MemLLM: Finetuning LLMs to Use Explicit Read-Write Memory

Anonymous ACL submission

Abstract

While current large language models (LLMs) perform well on many knowledge-related tasks, they are limited by relying on their parameters as an implicit storage mechanism. As a result, they struggle with memorizing rare events and with updating their memory as facts change over time. In addition, the uninterpretable nature of parametric memory makes it challenging to prevent hallucination. Model editing and augmenting LLMs with parameters specialized for memory are only partial solutions. In this paper, we introduce MemLLM, a novel method of enhancing LLMs by integrating a structured and explicit read-and-write memory module. MemLLM tackles the aforementioned challenges by enabling dynamic interaction with the memory and improving the LLM's capabilities in using stored knowledge. Our experiments indicate that MemLLM enhances the LLM's performance and interpretability, in language modeling in general and knowledge-intensive tasks in particular. We see MemLLM as an important step towards making LLMs more grounded and factual through memory augmentation.

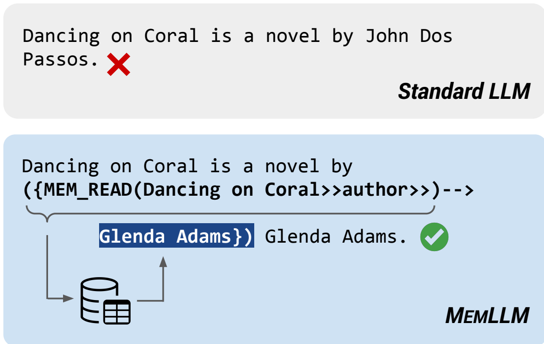


Figure 1: A vanilla pretrained LLM can produce hallucinated nonfactual output. In contrast, MemLLM dynamically queries its explicit memory for stored facts, resulting in more factually grounded text generation.

1 Introduction

State-of-the-art large language models (LLMs) perform well in knowledge-intensive tasks (Yu et al., 2023; Chowdhery et al., 2023). They solve these tasks utilizing the information memorized in their vast array of parameters (Roberts et al., 2020). However, the effectiveness of parameter-based memorization is limited for infrequent entities and concepts (Kandpal et al., 2023; Mallen et al., 2023) and is prone to temporal degradation (Kasai et al., 2023; Jang et al., 2022). Parametric model editing may address some of these issues (Sinitsin et al., 2020; De Cao et al., 2021; Mitchell et al., 2022), but struggles with maintaining locality – possibly damaging model performance in unrelated areas (Yao et al., 2023b; Gu et al., 2024). Moreover, model editing often deteriorates performance when applied to sequential editing or batch updates. This is because it primarily focuses on applying (and evaluating) single edits one-by-one (Huang et al., 2023). Finally, model editing may struggle to generalize and maintain previous edits when updating multiple facts simultaneously (Yao et al., 2023b).

Other parametric solutions, like augmenting LLMs with extra parameters such as memory pools can preserve knowledge for subsequent utilization (Wang et al., 2023a, 2024b). However, parametric memorization is prone to distortion and hallucinated nonfactual output. In addition, parametric mechanisms like memory pools have limited capacity and lack interpretability (Maynez et al., 2020; Ji et al., 2023).

Another approach is to augment LLMs with a non-parametric memory component that interacts with the LLM either through natural language text or a formalized API (Wang et al., 2023b). Although prior work has demonstrated enhanced abilities in extended dialogs, long-text generation and question answering (Packer et al., 2023; Hu et al., 2023; Zhou et al., 2023), these methods are primarily prompt-dependent and necessitate customization for each specific task and model. They also suffer from the lack of a structured memory. This undermines interpretability and interoperability (Wang et al., 2023b).

In this paper, we introduce MEMLLM, an LLM endowed with an explicit memory component. It has the general advantages of some of the memory-focused work we discussed: we can **keep information accessible indefinitely**, beyond the context window, including **infrequent information** that standard LLMs struggle with.

The LLM has **both read and write access** to the memory component, i.e., it can store information in the memory as it processes text (or interacts with a user) and retrieve it when it needs it (Figure 1).

We specify an **API for read and write access**. The LLM issues read commands to retrieve from the memory and write commands to write to the memory. We adopt **finetuning** to train the model to access the memory. Based on the API specification, we create a dataset with training examples of API read and write commands and finetune the LLM on it. Our published training dataset can be used to finetune any language model, endowing it with explicit memory without requiring architectural changes.

Our memory has an explicit structured schema, similar to a database schema. It is **interpretable** and inspectable for humans. It is **editable** by humans. It is **scalable** since databases have excellent scalability properties. It is **interoperable** since the contents of the memory can be exported (e.g., to a different LLM supporting explicit memory) and contents can be imported from data resources (e.g., from Wikidata).

Our evaluation on Re-DocRED (Tan et al., 2022) demonstrates that MEMLLM achieves better perplexity compared to baselines without memory components, with strong gains on named entities. We also show that MEMLLM outperforms non-memory-based methods on knowledge editing.

2 Related work

External memory augmentation. Augmenting an LLM with memory as an external component can enhance its ability to process larger contexts and maintain reliable records by storing facts and knowledge. Such components include databases, knowledge bases and knowledge graphs that LLMs interact with via natural or formal language (Guu et al., 2020; Lewis et al., 2020; Liu et al., 2022; Yao et al., 2023a; Park et al., 2023; Zhou et al., 2023; Schick et al., 2023; Hu et al., 2023). For instance, Retrieval Augmented Generation (RAG) retrieves relevant text snippets from large document databases, to improve factuality (Guu et al., 2020; Lewis et al., 2020). Other recent solutions store summarized information from previous contexts for future retrieval, improving performance in long-form generation, summarization, question answering and dialog coherence (Park et al., 2023; Zhou et al., 2023; Packer et al., 2023; Chen et al., 2023; Liang et al., 2023).

Our framework aligns with external memory methodologies but stands out with its structured format for storing information. This explicit memory facilitates large-scale knowledge editing and makes the model's output generation process more interpretable. While similar structured storage approaches exist, they are often task-specific, such as data record management (Hu et al., 2023). Unlike these, our method is designed for generic language modeling, making it adaptable to a variety of tasks without extensive prompt engineering. Our published training dataset can be used to endow any trainable language model with explicit memory without requiring architectural changes.

Memory as a state. The term *memory* can refer to recurrent architectures that represent past context with vectors (Hochreiter and Schmidhuber, 1997; Cho et al., 2014). Transformer-based models use similar mechanisms with memory tokens (to transfer context across segments) and memory pools (to share information across multiple contexts) (Burtsev et al., 2020; Bulatov et al., 2022; Wang et al., 2024b). While recent advances use vector- or parameter-based memory systems for long-range dependencies (Martins et al., 2022; Wu et al., 2022a,b; Cheng et al., 2023; Wang et al., 2023a; He et al., 2024), they are limited by memory vector capacity (Jelassi et al., 2024). In contrast, MEMLLM has no such architectural limitations and features explicit, interpretable and editable memory.

Knowledge editing. The goal of knowledge editing is to apply data-efficient changes to a model's behavior for a set of edits while keeping other knowledge unaffected (Yao et al., 2023b; Gu et al., 2024; Zhang et al., 2024). Meta-learning and locate-then-edit are two classes of parametric methods that modify model weights. In meta-learning, a hypernetwork is trained and applied to the model weights during test time (De Cao et al., 2021;

Mitchell et al., 2021). In locate-then-edit, the weights triggered by a knowledge expression are located and modified (Dai et al., 2022; Meng et al., 2022). There are also memory-based methods that do not alter the original model weights but use an external memory (Gu et al., 2024). E.g., methods like SERAC, GRACE and DEFER use retrieval-based memory to fetch previously given edits and apply them to new inputs (Mitchell et al., 2022; Hartvigsen et al., 2024). In WISE (Wang et al., 2024a), in addition to the LLM, two additional parametric models are trained: a side memory and a routing network. Based on the query, the routing network decides which memory to use, the side memory or the main LLM. Evaluations show that multiple edits at a time (batch editing) or successive edits (sequential editing) are challenging tasks – but certainly critical for the intended application of knowledge editing. While most methods can handle a few edits at a time, their performance drops when applying more (Yao et al., 2023b; Wang et al., 2024a). Due to the explicit memory structure of MEMLLM, it can handle a large number of edits while maintaining performance.

3 Methodology

Our approach to endowing an LLM with an explicit memory is finetuning with the standard language modeling objective. We now present a finetuning regime that teaches the LLM (1) to extract knowledge from text and write it to the memory and (2) to read knowledge from the memory and leverage it for better language modeling. Following Schick et al. (2023) and Modarressi et al. (2023), we define an API through which the LLM initiates memory writes and memory reads.

3.1 Memory structure

The memory stores information in relation triples. Each triple has the form $r = \langle e_s, t, e_o \rangle$, where e_s is the first entity or subject, e_o the second entity or object and t the relation. Example: ⟨Washington D.C., capital of, United States⟩. The entities and relations are stored as raw text and vectors, each in separate tables. The vector representations (created with Contriever (Izacard et al., 2022)) abstract away from different surface forms of the same entity, e.g., "US" vs "USA". In the interest of brevity, we use the symbols e and t for both the entity/relation itself and for its vector.

Our **query format** for querying the memory is:

$$\mathbf{q} \in \{\langle e_s^q, t^q, * \rangle, \langle *, t^q, e_o^q \rangle\}$$

where e_s^q, e_o^q, t^q are subject entity, object entity, and relation. $*$ indicates the position in the triple of the entity we are querying for. These two templates give us sufficiently specific queries (as opposed to, e.g., queries with two variables) that are likely to return useful entity information.

We now want to retrieve triples from the memory that match the query. Given that the surface form of an entity (and also the relation) can vary (e.g., "US" vs "USA"), our match criterion is not exact match, but rather vector similarity. We refer to entities/relations that are similar to the query entity and the query relation as *candidate* entities/relations.

For retrieval, we first determine a set of candidate entities:

$$\mathcal{C} = \{e' | \cos(e^q, e') \geq \tau_e\}$$

That is, all entities with an above-threshold similarity are considered candidate entities.

Similarly, we determine a set of candidate relations:

$$\mathcal{T} = \{t' | \cos(t^q, t') \geq \tau_t\}$$

If the query is a query for an object, i.e., $\mathbf{q} = \langle e_s^q, t^q, * \rangle$, then we retrieve the following final set $\mathcal{E}$ of entities from the memory:

$$\{e_o | \exists (e, t, e_o) \in \mathcal{M} : e \in \mathcal{C}, t \in \mathcal{T}, \\ 0.5(\cos(e, e_s^q) + \cos(t, t^q)) \geq \tau_r\}$$

where $\mathcal{M}$ is the memory. That is, we look for all triples in the memory with entities/relations from the candidate sets such that their average similarity to query subject and relation is above the threshold. Subject queries are handled analogously.[1]

3.2 Memory-API and Inference

We now describe the API that specifies how the LLM initiates memory writes and memory reads.

Memory writes. We process the input sentences one by one. For sentence s_i the input x_i^{MW} to the LLM is formatted as follows:

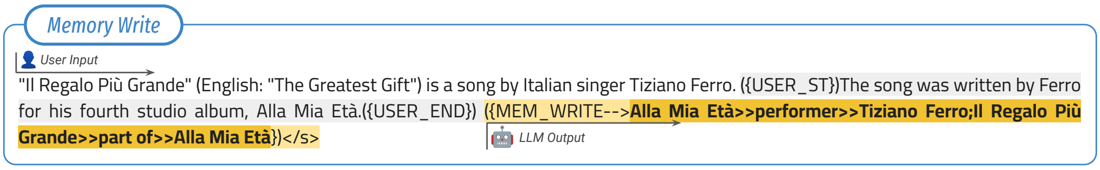
$$x_i^{\text{MW}} = S_{<i} + (\texttt{\{USER\_ST\}}) + s_i + (\texttt{\{USER\_END\}})$$

where $S_{<i}$ are the $i-1$ preceding sentences and s_i is bracketed by tags to mark it as the *focus sentence*. The LLM's task is then to extract all relations occurring in the focus sentence and to generate a write command that stores them in the memory:

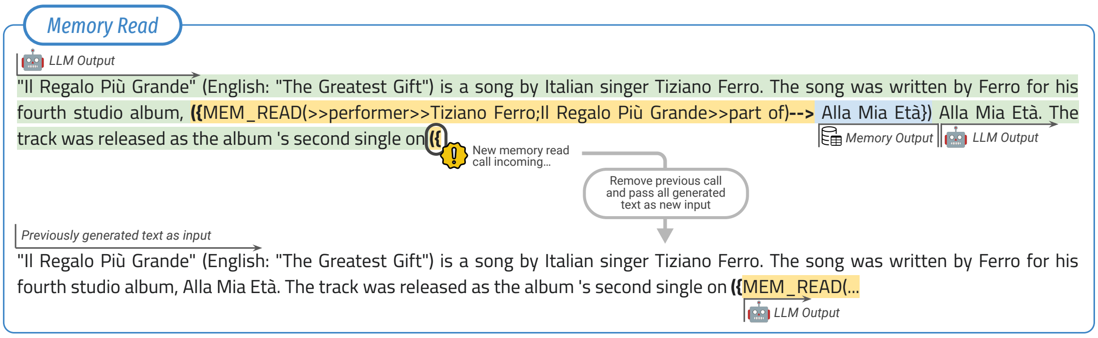
$$y[x_i^{\text{MW}}] = (\texttt{\{MEM\_WRITE-->}e_s^1 \text{»} t^1 \text{»} e_o^1; e_s^2 \text{»} t^2 \text{»} e_o^2; \ldots \texttt{\}})$$

Context $S_{<i}$ is necessary to extract relations from the focus sentence, e.g., if the focus sentence refers

[1] We discuss how we set the thresholds in Appendix D



(a) For memory writes, the input is given in two parts. (i) The pretext provides context for the model (e.g., antecedents for pronouns). (ii) The focus sentence is the span of text (bracketed by `({USER_ST})` and `({USER_END})`) from which the model is tasked to extract all relations. The model calls the API starting with the `({MEM_WRITE-->` command followed by the extracted relations. `})</s>` closes the API call. In each document, MemLLM scans the sentences one by one.



(b) The model decodes one token at a time, as in standard language modeling. It is also trained to generate memory read commands at points when they can retrieve useful information. In the example, after decoding some tokens, the model generates a `({MEM_READ(` command followed by queries. `-->` triggers execution of the queries. Returned results are appended. The model then uses the retrieved results for decoding the posttext. Whenever, during further decoding, the model initiates a new memory read by emitting `({`, we remove the previous one because it is unlikely to still be useful.

Figure 2: MemLLM inference with memory read and memory write: Examples

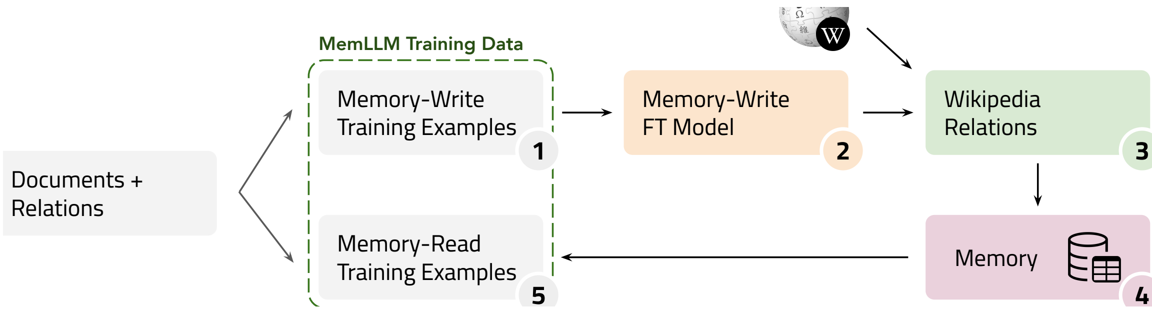


Figure 3: MemLLM training data pipeline.

to a previously introduced entity with a pronoun. We finetune the LLM to only extract relations from the focus sentence (not from the preceding context); see §3.3 for details.

To extract all relations from a document and write them to the memory, we iterate over the sentences of a document one by one.

Memory reads. The LLM can at each point in time either emit a regular token or initiate an API MEM_READ call by generating:

`({MEM_READ(`

It then continues by generating subject or object queries as introduced above: $\mathbf{q} \in \{\langle e_s^q, t^q, *\rangle, \langle *, t^q, e_o^q\rangle\}$. The syntax for the memory-read API call is:

`({MEM_READ(`e_s^{q1}»t^{q1}»;»t^{q2}»$e_o^{q2}$$;\ldots$`)-->`

The entity sets $\mathcal{E}$ are then retrieved from the memory (§3.1), merged and appended to the API call:

`({MEM_READ(`e_s^{q1}»t^{q1}»$;\ldots$`)-->`$e_1, e_2, e_3, \ldots$`})`

The LLM then continues decoding. Figure 2b gives an example. The LLM starts generating a sentence that refers to an album by the Italian singer Tiziano Ferro. It has learned that just before naming the album is a good point at which to initiate a memory read. Two queries are generated (including: "What is the song "Il Regalo Più Grande" part of?"). One entity is returned by the memory ("$--\ >$ Alla Mia Età})") and written to the buffer. The LLM then generates the name of the correct album (" Alla Mia

Età."). This example illustrates that our memory has the potential of reducing hallucinations because through the memory an explicit representation is available of the fact that "Il Regalo Più Grande" is part of the album "Alla Mia Età".

We remove memory-read API calls from the context if they are no longer useful. This happens in three cases: (i) The returned set $\mathcal{E}$ of entities is empty. (ii) The number of retrieved entities exceeds a threshold Q_{thr} ($Q_{thr} = 30$). Such large retrieval results are unlikely to be helpful. (iii) The model emits "({", initiating a new memory-read API call.

Our motivation for removing the API call is as follows. Omitting API verbiage preserves the text's natural flow and reduces the context to those parts of the input that are still informative for high-quality generation.

3.3 Finetuning the LLM

We now describe how we create the dataset for finetuning the model to generate memory-write and memory-read API calls. One innovation of our work is that we create these API training data from corpora annotated with entities and relations, including Re-DocRED (Tan et al., 2022), which we will use as an example below.

Memory-write data. Figure 3 shows how we use Re-DocRED's annotated relations. For each sentence s_i, we retrieve from Re-DocRED all relation triples such that one entity has a full mention (i.e., not a pronoun) in s_i and the other entity has a full mention either in s_i or in the pretext ($S_{<i}$). The memory-write training example consists of the context x_i^{MW} and the memory-write command $y[x_i^{\text{MW}}]$; see §3.2 and Figure 2a. Since we want to teach the LLM to generate memory-write API calls, we compute the training loss on $y[x_i^{\text{MW}}]$ only.

The set of relation triples can be empty for a sentence s_i. In that case we generate a memory-write command in $y[x_i^{\text{MW}}]$ that contains no relations. This encourages the LLM not to generate spurious relations for such "empty" sentences.

Memory-read data. For effective memory reads, the LLM has to learn (i) to identify where to initiate a query to the memory, (ii) to generate queries that retrieve helpful information from the memory and (iii) to make good use of the information that is returned by the memory. We now describe how we generate our training data with all three capabilities in mind.

Given a Re-DocRED document d, we generate a different training instance d' for each memory-read API call. To produce d', we scan d's annotated entity mentions from the beginning to the end of the document. For each entity mention e_{target}, we collect all relation triples in which it participates. Such triples are a good basis for memory-read API calls that – when issued before e_{target} first appears – will help the LLM to correctly generate e_{target}; this is why we refer to e_{target} as the target entity. We keep only that subset of the triples in which the mention of the other entity e_q that the triple refers to (the query entity) has already occurred. (The LLM will in general not be able to generate a query containing e_q if e_q has not yet occurred.) We also discard all triples that we previously encountered during our scan. (These are already known at this point, so there is little utility initiating a query for them.) We then generate a query for each remaining triple: either $\langle e_q, t, * \rangle$ (e_{target} = object) or $\langle *, t, e_q \rangle$ (e_{target} = subject). The memory-read API call for the queries generated for e_{target} is placed immediately preceding e_{target}. This will retrieve e_{target} from the memory in many cases and then make it easy for the LLM to correctly predict e_{target} at the next position.

Next we retrieve results for the query from the memory. The memory we use here is the one that is populated from Wikipedia by the trained memory-write model; see §3.3 and Figure 2a. The memory write model misses some relations and incorrectly identifies others, resulting in an imperfect memory. We intentionally use this imperfect memory because it aligns the training data with the ultimate inference conditions.

If the query returns a large number of results from memory (more than $Q_{thr} = 30$), we discard it as unlikely to be helpful. (See Appendix B for details.) An example is the query $\langle *, \text{country}, \text{United States} \rangle$ where Wikidata defines the relation "country" as "sovereign state that this item is in". There are thousands of entities that satisfy this query. Such an unspecific result is not useful. Otherwise we add the queries and the query result to d'; see Figure 2b and §3.2.

Finally, we add the rest of d to d' up to the next memory read (indicated by "({") or (if there isn't one) the entire remainder of d.

To summarize, each training example d' is a concatenation of (i) the pretext, including the first two letters ("({") of the API call, (ii) the API call proper "MEM_READ(e_s^{q1}»t^{q1}»;...)-->", (iii) the query result from the memory "$e_1, e_2, e_3, \ldots$})"

and (iv) the posttext. The posttext consists of the rest of the following text until the next memory read or (if there isn't one) the entire rest of d.

The loss is applied to the API call (ii) – this teaches the model to generate the correct API call. The loss is also applied to the posttext – this teaches the model (a) to make good use of the information provided in the query result for predicting entities and (b) to predict the next memory read (as indicated by "`({`"). (iii) is not subject to the loss since the query results are generated by the memory, not by the LLM. For the training example d' that contains the very first "`({MEM_READ(`" in the document (and only for this d'), the loss is also applied to the pretext – because the LLM needs to learn where to generate this first "`({MEM_READ(`".

4 Experiments

To train and evaluate MEMLLM, we construct training and evaluation datasets as described in §3.3. We require datasets annotated with entities and relations. We use three such datasets. (i) Re-DocRED (Tan et al., 2022): Wikipedia texts annotated (in a Wikidata format) with named entity mentions, coreference information and 96 relations (occurring intra- and inter-sentence). Re-DocRED includes many relation instances missing in DocRED (Yao et al., 2019). (ii) DocRED's distant supervised training set. It includes >100K documents but fewer relations per document. The size of this dataset makes the training more effective and robust. (iii) A "counterfactual" version of Re-DocRED (Anon., 2024). Anon. (2024) introduces an entity replacement strategy to find and apply suitable replacements over Re-DocRED.[2] In our initial tests, we found that teaching the model to produce counterfactual answers (which often contradict its parametric memory) increases robustness against pretrained knowledge bias.

Re-DocRED is human-annotated and mostly consists of relations with explicit evidence. In contrast, the distant supervised DocRED training set lacks explicit evidence and contains many false positives due to its automated annotation method. To address this, we implement a few-shot-based filtering approach to remove false-positive relations. We also apply this filtering to Re-DocRED relations that lack explicit evidence. See Appendix C for details. For finetuning, we first train on distant supervised and counterfactual data. Then we continue the training on Re-DocRED.

We finetune two Mistral-7B (Jiang et al., 2023) models using LoRA (Hu et al., 2022), a memory-write model and a memory-read model. See Appendix D for details on finetuning and hyperparameters. Our baselines are the original Mistral-7B and the memory-read model with its memory capabilities disabled. The latter baseline lets us ascertain to what extent improvements are due to in-domain finetuning (as opposed to the memory).

4.1 Perplexity evaluation

Following Liu et al. (2022), we report: (1) OVERALL PPL (PPL on the entire input text), (2) TARGET PPL (PPL on the target entities) and (3) ENTITY PPL (PPL on all named entities). The model produces a token w_i with probability $p(w_i|w_{<i})$:

$$\begin{aligned} p(w_i|w_{<i}) &= p(w_i|w_{<i}, \mathrm{MR})p(\mathrm{MR}|w_{<i}) \\ &\quad + p(w_i|w_{<i})(1 - p(\mathrm{MR}|w_{<i})) \end{aligned}$$

where $p(\mathrm{MR}|w_{<i})$ is the probability of initiating a memory read (MR) with the "`({`" token.[3]

In case of MR, w_i is conditioned on both MR (including the MR call and the returned result, see Figure 2b) and the pretext $w_{<i}$. If there is no MR, then w_i is only conditioned on $w_{<i}$.

Table 1 gives perplexity results on Re-DocRED test. MEMLLM outperforms the two baselines on all three PPL measures (①). This increase for triples appearing for the first time in the text suggests that memory-reads successfully recall relevant information for language modeling. This improvement benefits not just all entities in the text (ENTITY PPL) but the entire text (OVERALL PPL). TARGET PPL (the focus of this work) improves by .162 (1.094 vs 1.256). In comparison to the relatively small in-domain improvement (1.590 vs 1.609), the substantial .162 improvement demonstrates the effectiveness of MEMLLM for target entities. This capability is crucial for generating factual text and preventing hallucinations.

For our **memory-read analysis,** instead of using the LLM to write to the memory, we directly

[2]This dataset is also included in the supplementary.

[3]We evaluate $p(w_i|w_{<i})$ by setting $p(w_i|w_{<i}, \mathrm{MR})$ to zero for all positions except for positions where memory reads actually occur. The reason is that taking into account an MR call at each position results in a tree with 2^n leaves at position n in the text, each requiring a memory call. This is too expensive to compute. As a result, we evaluate with smaller values of $p(w_i|w_{<i})$ than the true $p(w_i|w_{<i})$ estimated by the model and, consequently, with higher perplexities, thus unfairly penalizing MEMLLM. Note that this is a problem for fairness of our perplexity evaluation, but not for a real application (where we only pursue a single path at each point).

	Memory	PPL OVERALL	PPL TARGET	PPL ENTITY
Baseline #1 (Mistral-7b)	(no memory)	1.762	1.267	1.540
Baseline #2 (Memory Disabled)		1.609	1.256	1.471
① **MEMLLM**	MW[*Wikipedia (Full)*]	1.590	1.094	1.432
② MEMLLM	MW[*Wikipedia (Abs.)*]	1.589	1.084	1.428
③ MEMLLM	MW[*Re-DocRED (Test)*]	1.582	1.037	1.411
④ + Gold MR Pos. & Queries		1.533	0.662	1.266
⑤ MEMLLM	Re-DocRED (Test)	1.571	0.954	1.384
⑥ + Gold MR Position		1.542	0.803	1.316
⑦ + Gold Queries		1.489	0.310	1.147
⑧ + Gold Target		1.488	0.305	1.145
⑨ + Gold Target Only		1.478	0.177	1.107

Table 1: MEMLLM performance on OVERALL PPL (all text), TARGET PPL (target entities) and ENTITY PPL (all entities). We show the effect of memory content ("**Memory**"). "MW[X]": the memory is populated with triples generated by memory-writes with MEMLLM run on X. ⑤–⑨: the triples are from Re-DocRED's validation set.

populate the memory with the relations from the validation set (indicated as "Re-DocRED (Test)" in column "Memory"). This lets us investigate what would happen if the memory-write process were error-free, i.e., all remaining errors are due to the memory-read process. We look at four potential sources of error in memory reads and present in each case an ablation in which this source of error is eliminated: the position of the memory read is the gold position immediately before the target entity (⑥ "Gold MR Position"), the query to the memory is the gold query (⑦ "Gold Queries"), the correct target entity is returned by the memory (⑧ "Gold Target"), the correct target entity is returned by the memory and no other entities (⑨ "Gold Target Only"). ⑨ is the lower bound perplexity for perfect memory reads (and perfect memory writes).

Comparing ⑨ and ⑧ on TARGET PPL (.177 vs .305) shows the effect of "ambiguity". The +.128 gap is due to the memory returning targets in addition to the gold target.

Moving from ⑧ to ⑦ (.305 to .310) indicates the impact of the memory retrieval process. In ⑦, we exclusively use the gold queries, but without ensuring the inclusion of the gold target in the results. The next comparison highlights the impact observed when the LLM itself generates queries (⑥) vs when only gold queries are created (⑦). Finally, the effect of the model itself selecting the position for the memory read (⑥) versus predetermining that position (⑤) is shown in the rise from .803 to .954.

To isolate the factor **memory-write performance**, we fix (i) gold memory-read positions and queries and (ii) the input corpus for extracting relations (we use Re-DocRED test). We only vary the method by which the memory is populated: running the memory-write model on the input corpus (④) vs reading out the relations from the gold data and directly storing them in memory (⑦). As expected, PPL improves when directly stored (i.e., 100% precision and recall) triples are used (⑦) vs when MEMLLM extracts and writes triples to memory (④). This indicates that there is room for improvement by training MEMLLM to do a better job at information extraction.

Scaling the stored triples. In a real-world scenario, the size of the memory and, consequently, the size of query results will get large. This increases the risk of unhelpful information being returned from the memory. To investigate this, we compare our main experiment (①, using the full Wikipedia, 111M relations) with two ablations that use only Wikipedia abstracts (②, 38M relations) and only Re-DocRED test (③). Table 1 shows that there is a relatively small negative effect of memory size: ③ (memory stripped down to the relations generated from Re-DocRED test) is only slightly better than ① (full memory). This suggests good scaling properties of our approach.

4.2 Knowledge Editing Evaluation

To test whether MEMLLM facilitates knowledge editing, we evaluate prompt-based knowledge editing. Following Hartvigsen et al. (2023) and Yao et al. (2023c), we measure reliability (REL), generalization (GEN) and locality (LOC). Each example includes a prompt, an edit, a generalization test prompt and a locality test prompt. The task is to apply the edit on the original prompt to the model. The goal is for the model to respond to original and generalization test prompts in accordance with the edit. The locality test checks whether unrelated knowledge is affected. An ideal method effectively applies edits, generalizes correctly and does not harm unrelated knowledge.

Following Wang et al. (2024a), we evaluate MEMLLM on ZsRE, a closed-book question answering dataset (Levy et al., 2017) with locality prompts selected from Natural Questions (NQ) (Kwiatkowski et al., 2019). We apply 1000 edits from the evaluation set by appending them to the end of the questions (the prompts) using the following text: "It is or they are" and bracketing them with tags. Example: "`({USER_ST})`What city was Luca Verdecchia born? It is or they are Naples`({USER_END})`." The memory-write model should then extract and store Verdecchia's place of birth, i.e., Naples. We evaluate MEMLLM using a 5-shot QA prompt. The first four examples are typical question-answer pairs. The fifth in addition includes a full memory-read call. A prompt – a generalization or locality test prompt – is appended to the 5 shots and a memory-read API call executed after the question mark.

We expect the model to answer the questions based on the memory filled with the edits. Some edits in the dataset overlap or are intended to replace previous edits. Therefore, if a newly extracted triple has an exact matching entity and relation with an old triple, we replace the old one with the new one.

Table 2 compares knowledge editing results for MEMLLM with three baselines. MEMLLM outperforms the baselines (AVG of .84). High reliability (.78) and generalization (.76) scores suggest that MEMLLM (i) manages to extract and store the relation triple based on the edit and (ii) utilizes the edit in the memory-read process to answer the original and the generalization test questions correctly. Moreover, since MEMLLM uses an explicit memory the applied edits only mildly affect the answers to unrelated questions: MEMLLM has a score of .97 on locality. This indicates that there is little cumulative deterioration of the explicit memory.

Method	REL	GEN	LOC	AVG
DEFER	0.02	0.02	0.67	0.24
GRACE	**1.00**	0.02	**1.00**	0.67
WISE	0.70	*0.67*	**1.00**	*0.79*
MEMLLM	*0.78*	**0.76**	*0.97*	**0.84**

Table 2: Knowledge editing results on ZsRE with 1000 sequential edits. AVG: mean of REL, GEN, LOC. Baseline results (using the same model, Mistral-7B, and edit set) are from Wang et al. (2024a). Bold (italics): (second) best result.

Qualitative Analysis. Leveraging MEMLLM's interpretable design, we identified the causes behind the 22% performance gap in reliability (REL, MEMLLM vs GRACE). Out of 216 errors, 45 were due to memory writes resulting in no triples or triples without the desired edit. In 95 cases, the edit was captured in the memory write but not retrieved by the memory read, either due to a bad query or incorrect relation extraction during the memory write. Another 63 errors occurred when the model did not effectively use the edits even though they were correctly retrieved. Many of these errors are due to the limitation to 96 relations (see §4). For example, the question "How endangered does the IUCN consider Hyloxalus parcus?" involves a relation that is not covered: "IUCN conservation status". In another case, the question "What family lineage was Xiao Jia part of?" retrieves the correct edit ("Southern Ming Dynasty") but for an incorrect relation: `({MEM_READ(Xiao Jia»country of citizenship»)})`, as the relation "family" is not one of the covered 96. The model may then not recognize the query result as relevant to the question and ignore it. Addressing this limitation by supporting more relations would resolve many of these errors. Even with this limitation and not being specifically designed for knowledge editing, MEMLLM outperforms other model editing methods. We believe that expanding its capability to handle a broader range of relations would greatly enhance its performance.

5 Conclusion

We present MEMLLM, a novel approach to endowing an LLM with an explicit structured memory. We publish a training dataset that can be used to extend any standard LLM with such a memory. We show that MEMLLM improves language modeling (as measured by entropy) and outperforms state-of-the-art knowledge editing methods on ZsRE.

Limitations

While the structured relation-based memory improves factuality and interpretability, it has its own limitations. The current version of MEMLLM handles only 96 relation types commonly used in Wikidata. However, to handle all types of knowledge extraction and storage, the model should be capable of extracting other types of relations. Composite relations that could be inferred from multiple already extracted relations are not detected or utilized in the current version of MEMLLM. For instance, if we extract (California, country, United States) and (Apple Inc., located in, California), we expect the relation (Apple Inc., located in (or Country), United States) to be inferred. MEMLLM is not a memory-aware solution. This means if a fact is not stored in the memory, but the decoding process generates a partial prompt that requires that fact, the model would either continue generation based on its parametric knowledge or hallucinate. We refer all these limitations to future work, as in this paper we have laid the initial groundwork for building a more complex and comprehensive method.

References

Anon. 2024. Consistent document-level relation extraction via counterfactuals. Submitted; This reference is anonymized to comply with the double-blind review regulation.

Aydar Bulatov, Yuri Kuratov, and Mikhail Burtsev. 2022. Recurrent memory transformer. In *Advances in Neural Information Processing Systems*.

Mikhail S Burtsev, Yuri Kuratov, Anton Peganov, and Grigory V Sapunov. 2020. Memory transformer. *arXiv preprint arXiv:2006.11527*.

Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. 2023. Walking down the memory maze: Beyond context limit through interactive reading. *arXiv preprint arXiv:2310.05029*.

Xin Cheng, Di Luo, Xiuying Chen, Lemao Liu, Dongyan Zhao, and Rui Yan. 2023. Lift yourself up: Retrieval-augmented text generation with self-memory. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024. Model editing can hurt general abilities of large language models. *arXiv preprint arXiv:2401.04700*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with grace: Lifelong model editing with discrete key-value adaptors. In *Advances in Neural Information Processing Systems*.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2024. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 36.

Zexue He, Leonid Karlinsky, Donghyun Kim, Julian McAuley, Dmitry Krotov, and Rogerio Feris. 2024. Camelot: Towards large language models with training-free consolidated associative memory. *arXiv preprint arXiv:2402.13449*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Chenxu Hu, Jie Fu, Chenzhuang Du, Simian Luo, Junbo Zhao, and Hang Zhao. 2023. Chatdb: Augmenting llms with databases as their symbolic memory. *arXiv preprint arXiv:2306.03901*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2023. Transformer-patcher: One mistake worth one neuron. In *The Eleventh International Conference on Learning Representations*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

Joel Jang, Seonghyeon Ye, Changho Lee, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, and Minjoon Seo. 2022. TemporalWiki: A lifelong benchmark for training and evaluating ever-evolving language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6237–6250, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Samy Jelassi, David Brandfonbrener, Sham M Kakade, and Eran Malach. 2024. Repeat after me: Transformers are better than state space models at copying. *arXiv preprint arXiv:2402.01032*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Jungo Kasai, Keisuke Sakaguchi, yoichi takahashi, Ronan Le Bras, Akari Asai, Xinyan Velocity Yu, Dragomir Radev, Noah A. Smith, Yejin Choi, and Kentaro Inui. 2023. Realtime QA: What's the answer right now? In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Xinnian Liang, Bing Wang, Hui Huang, Shuangzhi Wu, Peihao Wu, Lu Lu, Zejun Ma, and Zhoujun Li. 2023. Enhancing large language model with self-controlled memory framework. *arXiv preprint arXiv:2304.13343*.

Qi Liu, Dani Yogatama, and Phil Blunsom. 2022. Relational memory-augmented language models. *Transactions of the Association for Computational Linguistics*, 10:555–572.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.

Pedro Henrique Martins, Zita Marinho, and Andre Martins. 2022. ∞-former: Infinite memory transformer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5468–5485, Dublin, Ireland. Association for Computational Linguistics.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore. Association for Computational Linguistics.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. In *International Conference on Learning Representations*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022. Memory-based model editing at scale. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 15817–15831. PMLR.

Ali Modarressi, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. 2023. Ret-llm: Towards a general read-write memory for large language models. *arXiv preprint arXiv:2305.14322*.

Charles Packer, Vivian Fang, Shishir G Patil, Kevin Lin, Sarah Wooders, and Joseph E Gonzalez. 2023. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *In the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*, UIST '23, New York, NY, USA. Association for Computing Machinery.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitry Pyrkin, Sergei Popov, and Artem Babenko. 2020. Editable neural networks. In *International Conference on Learning Representations*.

Qingyu Tan, Lu Xu, Lidong Bing, Hwee Tou Ng, and Sharifah Mahani Aljunied. 2022. Revisiting DocRED - addressing the false negative problem in relation extraction. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8472–8487, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024a. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *arXiv preprint arXiv:2405.14768*.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023a. Augmenting language models with long-term memory. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yu Wang, Xiusi Chen, Jingbo Shang, and Julian McAuley. 2024b. Memoryllm: Towards self-updatable large language models. *arXiv preprint arXiv:2402.04624*.

Zekun Wang, Ge Zhang, Kexin Yang, Ning Shi, Wangchunshu Zhou, Shaochun Hao, Guangzheng Xiong, Yizhi Li, Mong Yuan Sim, Xiuying Chen, et al. 2023b. Interactive natural language processing. *arXiv preprint arXiv:2305.13246*.

Qingyang Wu, Zhenzhong Lan, Kun Qian, Jing Gu, Alborz Geramifard, and Zhou Yu. 2022a. Memformer: A memory-augmented transformer for sequence modeling. In *Findings of the Association for Computational Linguistics: AACL-IJCNLP 2022*, pages 308–318, Online only. Association for Computational Linguistics.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023a. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. DocRED: A large-scale document-level relation extraction dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 764–777, Florence, Italy. Association for Computational Linguistics.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023b. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240, Singapore. Association for Computational Linguistics.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023c. Editing large language models: Problems, methods, and opportunities. *CoRR*, abs/2305.13172.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2023. Generate rather than retrieve: Large language models are strong context generators. In *The Eleventh International Conference on Learning Representations*.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. 2024. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Peng Cui, Tiannan Wang, Zhenxin Xiao, Yifan Hou, Ryan Cotterell, and Mrinmaya Sachan. 2023. Recurrentgpt: Interactive generation of (arbitrarily) long text. *arXiv preprint arXiv:2305.13304*.

A Memory-write Decoding Method

While one might use MEMLLM with greedy decoding for memory writes, we suggest that the finetuned model may end the memory-write too early, before completely extracting all relations. Therefore, to ensure the model captures all relevant relations, we implement a late stopping strategy. In this approach, similar to greedy decoding, we consistently select the top-scoring token as the next token, unless it's the closing token ")}". If the closing token scores highest, we note its position, calculate the average log probability score of the sequence up to that point, and proceed with the second highest scoring token—-typically the ";" separator—-resuming greedy decoding. By tracking the positions where the closing token was predicted, along with their corresponding logprob scores, we maintain the generation process until there are no enhancements in the scores for K=5 consecutive times. Subsequently, we halt the generation and select the position with the highest score as the cutoff point.

B Filtering Ambiguous Queries

As we aim to assist the model with the stored memory content, having concise query results would facilitate reaching this objective. Getting precise outputs from the memory would require queries that are tailored in a way which lead to an exact match or related entities to the target entity. To reduce the chances of getting a vast and wide-range amount of outputs from the memory, we exclude queries that potentially leads to such results. In Table 3, we demonstrate query patterns that we intuitively assume based on the queried entity and the relation type that would lead to an ambiguous result. Therefore, we drop any query that would match with one of the mentioned patterns.

C Filtering Distant Supervision Relations

To increase the number of training examples, we also include examples from the distant supervision subset of DocRED. Distant supervision (Mintz et al., 2009) assumes that a relation r exists between two entities (e_s, e_o) in a text if the text includes both entities and the $r = \langle e_s, t, e_o \rangle$ relation triple exists in a knowledge base. While this method is valuable for relation extraction, it may introduce noisy examples without any evidence of the relation in the text. This noise could adversely affect our training pipeline.

The experimental setup is as follows: We start with a partial document ($S = \{s_1, s_2, \ldots, s_i\}$) mentioning two entities (e_1, e_2), with at least one of them present in the last sentence (i.e., the focus sentence), s_i. Our aim is to determine whether the potential relation r between e_1 and e_2 has any evidence in the last sentence.

To filter out negative examples, we use large language models (i.e., Mixtral). We design 8-shot in-context learning examples to detect if there is evidence of a relation in the focus sentence. Initially, we curate a test set to evaluate the performance of this filtering mechanism. We select 1000 examples from the human-annotated split of DocRED as positive examples where the focus sentence is annotated as evidence. For negative examples, we choose 1000 examples where the focus sentence contains at least one entity but there is no evidence for the relation in the focus sentence.

For prompting, we apply three different strategies. In the first approach (baseline), we expect the LLM to answer with “Yes” or “No” to report if the focus sentence contains evidence. With the second approach (justification), we expect the LLM to provide justification after giving the answer. In the final approach (reasoning), we expect the LLM to generate a natural sentence representing the relation, then provide reasoning, and finally generate the answer with “Yes” or “No” in the last sentence, similar to chain-of-thought prompting.

We present the initial results in Table 4. The

Query ($\mathbf{q}$)	Relation type (t^q)
$\langle *, t^q, e_o^q \rangle$	country of citizenship, country, country of origin, religion, place of birth, place of death, work location, location, basin country, residence, location of formation, publication date, production company, platform, original language of work, applies to jurisdiction, located in the administrative territorial entity, headquarters location, inception, employer, date of birth, date of death, educated at
$\langle e_s^q, t^q, * \rangle$	contains administrative territorial entity

Table 3: List of ambiguous queries subject to the filtering process.

Filtering Approach	Prec.	Rec.	F1	Acc.
Baseline	0.58	0.83	0.68	0.61
Justification	0.56	0.82	0.66	0.59
Reasoning	**0.78**	**0.84**	**0.80**	**0.80**

Table 4: Comparing performance of different prompting strategy for filtering distant supervision data. The reasoning approach similar to chain-of-thought prompting performs best among other strategies.

results suggest that the reasoning approach outperforms the other two approaches by a large margin. Also, it suggests that the filtering would lead to higher quality based on the high recall score, 0.84. We demonstrate the best-performing prompt in Figure 4.

After applying this method, we combine the filtered distant dataset with the human-annotated data. Due to the annotated data's significantly smaller size compared to the distant split, we oversample the former by a factor of 10 before incorporating it into the training data.

D Hyperparameters Details

We finetune MEMLLM, with a Mistral-7B-v0.1 model (Jiang et al., 2023) using an Adam optimizer (Kingma and Ba, 2015), with the learning rate set to 2×10^{-5}, 2 epochs, and a batch size of 96. For LoRA specific parameters, we apply a dropout rate of 0.1, with a rank of 16 and an alpha weight of 8.

To select the memory retrieval hyperparameters (§3.1), we must balance explicitness with the need to accommodate variations in entity mentions and relation types. This balance is influenced by the data and the entities involved, but generally, a larger τ_e increases explicitness. However, it also limits the number of similarly mentioned entities that can be retrieved, which depends on the use case. A smaller τ_e could retrieve more entities, but it would also increase query execution time. The selection of τ_t depends on the supported relation types and the required flexibility in retrieving closely related relation types. For instance, in model editing, where handling loosely similar relation types is necessary, a more relaxed τ_t value is appropriate. Finally, τ_r determines the final number of outputs retrieved during the memory-read. A larger τ_r makes the memory more explicit in both entity and relation type. We set τ_e and τ_t to 0.7 and τ_r to 0.85. We set these values to $\tau_e = 0.85$, $\tau_t = 0.2$ and $\tau_r = 0.6$ respectively for model editing experiments.

Reasoning Prompt - Distant Supervision Filtering

To determine whether the main sentence contains information about the given relation, both the main sentence and the context will be provided. The goal is to identify whether there is evidence of the relation in the main sentence, supported by the context. If there is no relation or the evidence exists solely in the context without requiring the main sentence, respond with No. Otherwise, respond with Yes. Provide reasoning to support your response.
Context:
Main Sentence: James Michael Osting (born April 7 , 1977) is a former Major League Baseball pitcher .
Relation: ("Osting", "date of birth", "April 7 , 1977")
Evidence: The relation indicates that Osting was born on April 7, 1977. The main sentence explicitly mentions that Osting was born on April 7, 1977. The answer is Yes.

Context: Splashdown is a Hot Tuna album released in 1984 containing the tracks from a previously unreleased live acoustic performance that had been played on the short - lived radio station WQIV in the mid-1970s . During the recording , news of the Apollo - Soyuz mission returning to Earth after the first USA - USSR rendezvous in space reached the station , and the astronauts ' radio transmissions were played at the same time as Jorma and Jack continued with " Police Dog Blues . " The transmissions mixed with the song were preserved for this release as the last track of side 1 .
Main Sentence: The album was Hot Tuna 's first release on Relix Records , and one of the first Relix releases .
Relation: ("Hot Tuna", "country of origin", "USA")
Evidence: The relation indicates that the origin of Hot Tuna is the country of the United States. There is no evidence in the main sentence regarding the country of origin of Hot Tuna. The answer is No.

Context:
Main Sentence: The Chemung Canal Bank Building is located at 415 East Water Street in Elmira , Chemung County , New York , United States .
Relation: ("Chemung County", "capital", "Elmira")
Evidence: The relation indicates that Elmira is the capital of Chemung County. The main sentence only specifies the location of Elmira within Chemung County but does not mention Elmira as the capital of Chemung County. The answer is No.

Context: Carrie Lam Cheng Yuet - ngor , GBM , GBS (; born 13 May 1957) is the 4th and current Chief Executive of Hong Kong . Before that she was the Chief Secretary for Administration , the most senior rank of principal officials of Hong Kong , from 2012 to 2017 .
Main Sentence: After graduating from the University of Hong Kong , Lam joined the civil service in 1980 and served in various bureaux and departments .
Relation: ("Lam", "educated at", "University of Hong Kong")
Evidence: The relation indicates that Lam received education at the University of Hong Kong. The main sentence mentions that Carrie Lam Cheng Yuet-ngor graduated from the University of Hong Kong. The answer is Yes.

Context: Pacific Fair is a major shopping centre in Broadbeach Waters on the Gold Coast , Queensland , Australia . It was Queensland 's largest regional shopping centre until 2006 . Pacific Fair was developed by Hooker Retail Developments and opened in 1977 on what was swampland with 96 specialty stores and two anchor tenants . Since then , Pacific Fair has undergone numerous expansions and has grown to have more than 300 specialty stores and four anchor tenants . In January 2014 , work began on a major redevelopment project to meet the predicted regional growth on the Gold Coast . Prior to the redevelopment , the shopping centre had four main major stores including a four - level Myer , Kmart , Target , Coles and Toys ' R ' Us . Daimaru operated in the centre before its Australian withdrawal , albeit briefly .
Main Sentence: It also had a 12-screen Birch Carroll and Coyle Cinema (re - opened as Event Cinemas in late 2015) .
Relation: ("Event Cinemas", "country", "Australia")
Evidence: The relation indicates that Event Cinemas is located in the country of Australia. The main sentence mentions that Event Cinemas is part of Pacific Fair which is located in Australia. The answer is Yes.

Context: Benjamin Winslow Harris (November 10 , 1823 - February 7 , 1907) was a nineteenth - century politician , lawyer and judge from Massachusetts . He was the father of Robert Orr Harris . Born in East Bridgewater , Massachusetts , Harris pursued an academic course at Phillips Academy , Andover , graduating in 1847 . He graduated from Dane Law School of Harvard University in 1849 . He was admitted to the bar in Boston , Massachusetts in 1850 , commencing practice in East Bridgewater . He served in the Massachusetts Senate in 1857 , was a member of the Massachusetts House of Representatives in 1858 , was district attorney for the southeastern district of Massachusetts from 1858 to 1866 and was collector of internal revenue for the second district of Massachusetts from 1866 to 1873 . Harris was elected a Republican to the United States House of Representatives in 1872 , serving from 1873 to 1883 , not being a candidate for renomination in 1882 . There , he served as chairman of the Committee on Naval Affairs from 1881 to 1883 . Afterwards , he resumed practicing law in East Bridgewater , Massachusetts and was judge of probate for Plymouth County , Massachusetts from 1887 to 1906 .
Main Sentence: Harris died in East Bridgewater on February 7 , 1907 and was interred in Central Cemetery in East Bridgewater .
Relation: ("Benjamin Winslow Harris", "place of birth", "East Bridgewater")
Evidence: The relation indicates that Benjamin Winslow Harris was born in East Bridgewater. The main sentence lacks information about Benjamin Winslow Harris's place of birth. The evidence for East Bridgewater as his birthplace is exclusively found in the context, not in the main sentence. The answer is No.

Context: Greatest Hats is the first compilation album by the Canadian new wave / synthpop group Men Without Hats , released in 1996 .
Main Sentence: A slightly modified version of the album was released in the US in 1996 , entitled Collection .
Relation: ("Collection", "performer", "Men Without Hats")
Evidence: The relation indicates that Men Without Hats is the performer of the Collection album. The main sentence says that Men Without Hats released slightly modified version of the Greatest Hats album which is the album Collection. The answer is Yes.

Context: Aaron Hobart (June 26 , 1787 - September 19 , 1858) was a U.S. Representative from Massachusetts . Born in Abington , Massachusetts , Hobart pursued classical studies and graduated from Brown University in 1805 . He studied law , was admitted to the bar and commenced practice in Abington . He served as member of the Massachusetts House of Representatives and served in the Massachusetts State Senate . Hobart was elected as a Democratic - Republican to the Sixteenth Congress to fill the vacancy caused by the resignation of Zabdiel Sampson . He was reelected as a Democratic - Republican to the Seventeenth Congress , elected as an Adams - Clay Republican to the Eighteenth Congress , and reelected as an Adams candidate to the Nineteenth Congress , and served from November 24 , 1820 , to March 3 , 1827 . He declined to be a candidate for renomination in 1826 .
Main Sentence: He then served as an Executive councilor 1827 - 1831 and served as probate judge 1843 - 1858 .
Relation: ("Aaron Hobart", "date of death", "1858")
Evidence: The relation indicates that Aaron Hobart passed away in the year 1858. The main sentence does not contain information about the given relation. The evidence of Aaron Hobart's date of death in 1858 is solely present in the context and is not mentioned in the provided main sentence. The answer is No.

Context: [[context]]
Main Sentence: [[focus_sentence]]
Relation: ("[[entity1]]", "[[relation]]", "[[entity2]]")
Evidence:

Figure 4: The prompt for the distant supervision dataset filtering. This prompt includes the natural representation of the relation, the reasoning, and the final answer.