\delta}{d_h} \right).$$

Proof sketch. Let $A = \sqrt{h}\, W$. Gaussian concentration for the input codes and, conditionally on them, for the shared projection shows that all projected token inner products $\langle Au_x, Au_y \rangle$ approximate $\mathbf{1}\{x = y\}$ uniformly, with error at most η of the stated order. Since the similarity between two contexts averages these inner products over their h positions, it differs from their positional Hamming similarity by at most η. Summing over the M_c stored contexts with label c gives $|G_c(q) - S_c(q)| \leq M_c \eta \leq M_{\max} \eta$. The token-pair bound is uniform, so this conclusion holds for every query and label simultaneously. The detailed proof is deferred to Appendix A.1. □

Square-memory specialization. For the standard MEMO choice $d_h = d/h$ (i.e., with a squared memory matrix), we have $d_h \leq d$, so the terms that depends on d_h dominate, therefore $\mathcal{B}_{\text{ctx}}(\delta) = M_{\max} O\left(\sqrt{\frac{hL_\delta}{d}} + \frac{hL_\delta}{d}\right)$. When $d \gg h \log(n/\delta)$, the square-root term dominates, and $d \gtrsim h \log(n/\delta)/\varepsilon^2$ suffices to make this bound at most $\varepsilon M_{\max}$. When $d \lesssim h \log(n/\delta)$, the bound does not guarantee a small relative error, although the actual error may still be small. In the worst case $M_{\max} = M$. The linear dependence on $M_{\max}$ is unavoidable for a uniform result, because duplicated contexts repeat the same kernel error coherently. However, the proof of Theorem 4 is pessimistic because it bounds every token-position comparison by the same worst-case error η. In practice, different token-position pairs may produce errors of different magnitudes and signs. A more refined, data-dependent analysis could exploit their repetition frequencies and possible cancellations, potentially leading to tighter bounds.

5.2 Output-Code Interference

We now analyze the second difference in equation (7). Fix the input codebook U, the projection W, the dataset, and the query. Under this conditioning, the values $G_y(q)$ are deterministic and the only remaining randomness is in the independent Gaussian output codes Z. From equation (6)

$$\widetilde{S}_c(q) - G_c(q) = G_c(q)\big(\|z_c\|^2 - 1\big) + \sum_{y \neq c} G_y(q)\,\langle z_c, z_y\rangle\,. \tag{8}$$

The first term is the *norm error* of the candidate code z_c, while the second is the *cross-talk* from the competing output codes.

We now introduce the quantity $\widehat{\mathcal{C}}_H(q) = \sum_{y\in\mathcal{V}} G_y(q)^2$, that is the squared norm of the score vector before the decoding phase in equation (6). It measures the total energy of the evidence assigned by the query to the output labels and therefore controls the amount of interference caused by their non-orthogonal codes. In particular, the typical cross-talk scale is of order $\sqrt{\widehat{\mathcal{C}}_H(q)/d}$. Observe that this is a quantity determined by the data and the query.

Theorem 5 (Output-code interference)**.** *Let $\delta \in (0,1)$ and set $t = \ln\left(\frac{4n}{\delta}\right)$, $\alpha = 2\sqrt{\frac{t}{d}} + \frac{2t}{d}$. Conditionally on (U, W), with probability at least $1-\delta$ over the output codebook Z, every candidate $c \in \mathcal{V}$ satisfies*

$$|\widetilde{S}_c(q) - G_c(q)| \le \mathcal{B}_{\mathrm{out}}(q,\delta) := \sqrt{\widehat{\mathcal{C}}_H(q)}\left(\alpha + \sqrt{2t(1+\alpha)/d}\right).$$

Proof sketch. Condition on the input codebook and projection, so the scores $G_y(q)$ are fixed. For each candidate c, the decoding error is the sum of a code-norm error and cross-talk from the other labels. The latter is Gaussian conditional on z_c, with variance $\frac{\|z_c\|^2}{d}\sum_{y\neq c} G_y(q)^2$. Gaussian and chi-square concentration bound the two terms. Both $|G_c(q)|$ and $\left(\sum_{y\neq c} G_y(q)^2\right)^{1/2}$ are at most $\sqrt{\widehat{\mathcal{C}}_H(q)}$, then applying a union bound over all candidates gives the stated result. The detailed proof is deferred to Appendix A.2. □

5.3 Full Prediction Guarantee

The two concentration results from Sections 5.1 and 5.2 can now be combined without assuming independence between individual stored examples.

Theorem 6 (Exact recovery of the ideal prediction)**.** *Assume the untied model, with arbitrary positive integer d_h. Fix a dataset and query, suppose that the ideal winner $c^*(q)$ is unique, and let $\delta \in (0,1)$. If $d \ge c_0 L_{\delta/2}$ (as defined in Theorem 4), then, with probability at least $1-\delta$ over (U, W, Z),*

$$\varepsilon_{\mathrm{score}}(q) \le \mathcal{B}_{\mathrm{ctx}}\left(\tfrac{\delta}{2}\right) + \mathcal{B}_{\mathrm{out}}\left(q, \tfrac{\delta}{2}\right). \tag{9}$$

On the same event, if $\mathcal{B}_{\mathrm{ctx}}\left(\frac{\delta}{2}\right) + \mathcal{B}_{\mathrm{out}}\left(q, \frac{\delta}{2}\right) < \frac{\Delta(q)}{2}$, then $\widehat{c}(q) = c^*(q)$.

Proof. Apply Theorem 4 with failure probability $\delta/2$. On the resulting event, $\max_c |G_c(q) - S_c(q)| \le \mathcal{B}_{\mathrm{ctx}}\left(\frac{\delta}{2}\right)$. For every fixed realization of (U, W), Theorem 5, again with failure probability $\delta/2$, gives $\max_c |\widetilde{S}_c(q) - G_c(q)| \le \mathcal{B}_{\mathrm{out}}\left(q, \frac{\delta}{2}\right)$. A union bound and equation (7) prove equation (9). The prediction statement follows from Theorem 3. □

Theorem 6 is the main recovery theorem. It separates the context error, controlled by $(d, d_h, M_{\max})$, from output interference, controlled by d and the distribution of the class scores.

If the context sketch were exact, then $G_y(q) = S_y(q)$ and $\widehat{\mathcal{C}}_H(q) = \mathcal{C}_H(q) = \sum_{y\in\mathcal{V}} S_y(q)^2$. Intuitively, suppose that M stored contexts all have Hamming similarity ρ with the query. If their labels are distinct, then $\mathcal{C}_H(q) = M\rho^2$, while if they all have the same label $\mathcal{C}_H(q) = M^2\rho^2$. In general, if k stored contexts share the same label y, their contributions are first aggregated into $S_y(q) = k\rho$. For any competing candidate $c \neq y$, they therefore produce the single cross-talk term $k\rho\langle z_c, z_y\rangle$, whose variance scales as $k^2\rho^2/d$, since $\mathrm{Var}(\langle z_c, z_y\rangle) = 1/d$. This quadratic dependence on k reflects the coherent accumulation of repeated labels. Consequently, output interference depends on how the total score is distributed across labels, as captured by $\mathcal{C}_H(q) = \sum_y S_y(q)^2$, rather than on memory size alone.

A square-memory recovery condition. The output bound contains the realized collision energy $\widehat{\mathcal{C}}_H(q)$. It can be replaced by ideal, count-only quantities. For readability, we give this explicit specialization under $d_h = d/h$. Let $\ell_{\mathcal{D}} = \big|\{y_j : 1 \le j \le M\}\big|$ be the number of distinct labels that actually occur in the memory[1].

Corollary 7. *Assume the untied square-memory model, so that h divides d and $d_h = d/h$, and suppose that the ideal winner is unique. Fix $\delta \in (0,1)$, and set $L = \ln\big(\frac{16n}{\delta}\big)$, $\alpha = 2\sqrt{\frac{L}{d}} + \frac{2L}{d}$, and $\gamma = \sqrt{2L(1+\alpha)/d}$. If $d \ge c_0 L$, we define*

$$\beta_{\mathrm{ctx}} = C_0 M_{\max}\Big(\sqrt{\tfrac{hL}{d}} + \tfrac{hL}{d}\Big), \qquad \beta_{\mathrm{out}} = \Big(\sqrt{\mathcal{C}_H(q)} + \sqrt{\ell_{\mathcal{D}}}\beta_{\mathrm{ctx}}\Big)(\alpha+\gamma),$$

for some absolute constants $c_0, C_0 > 0$. Then, with probability at least $1-\delta$, we have that $\varepsilon_{\mathrm{score}}(q) \le \beta_{\mathrm{ctx}} + \beta_{\mathrm{out}}$. Consequently, if the $\beta_{\mathrm{ctx}} + \beta_{\mathrm{out}} < \Delta(q)/2$, then $\widehat{c}(q) = c^*(q)$.

Proof. Apply Theorem 4 with failure probability $\delta/2$, so that the context error is at most β_{ctx}. Only the $\ell_{\mathcal{D}}$ observed labels have nonzero scores, hence

$$\sqrt{\widehat{\mathcal{C}}_H(q)} = \|(G_y(q))_{y\in\mathcal{V}}\|_2 \le \sqrt{\mathcal{C}_H(q)} + \sqrt{\ell_{\mathcal{D}}}\,\beta_{\mathrm{ctx}}. \tag{10}$$

We now apply Theorem 5 conditionally on (U, W), with failure probability $\delta/2$. Its tail parameter is at most L, so equation (10) bounds the output error by β_{out}. Finally, a union bound and equation (7) prove the score bound. The prediction claim follows from Theorem 3. □

Remark 8 (Readable square-memory asymptotics). Since $K_H \in [0,1]$, the ideal collision energy and the number of observed labels satisfy $\mathcal{C}_H \le M M_{\max}$, and $\ell_{\mathcal{D}} \le \min\{n, M\}$. For $d \gg hL$, the two terms β_{ctx} and β_{out} in the bound have the asymptotic scales $\beta_{\mathrm{ctx}} = O\big(M_{\max}\sqrt{hL/d}\big)$, and $\beta_{\mathrm{out}} = O\big(\sqrt{M M_{\max}}\sqrt{L/d} + \sqrt{\ell_{\mathcal{D}}}\, M_{\max}\sqrt{h}L/d\big)$. The second term in β_{out} is the interaction between context and output distortions. If $d \gg \ell_{\mathcal{D}} L$, it is of lower order in the total error, and then the error simplifies as $\varepsilon_{\mathrm{score}}(q) = O\big((M_{\max}\sqrt{h} + \sqrt{M M_{\max}})\sqrt{L/d}\big)$. In the worst case $M_{\max} = M$, this reduces to $\varepsilon_{\mathrm{score}}(q) = O\big(M\sqrt{hL/d}\big)$. Ignoring constants, the corresponding recovery requirement is $d \gtrsim \frac{L(hM_{\max}^2 + M M_{\max})}{\Delta(q)^2}$.

In this bound, the vocabulary size n appears only inside a logarithm. For a square memory, i.e., $d_h = d/h$, so d grows linearly with h if d_h is kept fixed. The term $\sqrt{M M_{\max}}$ reflects the fact that contexts with the same label add their contributions before output decoding. Finally, the recovery condition is easier to satisfy when the ideal margin $\Delta(q)$ is larger, because the prediction is more robust to MEMO's approximation errors.

6 Experimental Evaluation

6.1 Synthetic Validation

In this section we perform a first synthetic validation of the two-stage analysis of Section 5. More precisely, we use a controlled synthetic experiment to test separately the context-sketch approximation, output-code interference, and end-to-end recovery.

Synthetic datasets. We use $n = 1024$ and $M = 4096$. For each $h \in \{4, 8, 16\}$, the original dataset has 64 active labels with 64 contexts each. It is constructed so that the ideal winner scores 40 and every other active label scores 24, giving a margin of $\Delta(q) = 16$. For $h = 8$, we also merge the labels into 16 balanced groups. This increases $M_{\max}$ from 64 to 256 and the collision energy $C_H(q)$ by about a factor of four, while preserving the ideal winner and margin. The detailed construction is discussed in Appendix B.

Experimental protocol. We use the square-memory choice $d_h = d/h$, and evaluate $d \in \{128, 256, 512, 1024, 2048, 4096\}$. For each setting, the dataset and query remain fixed while the Gaussian representations are resampled. Figure 1 measures the two sources of error separately before testing their combined effect on the prediction. To isolate output decoding in the center panel, we set $G(q) = S(q)$, so the decoder receives the exact ideal scores instead of the approximate context scores. This removes the context-sketch error and allows us to measure *only* the output-code interference

[1] Observe that $\ell_{\mathcal{D}} \le n$

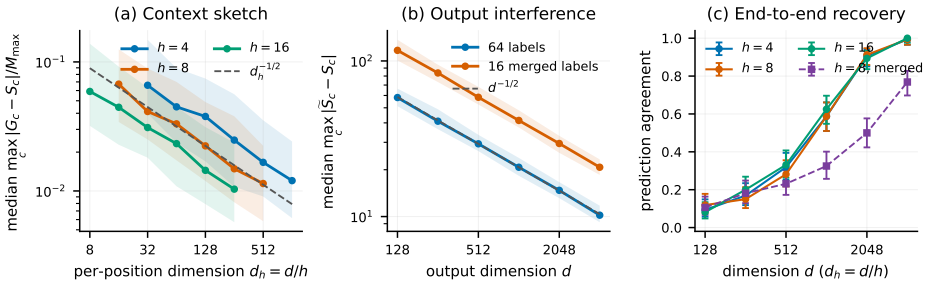

Figure 1: Controlled synthetic validation with $n = 1024$, $M = 4096$, and 160 independent realizations per point. Curves in the left and center panels show medians, with 10th-90th percentile bands, while error bars on the right are pointwise Wilson 95% intervals. **Left:** normalized context-sketch error for the original datasets with $h \in \{4, 8, 16\}$. **Center:** output-decoding error from the exact ideal scores for the original and merged assignments. **Right:** agreement of the complete untied model with the ideal classifier, including the merged $h = 8$ instance.

Results. The left panel is consistent with the predicted $d_h^{-1/2}$ decay in the well-resolved regime. In the center panel, merging the labels increases $\mathcal{C}_H(q)$ by approximately a factor of four and the typical output error by approximately a factor of two, consistently with the predicted $\sqrt{\mathcal{C}_H(q)/d}$ scale. End-to-end agreement increases with d and tends to one for the original assignment, while the merged instance recovers more slowly, despite exhibiting the same ideal margin. These observations support the qualitative scalings predicted by the analysis, but do not constitute a real-world data benchmark. This assessment is addressed in Section 6.2.

6.2 Real-Data Evaluation

In this section we ask whether the same picture remains useful on real sequential data. The experiment keeps three questions separate: (i) prediction of the true next token, (ii) agreement with the exact Hamming-kernel classifier, and (iii) computational cost. The second quantity measures the quality of the random approximation, although it is not the same as accuracy against the observed next token.

Dataset and compared methods. We evaluate a restricted 64-class next-token task on `WikiText-2` (Merity et al., 2016), using 100,000 training, 2,000 validation, and 5,000 test examples. These classes cover 44.79% of eligible test positions, so this is not a full language-model benchmark. For $h \in \{4, 8, 16\}$, we compare untied square-memory MEMO at $d \in \{128, 256, 512, 1024\}$. The baselines are the exact sum prototype, mean prototype, count-based suffix backoff, linear SVM, and Hamming k-NN. Our primary measure of sketch fidelity is agreement with the exact sum prototype on queries with a unique ideal winner, while predictive metrics are measured against the observed next token. Full preprocessing and method settings are given in Appendix C.

Table 1: `WikiText-2` results for $h = 8$. All entries, except agreement, are measured against the observed next token. Agreement is with the exact sum-prototype prediction on queries having a unique ideal winner. MEMO values are mean $\pm$ standard deviation over five codebook seeds.

Method	Top-1	Top-5	Macro-F1	Agreement
Sum prototype	22.30	53.44	3.36	100.00
Mean prototype	9.84	29.58	6.68	9.87
Suffix backoff	31.96	63.32	19.15	37.60
Linear SVM	34.28	60.96	18.80	38.06
Hamming 25-NN	25.66	55.62	9.01	53.54
MEMO, $d = 128$	16.35 ± 0.79	40.48	2.63	47.22 ± 6.07
MEMO, $d = 256$	18.76 ± 1.27	46.35	2.94	54.70 ± 3.16
MEMO, $d = 512$	20.18 ± 0.40	48.66	3.23	63.46 ± 1.89
MEMO, $d = 1024$	21.50 ± 0.76	50.99	3.30	72.79 ± 4.55

Results. The linear SVM gives the highest top-1 accuracy, followed closely by the suffix model. This is not a contradiction: these baselines learn a discriminative rule or preserve token conjunctions, whereas the ideal single-layer MEMO classifier is an additive, count-weighted Hamming rule. The poor top-1 accuracy of the mean prototype also shows that class frequency is an important part of the exact sum-prototype prediction on this task.

For MEMO, increasing d consistently improves fidelity to the exact classifier (the full curves, are reported in the Appendix D). At $h = 8$, mean agreement rises from 47.22% at $d = 128$ to 72.79% at $d = 1024$. At the latter dimension, top-1 accuracy is 21.50%, only 0.80 percentage points below the exact sum prototype (Table 1). The deployed two-stage MEMO state (the projected input codes, CMM, and output codes) occupies 20.50 MiB, compared with 65.00 MiB for the dense exact count table, about 68% less memory (Table 2). At $d = 512$, MEMO uses 9.25 MiB and obtains 20.18% top-1 accuracy and 63.46% agreement. Thus stronger compression gives a visible but gradual loss of fidelity rather than an abrupt failure.

GPU inference. We start observing that the two matrix products in MEMO's retrieval rule can be combined before inference. If Z has the output codes as columns, then we can write $\widetilde{S}(q) = \phi(q)^\top (CZ)$. The matrix CZ is computed once, so each query needs only one matrix product after its features are formed. We will refer to this implementation as *fused* MEMO, and gives the same scores as the two-stage retrieval (up to floating-point rounding). We empathize that it does not remove the output-code interference analyzed above.

We benchmark both versions on an NVIDIA GeForce RTX 4060 Laptop GPU at $h = 8$, using the same device for the sum prototype, mean prototype, and linear SVM. The inputs and model state are already loaded on the GPU. Each measurement includes the time needed to compute the scores and choose a label on the GPU, but excludes training, data transfers, and Python overhead. We cycle through batches of test queries. We run 20 batches to warm up the GPU, then time 100 batches. We repeat this three times and report the median result. The state sizes in Table 2 count the tensors required for inference, not peak GPU allocation.

Table 2: Inference on the RTX 4060 Laptop GPU at $h = 8$. Throughput is in millions of queries per second (Mq/s), while B is batch size.

Method	State (MiB)	$B = 1$ (μs)	$B = 256$ (Mq/s)	$B = 1024$ (Mq/s)
Sum prototype	65.00	18.59	11.36	33.33
Mean prototype	65.00	22.66	9.62	31.08
Linear SVM	65.00	26.35	8.33	29.41
MEMO fused, $d = 128$	2.06	20.22	12.56	48.41
MEMO fused, $d = 256$	4.12	20.21	11.90	40.00
MEMO fused, $d = 512$	8.25	20.19	11.90	27.78
MEMO fused, $d = 1024$	16.50	20.16	9.34	16.37
MEMO two-stage, $d = 1024$	20.50	25.31	2.91	3.21

At batch size 1024, fused MEMO with $d = 256$ processes 40.00 million queries per second, compared with 33.33 million for the exact sum prototype, while using 4.12 MiB rather than 65.00 MiB. Its mean top-1 accuracy is 18.76%, versus 22.30% for the exact classifier. With $d = 512$, accuracy rises to 20.18% and the state remains 7.9 times smaller, but throughput falls to 27.78 million queries per second. At $d = 1024$, MEMO comes closer to the exact prediction but becomes slower still. The literal two-stage implementation is substantially slower at this dimension, which shows the practical value of the algebraic fusion when only class scores are needed. The linear SVM has the highest predictive accuracy (34.28%), so MEMO does not dominate the alternatives on every metric.

The CPU measurements in Figure 2 of Appendix D, and the GPU measurements above, describe different hardware and should not be compared as a speedup.

Overall, MEMO can offer a favorable trade-off between accuracy, memory, and throughput thanks to a compressed representation and highly parallelizable matrix operations. The present experiment illustrates that trade-off, but does not establish universal superiority over either the exact count table or a discriminatively trained classifier.

AI USE STATEMENT

In this work, we used generative AI tools for drafting sections of the paper, retrieval and discovery of related literature, and writing source code. We have *not* used generative AI tools for deriving mathematical proofs, generating experimental results, or producing figures, and the direct generation of synthetic datasets via AI is not applicable to this work. Additionally, we used generative AI tools to aid and polish our writing. We have reviewed all AI-assisted work. Specifically, all AI-suggested related works were manually verified through an independent literature survey to ensure relevance and accuracy. LLM-generated code was thoroughly reviewed, tested for correctness, and validated against our theoretical models. We carefully read and revised all AI-drafted and polished text to ensure it strictly conveyed our intended scientific claims without altering the theoretical substance. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Complete proofs are given in Appendix A. Appendices B and C describe the datasets and experimental protocols, while Appendix D contains additional results. The supplementary material provides scripts to construct the datasets, rerun the experiments, and regenerate the figures and tables, together with fixed seeds, dependencies, and reference CSVs.

REFERENCES

Alan N. Amin, Debora S. Marks, and Eli N. Weinstein. Biological sequence kernels with guaranteed flexibility. *Journal of Machine Learning Research*, 26(216):1–63, 2025. URL http://jmlr.org/papers/v26/23-0455.html.

James A. Anderson. A simple neural network generating an interactive memory. *Mathematical Biosciences*, 14(3–4):197–220, 1972.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 113506–113543. Curran Associates, Inc., 2025. doi: 10.52202/085713-3786. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/ea4ca07aa108036f80cbb5b82285fd4b1-Paper-Conference.pdf.

Vincent-Pierre Berges, Barlas Oguz, Daniel Haziza, Wen-Tau Yih, Luke Zettlemoyer, and Gargi Ghosh. Memory layers at scale. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 3831–3842. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/berges25a.html.

Vivien Cabannes, Elvis Dohmatob, and Alberto Bietti. Scaling laws for associative memories. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 9718–9747, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/28f734334f9f325b515933e70b6b412c-Paper-Conference.pdf.

Muhammad Farhan, Juvaria Tariq, Arif Zaman, Mudassir Shabbir, and Imdad Ullah Khan. Efficient approximation algorithms for string kernel based sequence classification. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6938–6948, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in Modern Analysis and Probability*, volume 26 of *Contemporary Mathematics*, pp. 189–206. American Mathematical Society, 1984.

Teuvo Kohonen. Correlation matrix memories. *IEEE Transactions on Computers*, C-21(4):353–359, 1972.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016. URL https://arxiv.org/abs/1609.07843.

Tony A. Plate. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3): 623–641, 1995.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20*, pp. 1177–1184. Curran Associates, Inc., 2007.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

Sören Sonnenburg, Gunnar Rätsch, and Bernhard Schölkopf. Large scale genomic sequence svm classifiers. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pp. 848–855, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931805. doi: 10.1145/1102351.1102458. URL https://doi.org/10.1145/1102351.1102458.

Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge University Press, 2018.

Yuhuai Wu, Markus Norman Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. In *International Conference on Learning Representations*, 2022.

Fabio Massimo Zanzotto, Elena Sofia Ruzzetti, Giancarlo A. Xompero, Leonardo Ranaldi, Davide Venditti, Federico Ranaldi, Cristina Giannone, Andrea Favalli, and Raniero Romagnoli. Position paper: MeMo: Towards language models with associative memory mechanisms. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 15169–15180. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.785.

A Deferred proofs

A.1 Proof of Theorem 4

Proof. Set the matrix $A = \sqrt{h}W$, so that A is a standard Gaussian projection into $\mathbb{R}^{d_h}$. We define the largest error on a token pair by $\eta = \max_{x,y\in\mathcal{V}} |\langle Au_x, Au_y\rangle - \mathbf{1}\{x = y\}|$. Let $\eta_U = \max_{x,y\in\mathcal{V}} |\langle u_x, u_y\rangle - \mathbf{1}\{x = y\}|$, and $\eta_A = \max_{x,y\in\mathcal{V}} |\langle Au_x, Au_y\rangle - \langle u_x, u_y\rangle|$. By triangular inequality $\eta \le \eta_U + \eta_A$. For $x = y$, the inner product $\langle u_x, u_x\rangle = \|u_x\|^2$ follows the distribution $d^{-1}\chi^2$. We apply the standard chi-square concentration bound with parameter $2L_\delta$, obtaining

$$\mathbb{P}\left(\left|\|u_x\|^2 - 1\right| > 2\sqrt{\frac{2L_\delta}{d}} + \frac{4L_\delta}{d}\right) = \mathbb{P}\left(\left|d\,\|u_x\|^2 - d\right| > 2\sqrt{2dL_\delta} + 4L_\delta\right) \le 2e^{-2L_\delta}.$$

For $x \neq y$, the inner product $\langle u_x, u_y\rangle$ is a sum of independent centered sub-exponential random variables. By Bernstein's inequality (Vershynin, 2018, Corollary 2.9.2) we have

$$\mathbb{P}\left(\left|\langle u_x, u_y\rangle\right| > C_1\left[\sqrt{\frac{2L_\delta}{d}} + \frac{2L_\delta}{d}\right]\right) \le 2e^{-2L_\delta},$$

for some absolute constant $C_1 > 0$. By union bound over all pairs of tokens $x, y \in \mathcal{V}$ we have that $\eta_U \le C_2\left[\sqrt{\frac{L_\delta}{d}} + \frac{L_\delta}{d}\right]$ with probability at least $1 - 2n^2e^{-2L_\delta} = 1 - \delta/2$, for some absolute constant $C_2 > 0$.

We now study the additional error introduced by A. By thinking of the rows of A as random vectors $g_k^\top/\sqrt{d_h} \sim \mathcal{N}(0, I_d/d_h)$, we can rewrite

$$\langle Au_x, Au_y\rangle - \langle u_x, u_y\rangle = \frac{1}{d_h}\sum_{k=1}^{d_h}\left((g_k^\top u_x)(g_k^\top u_y) - \langle u_x, u_y\rangle\right),\ \forall x, y \in \mathcal{V}.$$

We now fix the codebook U. In this way, the vectors u_x and u_y are deterministic, while the randomness comes only from the independent Gaussian vectors g_k. The terms in the preceding sum are independent, centered, and sub-exponential. Applying the Bernstein's inequality again gives

$$\mathop{\mathbb{P}}_{A}\left(\left|\langle Au_x, Au_y\rangle - \langle u_x, u_y\rangle\right| > C_3\,\|u_x\|\,\|u_y\|\left[\sqrt{\frac{2L_\delta}{d_h}} + \frac{2L_\delta}{d_h}\right]\,\middle|\, U\right) \le 2e^{-2L_\delta}.$$

For some absolute constant $C_3 > 0$. Now observe that $\|u_x\|\,\|u_y\| \le \max_z \|u_z\|^2 \le 1 + \eta_U$, where the last inequality follows by taking $x = y = z$ in the definition of η_U. Therefore, a union bound over all n^2 pairs gives

$$\eta_A \le C_4(1 + \eta_U)\left[\sqrt{\frac{L_\delta}{d_h}} + \frac{L_\delta}{d_h}\right],$$

with conditional probability at least $1 - \delta/2$, for some absolute constant $C_4 > 0$.

On the event where the preceding bound for η_U holds, the assumption $d \ge c_0 L_\delta$ gives $\eta_U \le C_2\left(\frac{1}{\sqrt{c_0}} + \frac{1}{c_0}\right)$. By choosing the universal constant c_0 sufficiently large, we can ensure that $\eta_U \le 1$. Consequently, $1 + \eta_U \le 2$, and hence $\eta_A \le 2C_4\left[\sqrt{\frac{L_\delta}{d_h}} + \frac{L_\delta}{d_h}\right]$.

The bound for η_U fails with probability at most $\delta/2$, while for every codebook U satisfying that bound, the conditional estimate for η_A fails with probability at most $\delta/2$. Therefore, both estimates hold simultaneously with probability at least $1 - \delta$. On this event,

$$\begin{aligned}\eta \le \eta_U + \eta_A &\le C_2\left[\sqrt{\frac{L_\delta}{d}} + \frac{L_\delta}{d}\right] + 2C_4\left[\sqrt{\frac{L_\delta}{d_h}} + \frac{L_\delta}{d_h}\right] \\ &\le C_0\left[\sqrt{\frac{L_\delta}{d}} + \frac{L_\delta}{d} + \sqrt{\frac{L_\delta}{d_h}} + \frac{L_\delta}{d_h}\right],\end{aligned}$$

where $C_0 = \max\{C_2, 2C_4\}$. Finally, for every query $q \in \mathcal{V}^h$ and every class $c \in \mathcal{V}$,

$$|G_c(q) - S_c(q)| \le \frac{1}{h} \sum_{j:y_j=c} \sum_{r=1}^{h} \left| \langle Au_{q_r}, Au_{(s_j)_r} \rangle - \mathbf{1}\{q_r = (s_j)_r\} \right|$$

$$\le \frac{1}{h} \sum_{j:y_j=c} \sum_{r=1}^{h} \eta = M_c \eta \le M_{\max} \eta. \qquad \square$$

A.2 Proof of Theorem 5

Proof. Fix an element $c \in \mathcal{V}$ and define

$$L_c(q)^2 = \sum_{y \neq c} G_y(q)^2 = \widehat{\mathcal{C}}_H(q) - G_c(q)^2, \qquad v_c(q) = \sum_{y \neq c} G_y(q) z_y.$$

Since the vectors z_y (with $y \neq c$) are independent and distributed as $\mathcal{N}(0, I_d/d)$, we have that $v_c(q) \sim \mathcal{N}\left(0, \frac{L_c(q)^2}{d} I_d\right)$, and $v_c(q)$ is independent of z_c. Conditional on z_c, the cross-talk term in equation (8) is therefore one-dimensional Gaussian distributed as

$$\langle z_c, v_c(q) \rangle \mid z_c \sim \mathcal{N}\left(0, \frac{L_c(q)^2}{d} \|z_c\|^2\right). \tag{11}$$

Because $\|z_c\|^2 \sim d^{-1}\chi^2_d$, standard chi-square concentration (Vershynin, 2018) gives

$$\mathbb{P}\left(\left|\|z_c\|^2 - 1\right| > \alpha\right) \le 2e^{-t}.$$

On the complementary event, $\|z_c\| \le \sqrt{1+\alpha}$. The Gaussian tail bound applied to equation (11) gives, conditionally on z_c,

$$\mathbb{P}\left(|\langle z_c, v_c(q) \rangle| > L_c(q) \|z_c\| \sqrt{\frac{2t}{d}} \;\middle|\; z_c \right) \le 2e^{-t}.$$

Combining the two events with equation (8), with probability at least $1 - 4e^{-t}$ we have

$$|\widetilde{S}_c(q) - G_c(q)| \le |G_c(q)|\alpha + L_c(q)\sqrt{1+\alpha}\sqrt{\frac{2t}{d}} \le \sqrt{\widehat{\mathcal{C}}_H(q)} \left(\alpha + \sqrt{\frac{2t(1+\alpha)}{d}}\right),$$

where last inequality holds since both $|G_c(q)|$ and $L_c(q)$ are at most $\sqrt{\widehat{\mathcal{C}}_H(q)}$. Finally, since $4e^{-t} = \delta/n$, a union bound over the n candidates proves simultaneous claim. $\square$

B Synthetic datasets construction

Dataset construction. We set $n = 1024$, $M = 4096 = 4n$, and $h \in \{4, 8, 16\}$. For each h, we construct a separate dataset with fixed query $q = (1, \ldots, h)$. The first 64 labels are active in $\mathcal{D}$ (i.e., appear has next-token) and each has 64 contexts, while the remaining 960 labels have ideal score zero but remain possible output candidates. For the distinguished label $c^* = c^*(q) = 1$, exactly $40h$ of the $64h$ token-position entries are selected uniformly without replacement and matched to the corresponding query token. For each other active label $c \neq c^*$, exactly $24h$ entries are matched. Every remaining entry at position r is drawn uniformly from $\mathcal{V} \setminus \{q_r\}$, preventing accidental matches. Hence, we have $S_{c^*}(q) = 40$, and $S_c(q) = 24$ for every other active label $c \neq c^*$, and the ideal margin is $\Delta(q) = 40 - 24 = 16$. Scaling the planted match counts with h keeps the ideal classification problem unchanged across the three context lengths.

Merged-label instance. To increase label repetition without changing the ideal margin, we construct a second labeling of the $h = 8$ dataset. The contexts remain unchanged, but the 64 active labels are mapped into 16 groups according to the map $c \mapsto 1 + \big((c-1) \bmod 16\big)$. Each merged label therefore contains 256 contexts. The group containing c^* has score $40 + 3 \cdot 24 = 112$, whereas every other merged label has score $4 \cdot 24 = 96$. Thus the margin $\Delta(q)$ remains 16, while $M_{\max}$ increases from 64 to 256. In this case the collision energy increases from $\mathcal{C}_H(q) = 40^2 + 63 \cdot 24^2 = 37,888$ to $\mathcal{C}_H^{\text{merged}}(q) = 112^2 + 15 \cdot 96^2 = 150,784$. The two label assignments consequently have the same ideal winner and margin, but different class loads and levels of output interference.

C Experimental Details

Synthetic Dataset Construction. We use the standard `WikiText-2` training, validation, and test splits (Merity et al., 2016). Whitespace tokenization with an explicit `<eos>` marker gives a training vocabulary of $n = 33,278$ input tokens. The task is restricted to the 64 most frequent training targets[2]. These targets cover 45.12% of eligible training positions and 44.79% of eligible test positions. This is therefore a controlled next-token experiment with 64-classes, not a full language-model benchmark.

We sample $100,000$ training, $2,000$ validation, and $5,000$ test positions with a fixed seed. Every sampled position has at least 16 preceding tokens on the same line. The same positions considered are consequently used for $h \in \{4, 8, 16\}$, and no context crosses a line boundary. All methods receive exactly the same examples. Finally, MEMO uses the untied square construction $d_h = d/h$, dimensions $d \in \{128, 256, 512, 1024\}$, and five independent codebook seeds.

Compared methods. The *sum prototype* is the exact classifier in equation (3), implemented with token-position-label count tables rather than by linear scanning the training set. The *mean prototype* divides each score by its class size and therefore removes the empirical class frequency factor. We also use a count-based suffix model that backs off from at most five preceding tokens, a linear SVM trained with stochastic hinge loss on the positional one-hot features Ψ, and exact Hamming k-nearest neighbors. For the latter, $k \in \{1, 5, 25\}$ is chosen on validation data ($k = 25$ is selected for all three context lengths). Ties are resolved deterministically. We report top-1 and top-5 accuracy, macro-F1, and *agreement*[3] with the sum-prototype winner. Since the ideal winner is unique on at least 99.34% of the test queries, agreement is reported on that unambiguous subset. We emphasize that the agreement with the *exact sum prototype is our primary measure* of how faithfully MEMO implements the positional Hamming-kernel classifier, independently of that classifier's accuracy against the observed next token.

D Additional WikiText-2 results

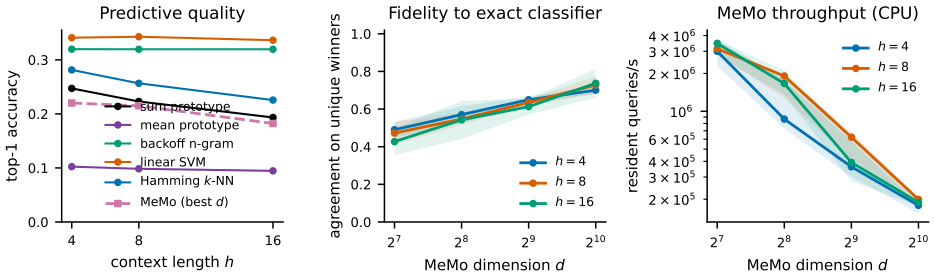

Figure 2: Real-data results on `WikiText-2` dataset. **Left:** top-1 accuracy of the five baselines and the best observed MEMO dimension for each h. **Center:** MEMO agreement with the exact sum prototype on queries with a unique ideal winner. The curves show means over five codebook seeds, and bands show the standard deviation. **Right:** number of queries per second by MEMO on an Intel Core i7-13700H CPU, for different values of d and h. Queries are processed in batches of 512 and loaded into memory before timing.

[2] After excluding `<unk>`.

[3] That is, the percentage of queries for which a method predicts the same token as the exact Hamming classifier.