Memory Modules for Language Models Trade Fact Recall Against Question Sensitivity

Vuk Rosić

https://github.com/vukrosic/llm-memory-env

Preprint, November 2026

Abstract

Memory modules add a learned lookup table to a language model, but published comparisons report mainly validation loss, which leaves open what each module stores and when it applies what it stored. We compare nine published memory modules and two composite designs under one protocol: identical web text with inserted ParaRel facts, a fixed weight budget, and six scores covering text prediction, fact recall in seen and unseen wordings, and sensitivity to the question being asked. At 43M weights, a composite of a word table, MoME, a zero-initialized n-gram table and mixture-of-experts layers reaches the lowest web loss of all modules tested (4.550 against 4.591 for the best published module) and matches the best published modules on fact recall. At this size, however, every table-based memory lowers the relation gap below that of a model without memory: the tables raise the stored answer even under questions that ask for a different relation. Test-time ablations place this leak in the word table and the attention-side table, while the n-gram table preserves question sensitivity. Module rankings also change with scale (rank correlation 0.30 on the relation gap between our two largest sizes), so single-scale comparisons of memory modules should be read with care.

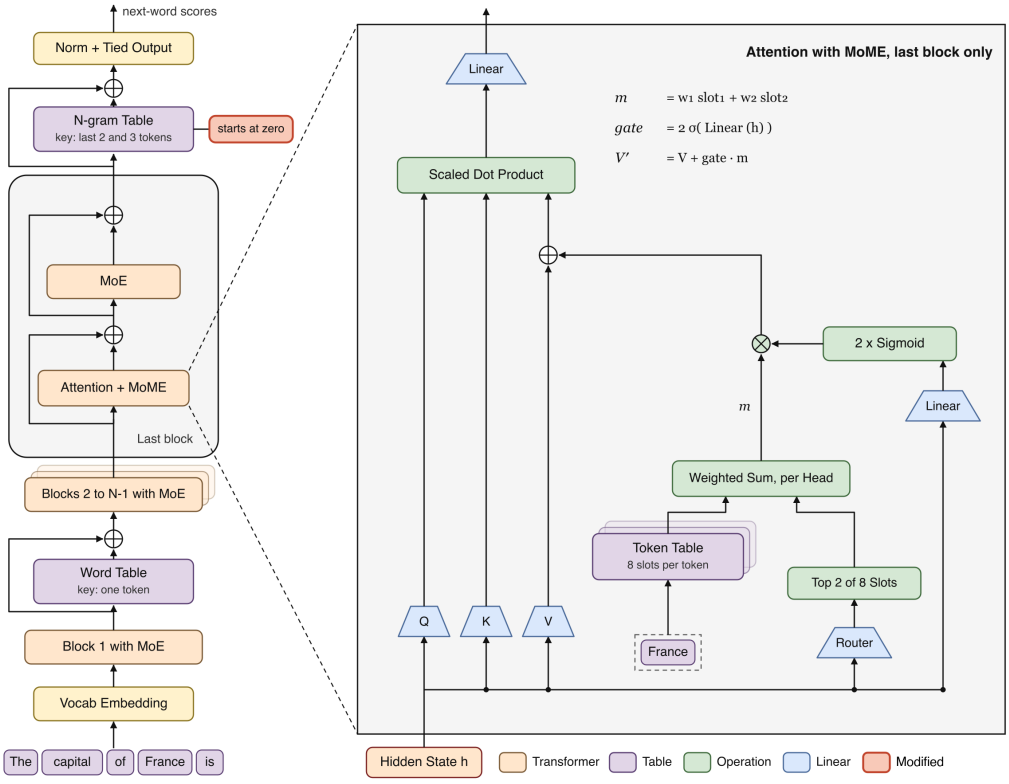


Figure 1: CompMem. Left: the full model, with a word table after block 1, MoE in every block and a zero-initialized n-gram table after the last block. Right: attention with MoME in the last block, where each head mixes 2 of a token's 8 slots into its value. FitGate replaces the MoME table with a gated (token, code) table added after attention.

1 Introduction

A transformer stores what it learns in its weights and reaches that knowledge through the hidden state. Memory modules add a second path: a large learned table whose rows are selected by the current token, the last few tokens, or a router, and whose output is added to the computation. Recent designs such as Engram [1], MoME [2], STEM [3] and value embeddings [7] report lower validation loss at equal weights, and Engram is presented as a complement to mixture-of-experts (MoE) layers [1].

Validation loss does not say what a table holds or when it applies it. A table keyed on the token "France" that has learned to raise "Paris" after *The capital of France is* may raise "Paris" after *The currency of France is* as well. A useful memory should recall a stored fact when it is asked, including in new wording, and hold that fact back when the question asks for something else. We measure both properties for every module under the same training data, weight budget and evaluation.

Contributions.

- A shared evaluation of nine published memory modules with six scores: web-text loss, recall of stored facts, recall in unseen wordings, recall with a distant subject, a *relation gap* that measures question sensitivity, and loss on the most common continuation of ambiguous phrases (Section 3).
- Two composite designs within Engram's weight budget. CompMem has the lowest web loss of all 21 configurations evaluated at 43M weights (nine published modules, three published-module + MoE controls and nine design variants) and ties the best published modules on all three recall scores; FitGate has the best stored-fact and distant-subject recall (Sections 4 and 5).
- Evidence for a recall–sensitivity trade-off: no published or new module improves both unseen-wording recall and the relation gap over the best baseline, and test-time ablations identify which table causes the leak (Section 5.2).
- Evidence that module rankings are not stable across model sizes, which limits what any single-scale comparison can establish (Section 5.3).

2 Related Work

Lookup-table memories. Engram [1] modernizes n-gram embeddings: it hashes the last few tokens into large tables, adds the gated row to the hidden state, and treats this conditional memory as a sparsity axis complementary to MoE. TN-gram [4] replaces Engram's separate per-order hash tables with shared tensor factors. Lngram [5] performs n-gram lookup over discrete symbols learned from hidden states instead of token IDs. STEM [3] replaces the feed-forward up-projection with a token-indexed embedding lookup. Value embeddings [7] add a per-token vector to the attention values. MoME [2] replaces each token's single memory row with a mixture of slots chosen by a learned gate over the hidden state, and compares against value embeddings, a bigram table and STEM. Memory Attention [6] forms attention values from token-indexed memory combined with contextual keys, replacing the value projection.

Measuring factual knowledge. ParaRel [9] provides facts with many paraphrased templates per relation, which we use both for training-time facts and for unseen wordings. CounterFact [10] scores paraphrase generalization and neighbourhood specificity for model editing; our new-wording and relation-gap scores are closest to these, applied to facts learned in pretraining rather than edited in afterwards.

3 Evaluation Protocol

Data. Models train on fresh web text, with each token read once, mixed with 10% fact sentences built from ParaRel. Each fact appears 2, 8 or 32 times during training in a fixed set of wordings; other wordings of the same relation are held out for evaluation.

Model sizes. All models are pre-norm decoder-only transformers with tied input and output embeddings. We report three sizes (Table 1). Every memory module must fit within the total weight count of Engram at that size, which is 43.54M at size L.

Scores. Lower is better for every score except the relation gap.

- *Web loss*: next-token loss on held-out web text.
- *Stored facts*: answer loss for facts seen 32 times, asked in a training wording.

Table 1: Model sizes. Weight budget is Engram's total weight count, which caps every module.

Size	Width	Layers	Heads	Vocabulary	Training tokens	Weight budget
S	64	4	4	8,000	0.9M	1.77M
M	96	4	4	12,000	3.1M	4.76M
L	256	6	8	20,000	41.0M	43.54M

- *New wording*: answer loss for the same facts, asked in a wording never seen in training.
- *Distant subject*: new wording in which the subject is far from the answer position.
- *Relation gap*: log-probability of the stored answer under its own question minus its log-probability under a question for a different relation about the same subject. A high gap means the model withholds the answer when the question changes.
- *Usual phrase*: loss on the most frequent continuation of phrases that have several continuations.

Statistics and tuning. Every module is evaluated on the same items, so we compare paired per-item differences and report z, the gap to the best published module in standard errors; positive z means better. Published modules are reimplemented from the descriptions in their papers and trained at learning rates 0.01 and 0.02, and the rate with the lower loss on separate tuning items is reported. A success criterion was fixed before any new design was trained: at sizes S and M, lowest web loss by $z \geq 2$, no other score lost by $z \leq -2$, and no more weights than Engram.

4 Composite Designs

Both designs add four components to the transformer, all within Engram's budget (Figure 1):

1. **MoE in every block**: one shared expert plus the top 2 of 8 routed experts. Its weights are taken from the table budget.
2. **A word table after block 1**, with one row per token, read through Engram's gated path.
3. **A table inside attention in the last block** (the two designs differ here).
4. **An n-gram table after the last block**: Engram's table keyed on the last two and three tokens, with its output projection initialized to zero.

CompMem. The attention-side table is MoME: each token owns 8 slots of width $d/8$; for each head a router reads the hidden state, picks 2 slots, mixes them and adds the result to that token's value vector through a gate $2\sigma(\text{Linear}(h))$. Apart from the zero initialization of the n-gram output, every part follows a published module; the contribution is the combination and its placement within one budget.

FitGate. The attention-side table is keyed by both a token and a code chosen by the reading head. Each head takes the 4 tokens it attended to most and uses its own state to select 2 of 16 codes; the row for each (token, code) pair is gated by the cosine between the head output and the row, and added after attention rather than to the values. The intent is that a row is applied only when it agrees with what the model already predicts.

5 Results

5.1 Comparison at 43M weights

Table 2 lists every published module and both designs at size L. No published module leads on more than two scores. MoE alone has the best relation gap, STEM the best unseen-wording recall, and Engram the best usual-phrase loss among published modules.

CompMem has the lowest web loss of all modules ($z = +14.3$ against MoME) and the lowest usual-phrase loss ($z = +4.8$), and ties the best published module on stored facts, new wording and distant subject ($|z| < 2$). FitGate has the best stored-fact recall (0.062, $z = +2.2$) and distant-subject recall (1.605, $z = +2.5$). Both lose the relation gap by a wide margin ($z = -14.4$ and -15.4 against MoE). The seven other design variants evaluated at this size, all within budget, reach web losses of 4.553–4.581 and relation gaps of 3.830–4.758; their full scores are in the released results.

Where the gain comes from. Adding MoE to a published module is a strong control, because MoE alone lowers web loss from 4.656 to 4.620. Table 3 separates the components. Most of the web-loss improvement over MoME comes

Table 2: All modules at size L (43.54M weight budget). Bold marks the best value in each column. ↑: higher is better; all other scores are losses. STEM (50.23M) and the token table (44.61M) exceed the budget and are shown for reference.

Module	Weights	Web	Stored	New wording	Distant subj.	Rel. gap ↑	Usual
No memory	9.84M	4.656	0.442	2.375	3.244	5.406	1.577
Token table	44.61M	4.641	0.321	2.129	2.734	5.122	1.539
Engram [1]	43.54M	4.604	0.127	3.299	3.839	4.757	1.456
MoE [8]	14.55M	4.620	0.283	1.817	2.555	**6.041**	1.548
Value embeddings [7]	25.20M	4.601	0.120	1.689	2.228	4.721	1.506
MoME [2]	25.25M	4.591	0.105	2.144	2.275	4.152	1.504
STEM [3]	50.23M	4.615	0.097	1.417	1.825	4.467	1.503
Bigram table [2]	25.21M	4.611	0.084	2.480	3.079	4.361	1.478
TN-gram [4]	44.03M	4.619	0.170	3.599	3.882	5.249	1.509
Engram + MoE	43.20M	4.584	0.074	3.353	3.710	4.976	1.433
MoME + MoE	29.96M	4.562	0.094	2.038	2.369	4.254	1.482
Value emb. + MoE	29.91M	4.581	0.111	1.709	2.106	4.902	1.480
CompMem (ours)	43.30M	**4.550**	0.091	**1.370**	1.706	4.383	**1.410**
FitGate (ours)	43.38M	4.558	**0.062**	1.484	**1.605**	4.245	1.422

from adding MoE, which uses only published parts; the word and n-gram tables add a further 0.012 ($z = 4.7$). Their larger effect is on recall: unseen-wording loss is 1.370 with them and 2.038 without. Adding MoE to the three tables lowers web loss by 0.020 at a slightly lower weight count, because MoE weights are taken from the tables.

Table 3: Component decomposition at size L. Weight counts differ, so this shows contributions rather than value per weight.

Model	Weights	Web loss	New wording
No memory	9.84M	4.656	2.375
MoE	14.55M	4.620	1.817
MoME	25.25M	4.591	2.144
MoME + MoE	29.96M	4.562	2.038
Word table + MoME + n-gram table	43.55M	4.570	1.658
CompMem (all three tables + MoE)	43.30M	**4.550**	**1.370**

Medium size. At size M, FitGate's recall lead is larger: stored facts 0.098 against 0.224 for value embeddings ($z = +5.0$), new wording 0.724 against 0.962 ($z = +2.3$), distant subject 1.068 against 1.784 ($z = +8.3$), with web loss 5.365 against 5.386 for MoME ($z = +4.7$). Its relation gap is 4.082 against Engram's 5.368 ($z = -8.4$). The published-only control MoME + MoE already beats every published module on unseen wording at this size, so part of FitGate's recall gain is attributable to known parts; FitGate still beats that control directly on web loss ($z = +2.8$), stored facts (+2.9), distant subject (+4.7) and relation gap (+5.4).

Success criterion. No module, published or new, meets the criterion fixed in advance. Each published module fails it against the other eight at every size, because they lead on different scores. We therefore read the criterion as a target and report all scores.

5.2 Recall and question sensitivity trade off

Figure 2 plots unseen-wording recall against the relation gap for all runs at size M. Modules that recall facts in new wording best (MoME, value embeddings, FitGate) have the lowest gaps, while the modules with the highest gaps (Engram, the bigram table, and a design keyed by token and a bucket of another token) recall worst. No run improves on the best baseline in both directions. At size L, every new design keeps the relation gap as its largest loss, and even the model with no memory has a higher gap (5.406) than CompMem (4.383). The same holds for all seven table-based published modules (4.152–5.249); MoE, which adds no table, is the only module above the model without memory. At size M the pattern is weaker: Engram, the bigram table and TN-gram have higher gaps than the model without memory, but also worse unseen-wording recall.

To locate the leak, we switched off each table of a trained FitGate model at test time and measured the stored answer's

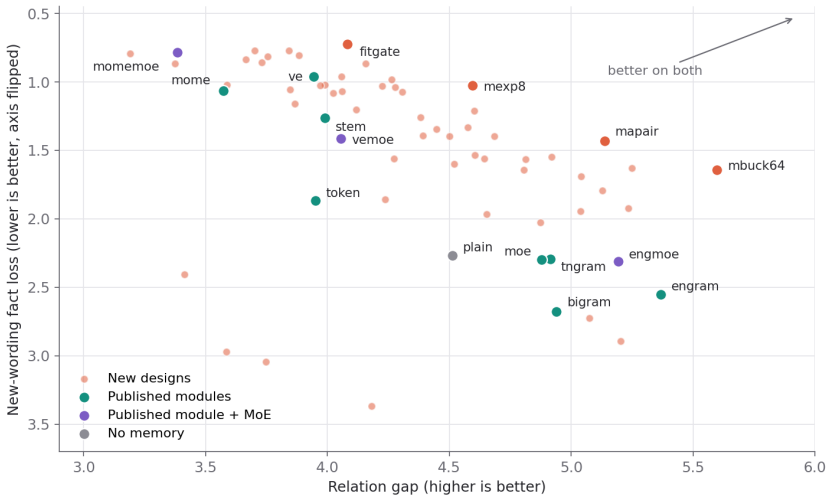


Figure 2: Unseen-wording loss (lower is better, axis flipped) against relation gap (higher is better) for all runs at size M. The upper-right region, better than every baseline on both scores, is empty. Labels are code names; `fitgate` is FitGate, `momemoe`, `vemoe` and `engmoe` are published modules with MoE.

log-probability under its own question and under another relation's question (Table 4).

Table 4: Test-time ablation of FitGate at size M. Log-probability of the stored answer; gap = own − other.

Table switched off	Own question	Other relation	Gap
None (all tables on)	−1.400	−5.482	4.082
N-gram table (after last block)	−2.009	−4.569	2.560
Word table (after block 1)	−4.605	−7.860	3.255
Gated table (inside attention)	−3.717	−8.898	5.181
All three	−8.827	−9.220	0.393

The three tables play distinct roles:

- The **word table** contributes most recall: removing it lowers the correct answer by 3.21 nats.
- The **gated attention table** adds recall but also leaks: it raises the correct answer by 2.32 and the same answer under the wrong relation by 3.42. Its cosine gate, close in form to Engram's gate, does not close when the relation changes.
- The **n-gram table** protects question sensitivity: without it the wrong-relation probability rises and the gap falls from 4.08 to 2.56.

A closely related design (mtrip, the same layout without the cosine gate) shows the same pattern. Because a correct answer's log-probability is bounded by zero while the wrong-relation probability can keep rising, a boost that ignores the question shrinks the gap as recall improves. A memory that keys only on the subject cannot distinguish the relation being asked.

5.3 Rankings change with scale

A token table with one row per token and Engram's weight count beats Engram on web loss at sizes S ($z = -7.9$ in its favour) and M ($z = -2.9$), and loses at size L ($z = +11.9$ for Engram). Among the nine published modules, the Spearman correlation of rankings between sizes M and L is 0.82 for distant subject, 0.80 for usual phrase, 0.68 for web loss, 0.63 for new wording, 0.52 for stored facts and 0.30 for the relation gap. Engram ranks first on the relation gap at M and fifth at L; STEM leads both unseen-wording scores at L and neither at M.

New designs show the same drift. Against the best published module, mtrip's web-loss lead grows from 0.015 at M to 0.026 at L, while its distant-subject lead shrinks from 0.496 to 0.035 and its stored-fact lead from 0.130 to 0.002. Text-prediction gains from these designs therefore persist with scale more reliably than recall gains.

5.4 Compute

The weight budget does not cap compute. Measured time per training step at size M, relative to Engram: MoME 1.06×, MoME + MoE 1.21×, CompMem 1.31×, FitGate 1.40×. A comparison at equal training time has not yet been run.

6 Limitations

- **Scale.** The largest models have about 43M weights and train on about 41M tokens. This is about one training token per weight, well below compute-optimal ratios, so all models are undertrained relative to standard practice. Given the ranking changes in Section 5.3, results at this scale do not establish behaviour at billions of parameters. Standard benchmarks (HellaSwag, ARC, MMLU, WinoGrande) are near chance at these sizes; BLiMP is the only one that moves clearly.
- **Selection on evaluation items.** CompMem and FitGate were chosen from about 90 design variants scored on the same evaluation items reported here, so their z values are biased upward. A held-out item set exists; CompMem has not been scored on it, and FitGate's held-out scores have not yet been analysed.
- **Ablation scope.** The test-time ablation (Table 4) was run on FitGate at size M only; it has not been run on CompMem or at size L.
- **Variance.** Each setting has one training run. Our z values capture variation over test items, not over training runs, and some will shrink with repeated training.
- **Inserted facts.** Facts not inserted in training first separate modules only at size L, so recall scores may partly reflect memorization of the inserted sentences rather than general knowledge.

7 Conclusion

Under one protocol, nine published memory modules lead on different scores, and none dominates. Combining a word table, MoME, a zero-initialized n-gram table and MoE within Engram's weight budget gives the lowest web loss at 43M weights while matching the best published recall. The same combination, like every table-based module at this size, lowers question sensitivity below that of a model without memory: the tables raise a stored answer even when the question asks for a different relation, and only the n-gram table counteracts this. Closing this gap, through keys or gates that depend on the relation being asked, and testing whether these rankings hold at larger scale are the main open problems. Code, all runs and per-item scores are released at https://github.com/vukrosic/llm-memory-env.

References

[1] X. Cheng, R. Tian, W. Zeng, D. Dai, et al. Conditional memory via scalable lookup: a new axis of sparsity for large language models. arXiv:2601.07372, 2026.

[2] M. Li, L. Sigal, R. Liao. MoME: mixture-of-memory embeddings for context-aware sparse lookup. arXiv:2609.15126, 2026.

[3] R. Sadhukhan, S. Cao, H. Dong, C. Zhao, et al. STEM: scaling transformers with embedding modules. arXiv:2601.10639, 2026.

[4] W. Zhou, Y. Gu, G. Iacovides, Y. Qiu, et al. Tensorizing Engram: sharing latents across n-gram embeddings is beneficial in LLMs. arXiv:2606.08347, 2026.

[5] Y. Zheng, G. Xia, X. Wang, L. Ren. Lngram: n-gram conditional memory in latent space. arXiv:2605.24869, 2026.

[6] J. Kang. Memory attention. arXiv:2609.28399, 2026.

[7] K. Jordan et al. modded-nanogpt: speedrunning GPT-2 training. GitHub repository, 2024.

[8] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, J. Dean. Outrageously large neural networks: the sparsely-gated mixture-of-experts layer. ICLR, 2017.

[9] Y. Elazar, N. Kassner, S. Ravfogel, A. Ravichander, E. Hovy, H. Schütze, Y. Goldberg. Measuring and improving consistency in pretrained language models. TACL, 2021.

[10] K. Meng, D. Bau, A. Andonian, Y. Belinkov. Locating and editing factual associations in GPT. NeurIPS, 2022.