Improving Factuality with Explicit Working Memory

Mingda Chen   Yang Li   Karthik Padthe
Rulin Shao   Alicia Sun   Luke Zettlemoyer   Gargi Ghosh   Wen-tau Yih
Meta FAIR
mingdachen@meta.com

Abstract

Large language models can generate factually inaccurate content, a problem known as hallucination. Recent works have built upon retrieved-augmented generation to improve factuality through iterative prompting but these methods are limited by the traditional RAG design. To address these challenges, we introduce EWE (Explicit Working Memory), a novel approach that enhances factuality in long-form text generation by integrating a working memory that receives real-time feedback from external resources. The memory is refreshed based on online fact-checking and retrieval feedback, allowing EWE to rectify false claims during the generation process and ensure more accurate and reliable outputs. Our experiments demonstrate that EWE outperforms strong baselines on four fact-seeking long-form generation datasets, increasing the factuality metric, VeriScore, by 2 to 6 points absolute without sacrificing the helpfulness of the responses. Further analysis reveals that the design of rules for memory updates, configurations of memory units, and the quality of the retrieval datastore are crucial factors for influencing model performance.

1 Introduction

In the realm of long-form text generation, a notable vulnerability of large language models (LLMs) is their propensity for hallucination, wherein the generated text contains factually inaccurate information. By prepending the input prompt with relevant documents from trustworthy sources, retrieved-augmented generation (RAG) (Lewis et al., 2020; Shi et al., 2024) has been shown to be a simple yet effective approach that substantially mitigates the hallucination issue. To further enhance the factual accuracy of model output, various iterative prompting methods have been proposed that build upon RAG. For instance, FLARE (Jiang et al., 2023) generates responses sentence by sentence, and if a newly generated sentence contains low-probability tokens, it retrieves a new set of documents and re-runs RAG to regenerate the sentence. Alternatively, Self-RAG (Asai et al., 2024) employs a self-critic component to verify the correctness of each partial generation and repeatedly queries a retrieval system to update the background knowledge, thereby producing more accurate and faithful responses. While these systems demonstrate significant empirical improvement, they are restricted in the traditional RAG design. Context-relevant knowledge through retrieval is the only online feedback to the model, incorporated as part of the input string.

In this work, we propose EWE (**E**xplicit **W**orking m**E**mory), an iterative framework that aims to provide more factual responses for knowledge-intensive long-form generation, with the help of an auxiliary fact-checking module. EWE augments an existing language model with an explicit working memory, which keeps track of the knowledge that is most relevant and useful at the current generation timestep. The memory is initially filled with the latent representation of some retrieved passages relevant to the input prompt. During the generation process, EWE actively monitors the newly generated partial response and pauses occasionally to refresh the memory with knowledge from retrieval and to check the output statement. If the statement is factually incorrect, it then refreshes the memory with the fact-checking feedback. With the updated memory, EWE first removes the incorrect statement and backtracks to the previous timestep, and then continues the generation
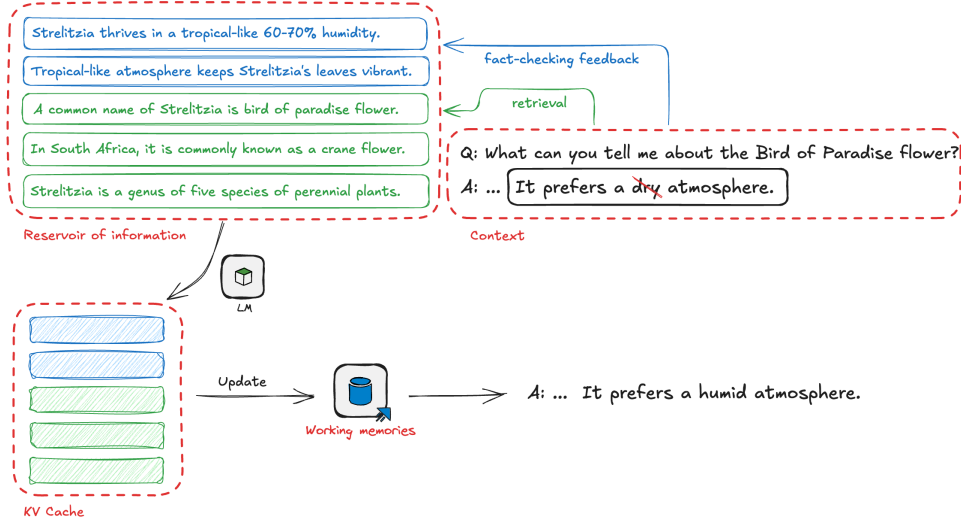
11199

Figure 1: Example pipeline illustrating how EWE pauses, receives feedback from retrievers and fact-checkers, and then re-generate to correct factual errors in its outputs. EWE handles knowledge from fact-checkers and retrievers separately as they tend to provide information with distinct properties. Retrieval offers more general background information, while fact-checkers focus more on specific details, targeting particular aspects of the output text.

process from there.

We assume that the main text generation model used here is a Transformer-based large language model, such as Llama (Dubey et al., 2024). Similar to the standard RAG setting, given an input prompt, we first retrieve k relevant text chunks of the same number of tokens, as the background knowledge. Unlike RAG, which directly prepends the input prompt with these k chunks, we apply the language model to them separately and store the KV cache of each chunk in a memory of k units. When predicting the next token, the language model effectively attends the current token to all k chunks in parallel, using their KV caches stored in the memory, and have the average as the attention value. When EWE pauses the generation and checks the newly generated, partial response, it has the opportunity to update the memory in multiple ways to guide the language model. For instance, if some claims in the new sentence are not supported, this feedback along with additional supporting documents can be used as a new unit appended to the existing memory. In addition, if the knowledge from an initial retrieved passage is no longer relevant, its corresponding memory unit can be removed or updated with embeddings of a new passage retrieved using the generated partial response as query.

EWE can be seen as a more general framework that subsumes many existing approaches. For example, if there is no stopping in generation and if the memory contains only one unit (i.e., $k=1$), then EWE degenerates to the simple vanilla RAG. If EWE pauses at the end of generation of every sentence and checks whether the new sentence contains any token with low probability as a proxy of factuality measure, then this particular instantiation, with one memory unit, is effectively FLARE. Notice that in a typical, more general configuration of EWE, the memory module consists of multiple units. When the memory is refreshed, not all the units need to be updated. If some knowledge is still required, their original raw data (e.g., passages) will not be reprocessed to create the embeddings, saving some redundant computational cost at inference time. While conceptually simple, the working memory design in EWE provides a more flexible and yet efficient way to incorporate various types of external online information, as different forms of feedback are encoded in parallel and stored in memory units (e.g., see Figure 1). It is worth noting that EWE is well-suited for real-time applications involving streaming user data, iterative retrieval, and real-time feedback. For example, when users input data through speech, where edits to pre-

viously entered information are less likely to occur, EWE allows us to start gathering relevant information and drafting responses concurrently. We notice that the design of leveraging working memory is also very related to some recently proposed methods for long-content models (e.g., Memory[3] (Yang et al., 2024)). If our memory is used only for encoding the knowledge from the passages in our corpus, then this can be viewed as the whole corpus is used as the context, along with the prompt, as the input to the model. The key differences are that EWE does not pre-encode every passage (although the KV caches of some frequently retrieved passages can certainly be precomputed in advance) and its memory can be dynamically updated as the generation progresses.

We demonstrate empirically that EWE generates more factual responses without sacrificing the relevance to the input questions, using four fact-seeking long-form generation datasets. In general, with the feedback from online fact-checking and targeted retrieval, EWE increases VeriScore (Song et al., 2024), the factuality metric we use, by 2 to 6 points absolute and is equally helpful in terms of instruction following, compared to the base model Llama-3.1$_{70B}$. We also perform human evaluations to ensure that our enhancements in VeriScore effectively translate to factuality.

2 Related work

Aiming to reduce hallucination and make the LLMs generate more factual responses, our proposed framework, EWE, detects knowledge gaps and acquires relevant information as needed, incorporating feedback from auxiliary models when available. Unlike chain-of-verification approaches (Dhuliawala et al., 2024, CoVe), which rely solely on the LLM for reasoning, EWE combines adaptive retrieval augmentation and explicit memories with a focus on factuality. This section discusses related work on these two aspects.

2.1 Iterative and Adaptive Retrieval Augmentation

Retrieval-augmented generation (RAG) typically involves a single retrieval step, followed by the language model generating a complete response. However, iterative retrieval methods (Gao et al., 2024, §5) have been proposed to generate responses in multiple segments, with each segment generated using different additional information retrieved through iterative retrieval. One such approach is ITER-RETGEN (Shao et al., 2023), which uses the model output of the previous iteration to formulate the query and retrieve more relevant knowledge for the current generation. Extending iterative retrieval, the process of adaptive retrieval (Gao et al., 2024, §5) examines partially generated responses in previous iterations to decide whether retrieving new information or regenerating a segment response is needed. For instance, FLARE (Jiang et al., 2023) follows a simple sentence-by-sentence generation process to answer fact-seeking questions. In each step, it generates a temporary next sentence and examines its acceptability based on model confidence. If the sentence is deemed questionable, it retrieves new text chunks using a query based on the temporary sentence and re-generates the next sentence using standard RAG. DRAGIN (Su et al., 2024) improves upon FLARE by introducing a new model confidence measure that combines attention scores and entropy values. This allows the model to pause the generation immediately after the confidence score of a token falls below a threshold. Additionally, DRAGIN uses preceding tokens with high attention scores on the stopping token to form a keyword-based query, which helps the model make a more confident next-token prediction.

Our work shares similarities with Self-RAG (Asai et al., 2024), particularly in the use of an auxiliary model to provide feedback. Unlike confidence measures based on token probability or attention score, Self-RAG fine-tunes the model to introspectively decide when to pause generation by outputting a special `retrieve` token. This triggers the retrieval of multiple passages, which are then used separately to generate candidate segments via standard RAG. Each segment is evaluated by a "critique" model for relevance, usefulness to the original prompt, and support from the retrieved passage. The critique model's output determines whether a candidate segment is included in the final output. Similarly, previous research has investigated various meth-

ods for incorporating external feedback during the generation process (Kang et al., 2023; Gao et al., 2023; Gou et al., 2024; Li and Flanigan, 2024).

Our approach, EWE, differs from existing iterative and adaptive retrieval augmentation methods in two key aspects. Firstly, traditional retrieval augmentation is replaced with memory augmentation, where the representation is the KV cache (similar to TurboRAG (Lu et al., 2024)) instead of the raw text, and different memory chunks that encode different passages are processed in parallel. This design allows for greater flexibility in incorporating diverse information types and improves efficiency when only part of the memory is updated, as the remaining portion can be reused. Secondly, feedback from the auxiliary model is passed to the language model through memory, enabling the core language model to naturally incorporate multiple streams of information and produce better responses. This design difference sets our approach apart from existing methods and allows for more effective integration of factuality feedback from the auxiliary model.

2.2 Memories in Long-context LLMs

Incorporating a large-scale corpus as additional knowledge can be achieved by prepending the given prompt with all documents in the corpus as an extremely long context input (Lee et al., 2024) to language models. It is thus natural to see that long-context LLMs share some technical components that apply to retrieval augmentation. The memory module in EWE is analogous to the explicit memory design in Memory[3] (Yang et al., 2024). Instead of encoding the knowledge in the training corpus completely in model parameters, or incorporating the knowledge primarily through retrieval augmentation, Memory[3] encodes 128-token chunks of the training corpus using their KV caches as memories. During inference, the model generates segments of 64 tokens. At the generation of each segment, it first uses the previous segment as query to retrieve 5 most relevant memories, and attends to them when generating the next segment. Retrieving memories of KV caches has been proposed in earlier work. For instance, Memorizing Transformers (Wu et al., 2022) effectively extends the context of the language model by k nearest neighbor lookup of the past key-value pairs (i.e., long-range memory) and attends them in the last layer of the models. LongMem (Wang et al., 2023) proposed a decoupled network architecture, using the backbone language model as memory encoder and a trained residual side-network as memory retriever and reader. The top-k attention key-value pairs stored in the memory are retrieved and incorporated at inference.

While we also use explicit memories to store KV caches in EWE, our goal is to pass new information at each step in the iterative decoding process, such as new information relevant to the current context via online retrieval and feedback from auxiliary models. We allow different operations on existing memories, including update, append, or delete, providing more flexibility for various downstream tasks.

3 Method

The overall generation process of EWE is similar to the decoding process of typical Transformer-based models, with two differences: (§3.1) EWE pauses generation periodically. When a new complete sentence has been generated, EWE uses the current context to retrieve a new set of passages as knowledge feedback. In addition, it runs a fact-checking model to judge whether the sentence contains any factually incorrect statements. If the sentence does contain factual errors, the correct facts will be used as the fact-checking feedback. Both types of feedback will be added to memories, and the sentence will be regenerated if the original one has factual errors. (§3.2) The generation is *memory-augmented*. In addition to the typical context like the input sentence and tokens generated in previous timesteps, embeddings of various forms of feedback stored in the memories will influence the generated tokens through self-attention.

3.1 Real-time Feedback

Following the design of recently proposed evaluation metrics on factuality (Min et al., 2023; Wei et al., 2024; Song et al., 2024), we determine whether a sentence is factually correct by checking if all of the claims extracted from this sentence are supported. While in general,

Ewe can use any textual knowledge as feedback, we focus on providing two types of feedback when the newly generated sentence contains factual errors: *fact-checking outcomes* and *relevant knowledge*.

Fact-checking outcomes This feedback consists of the correct information that refutes the inaccurate claims, such as *"Strelitzia thrives in a tropical-like 60%-70% humidity."* that proves *"Bird of Paradise prefers a dry atmosphere."* wrong in the example in Figure 1. In this work, we adapt the claim extraction model and verification model in VeriScore (Song et al., 2024) as the fact-checking model, where the factual knowledge is derived from the Google snippets when using the extracted claim as query.

Relevant knowledge Using the original input question and the sentence being fact-checked as query, we use Contriever (Izacard et al., 2022) to retrieve passages from C4 (Raffel et al., 2020) and Wikipedia, following the setting in MassiveDS (Shao et al., 2024). Passages with retrieval scores exceeding a certain threshold will be viewed as knowledge relevant to the current context and used to update the working memories.

We pause at every T_r timesteps to gather feedback from retrievers, and T_v timesteps from fact-checkers. However, if no new sentence is generated, the feedback collection process will be skipped.[1]

3.2 Refreshing Working Memories

The working memory in Ewe consists of k memory units, where each unit is designed to store the representations of each feedback message of M tokens. When updating working memories, we follow the *first in, first out* (FIFO) rule.[2] Given refreshed text chunks

of the same length from fact-checkers and retrievers, our model encodes them into the KV cache in parallel using the same positional IDs. Working memories in Ewe are stored as part of the language models' context preceding the model's own output text and prompts, allowing for flexible updates without reprocessing generated content. As shown in Figure 2, a separate embedding store is used for preserving these memories, which are then processed at each layer by concatenating them with the context. We then apply regular self-attention and aggregate the resulting hidden vectors using normalization terms from self-attention for each memory unit. Empirically, we find that adding hidden vectors produced by context only improves the fluency of long outputs, so we keep it in our model architectures. More formally,

$$\vec{h}_n = \sum_{i=1}^{k+1} \frac{\alpha_i \vec{h}_{n_i}}{\sum_{j=1}^{k+1} \alpha_j}, \qquad (1)$$

where $\vec{h}_n$ is the output vectors for self-attention at n-th layer in LMs, $\vec{h}_{n_i}$ is the hidden vectors produced by memories concatenated with the context and by only the context vectors, and α_i is the normalization term from self-attention that leads to $\vec{h}_{n_i}$.

4 Experiments

We present the main experimental results of Ewe in this section, along with details of the datasets and evaluation metrics we used, and the baseline we compared with. In this set of experiments, we set the retrieval and verification timesteps, T_r and T_v, to be 1 and 8, respectively.

4.1 Evaluation Datasets

We evaluated Ewe and baseline models using four fact-seeking long-form generation datasets: LongFact (Wei et al., 2024; 250 prompts), Fava (Mishra et al., 2024; 141 prompts), AlpacaFact (Dubois et al., 2023; Lin et al., 2024; 241 prompts) and Biography (Min et al., 2023; 181 prompts). We include more detailed descriptions of each dataset in the appendix.

---

[1]Ewe is not significantly slower than existing iterative retrieval methods like DRAGIN and FLARE. These methods also use an iterative generation process involving re-generation and retrieval. However, without working memories, existing iterative retrieval methods are slower than EWE when it comes to incorporating new retrieved information into the context.

[2]In our preliminary experiments, we explored other possibilities of updating the memories, e.g., replacing the least relevant memories based on retrieval scores or model confidence scores (similar to what you mentioned). However, we found that these methods did not perform better than the FIFO approach. Due to the simplicity and effectiveness of FIFO, we decided to

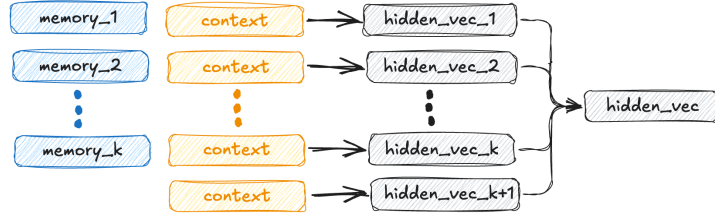leave further exploration of these alternatives for future work.

Figure 2: Diagram illustrating self-attention computations performed at each layer in EWE. We concatenate each memory with the context (except for the last hidden vector where we only use context), apply standard self-attention and then aggregate the resulting hidden vectors to produce the final hidden vectors.

4.2 Evaluation Metrics

We assess the quality of model responses to fact-seeking questions based on two key axes: *factuality* and *helpfulness*. For evaluating factuality, we considered multiple automatic metrics, such as FActScore (Min et al., 2023) and SAFE (Wei et al., 2024), but ultimately chose VeriScore (Song et al., 2024) as our primary evaluation metric. Although these metrics share a similar design that decomposes sentences into "atomic claims" and checks their support against an external knowledge source, VeriScore focuses on extracting more sensible verifiable claims and uses Google search snippets instead of Wikipedia as the knowledge source. As a result, VeriScore can be applied to responses on more diverse topics and is also more efficient, requiring fewer but more meaningful claims to be checked. We report the F_1 score from VeriScore, which is the harmonic mean of the precision and recall of the claims. Following Song et al. (2024), we set the minimum number of facts required for a model's response to achieve perfect recall as the median number of extracted claims per dataset[3]. We also used their fine-tuned models for claim extraction and verification, provided in their package[4].

To make sure that a model with a high factuality score does not simply give irrelevant but correct factual statements, we also need to check whether the response is helpful to the user. Following (Lin et al., 2024), we use AlpacaEval (Dubois et al., 2024) to compare the target model and baseline model in terms of their instruction-following ability. For the responses to the same input prompt, a large language model is used as judge to determine which of the two is better[5], and the win rate is thus used as a measure of helpfulness[6].

4.3 Baselines

We used instruction-tuned Llama-3.1 70B and 8B as the base models[7] and compared EWE with five baselines: base model only, retrieval augmentation (RA), Chain of verification (COVE)[8], an iterative retrieval approach DRAGIN (Su et al., 2024)[9], and a recently proposed semi-parametric decoding method NEST (Li et al., 2024). For base model only, Llama-3.1$_{70B}$ or Llama-3.1$_{8B}$, we simply gave the language model the prompt in the dataset and the instruction of requesting detailed information, without other additional information. With retrieval augmentation, we retrieved 10 passages using the input prompts as queries and then prepended the passages to the input[10]. NEST is a strong retrieval-based decoding algorithm. Following the original setup, we retrieved 100 passages to use as candidates. For COVE, we employ the "factor+revise" method, which Dhuliawala et al.

---

[3] The median numbers of extracted facts for LongFact, FAVA, AlpacaFact, Biography are 55, 49, 31, 43, respectively.

[4] https://github.com/Yixiao-Song/VeriScore

[5] We used GPT-4o as the judge.

[6] We found that the length-controlled win rates in AlpacaEval could conflate hallucinations and length effects, and thus report the version without length normalization.

[7] We opt to use Llama-3.1 in two different sizes instead of exploring different architectures. This decision is based on the fact that EWE is not tied to any specific model architecture. Consequently, it is more insightful to assess its generalization capabilities across various model sizes rather than across different architectures.

[8] We adapted an implementation from https://github.com/ritun16/chain-of-verification

[9] We used the authors' implementation https://github.com/oneal2000/DRAGIN

[10] Using more than 10 passages does not provide significant benefits in our preliminary experiments, so we limit our retrieval to the top 10 passages.

Model	LongFact		Fava		AlpacaFact		Biography	
	F_1	WR	F_1	WR	F_1	WR	F_1	WR
Llama-3.1$_{70B}$	64.3	-	52.0	-	63.8	-	37.1	-
+RA	72.7	41.2	56.8	37.1	66.0	43.1	43.8	49.4
+NEST	63.2	9.1	50.3	24.1	58.1	30.2	41.5	22.1
+DRAGIN	71.5	38.2	57.2	33.9	65.3	31.5	42.8	33.5
+CoVE	63.8	39.3	49.5	33.4	61.5	33.3	37.7	31.3
+CoVE w/ Retrieval	67.4	31.8	52.6	23.1	64.0	28.8	38.2	29.4
+EWE	**75.9**	**50.1**	**61.0**	**50.1**	**66.9**	**49.9**	**49.7**	**50.2**
Llama-3.1$_{8B}$	63.1	**40.6**	51.0	**36.5**	**65.3**	26.7	28.9	**24.2**
+RA	65.9	28.1	51.8	16.8	63.9	18.5	41.4	21.3
+NEST	62.3	4.2	50.2	14.1	57.8	9.1	**41.8**	21.8
+DRAGIN	63.9	15.9	51.1	10.0	61.3	11.1	34.7	11.4
+CoVE	44.1	8.8	38.7	11.0	51.3	15.1	25.1	13.3
+CoVE w/ Retrieval	53.5	12.2	39.5	5.3	54.6	12.5	29.1	10.2
+EWE	**67.3**	**40.5**	**53.1**	36.2	**65.5**	**28.0**	**42.2**	21.5

Table 1: Evaluation on factuality and helpfulness of the model responses to prompts provided in four long-form question answering datasets. For each dataset, we report F_1 scores from VeriScore and win rates (WR) from AlpacaEval. We use Llama-3.1$_{70B}$ as the baseline method in AlpacaEval win rate experiments.

(2024) demonstrated to be the most effective. Additionally, we improve CoVE by integrating retrieved passages from our retrieval datastore during the verification step. This augmentation helps us establish a stronger and more comparable baseline method, considering that most other baseline methods also utilize retrieval. For all our experiments, the maximum generation step was set to 1024. We also use C4 and Wikipedia in MassiveDS as our retrieval datastore and Contriever as the retriever. Llama-3.1$_{70B}$ is used as the baseline method for all AlpacaEval comparisons.

4.4 Results

Our main results are shown in Table 1. For the Llama-3.1$_{70B}$ base model, we find that in terms of factuality, retrieval augmentation generally improves the results consistently across different datasets. This is expected as for fact-seeking prompts, specifically conditioning generation on relevant factual knowledge has been demonstrated to be an effective way to mitigate hallucinations. NEST performs better than the base model on the Biography dataset, but not on others, and it appears that the VeriScore F_1 is lower than the standard retrieval augmentation. It might suggest that the configuration or hyperparam-

eter settings of NEST need to be further optimized, as NEST was originally evaluated by Biography with Llama-2. DRAGIN performs similarly to RA, likely because their query formulation method is not optimized for long-form generation, resulting in less useful retrieved passages. Similarly, with CoVE, we notice that it often produces shorter model responses, leading to significantly lower recall performance despite high precision, which results in a less favorable VeriScore F_1. While augmenting CoVE with retrieval slightly alleviates this issue, it still lags behind. Perhaps more interestingly, with online fact-checking feedback and refreshed knowledge from retrieval, EWE achieves the highest VeriScore F_1 on all datasets. On the helpfulness of the responses, it appears that AlpacaEval generally prefers the output from the base model, except for EWE, where the win rates are roughly 50%.

When using Llama-3.1$_{8B}$ as the base model, we have observed a similar trend. Retrieval augmentation improves factuality in terms of VeriScore F_1 and EWE still gives the best factuality results. However, compared to the models based on Llama-3.1$_{70B}$, we notice that the improvement is generally smaller. We
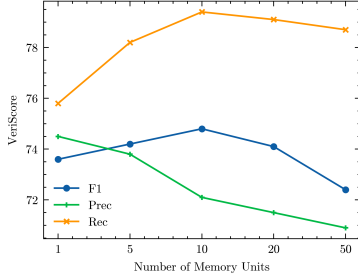
Figure 3: VeriScore over 50 prompts from Long-Fact when varying number of memory units used for storing retrieved passages and fact-checking feedback.

Figure 4: VeriScore over 50 prompts from Long-Fact when varying lengths of memory units used for storing retrieved passages and fact-checking feedback.

hypothesize that the smaller base language model is less capable in leveraging feedback, and may not always regenerate a sentence that is factually correct. In terms of helpfulness, we can see that EWE generally performs comparably to its base model Llama-3.1$_{8B}$, as they have similar win rates when judged against the output of the same Llama-3.1$_{70B}$ base model.

5 Analysis

We provide some insights based on different ablations in this section. Llama-3.1$_{70B}$ were used as the base model for all the experiments in this section. Due to space limits, we include analysis on model confidence, retrieval knowledge, and feedback forms for fact verifiers in the appendix.

5.1 Memory Configurations

In this analysis, we explore the influence of memory configurations on factuality, based on experiments on 50 randomly sampled prompts from LongFact. We examine this impact through two dimensions: the number of memory units and the lengths of memory units.

In Figure 3, we investigate how varying the numbers of memory units used for storing fact-checking feedback and retrieved passages may impact factuality. We apply the same configuration to both, as similar trends are observed when adjusting the settings for each type of memory. Overall, we find that having a large number of memory units for either fact-checking feedback or retrieved passages *negatively* affects precision and, consequently, factuality (though recall remains largely unaffected). This is likely because a significant amount of stale information remains in work-
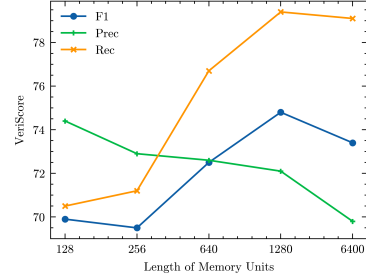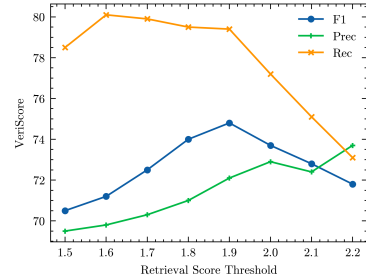

Figure 5: VeriScore over 50 prompts from Long-Fact with different retrieval score thresholds. Higher thresholds indicate passages that are less relevant would be excluded in working memory.

ing memory for an extended period without being updated, as we adhere to the FIFO rule for updating working memory. Consequently, this information becomes outdated as the generation process continues.

In Figure 4, we examine the impact of varying lengths of memory units on factuality. Our results indicate that shorter memory units tend to increase precision but decrease recall, whereas longer memory units tend to decrease precision but increase recall. We hypothesize that this occurs because short memory units allow the attention mechanism to allocate weights more effectively to individual passages. In contrast, longer memory units combine multiple passages into a single unit, which may compel the attention to focus on less relevant passages when they are grouped with more relevant ones. However, longer memory units also provide advantages in terms of improved reasoning across documents, which is beneficial for recall.

In Figure 5, we examine the impact of varying retrieval score thresholds. Notably, initially raising the threshold (allowing more rel-

RA	0.61
EWE	0.65

Table 2: Cohen's Kappa inter-annotator agreement between VeriScore and human predictions regarding whether sentences are factual or not.

evant information to be retained in working memory) enhances both precision and recall. Further increasing the threshold continues to improve precision while maintaining recall at a similar level. However, increasing the threshold beyond this point negatively affects F_1 and recall, despite still benefiting precision, likely due to the exclusion of too many retrievals. These findings indicate that our model might gain from a more advanced memory updating mechanism, which we intend to investigate in future work.

5.2 Human Evaluation

We conduct a human evaluation to determine if using VeriScore during inference impacts its alignment with human judgments. To achieve this, we randomly select 100 generation samples each from our retrieval-augmented baseline and EWE, and recruit 6 human annotators. During sample selection, we categorize sentences into buckets based on the number of nonfactual claims predicted by VeriScore. We then randomly choose samples from each bucket to ensure an equal representation of sentences with varying numbers of nonfactual claims. Annotators are given a sentence from model responses along with the full response for context. They are instructed to classify the sentence as factual, nonfactual, or "unable to decide" if no claim is present. Additionally, they are encouraged to use online search engines if they are unsure about specific information. Detailed instructions are provided in the appendix. Our evaluation gathers 120 annotations. Since annotators might choose "unable to decide" for various reasons, such as the sentence being functional and not making any claims, we exclude these from score computation. We calculate Cohen's Kappa to assess inter-annotator agreement between humans and VeriScore, excluding 25 "unable to decide" annotations. The results are presented in Table 2. Since a Cohen's Kappa score above

0.61 is considered substantial and the scores do not decrease with EWE, we conclude that using VeriScore during inference has indeed enhanced the factuality of EWE.

6 Conclusion

We present EWE, a novel system that incorporates a working memory mechanism during the generation process. EWE pauses at given intervals and refreshes its working memory based on feedback from retrieval and fact-checking models, ensuring that the generated content remains accurate and relevant. Our experiments demonstrate the effectiveness of EWE by benchmarking it on 8B and 70B Llama-3.1 models, resulting in significant improvements in both factuality and helpfulness across four fact-seeking long-form generation datasets. Furthermore, our analysis reveals that updating the working memory with more relevant information at each timestep, allowing attention to focus on each passage, and utilizing high-quality retrieval datastores with extensive knowledge coverage are crucial factors for improving factuality of models.

7 Limitations

Although EWE can be applied to various types of data, our research has been limited to English text datasets with fact-seeking prompts. Its performance on other English benchmarks and multilingual datasets remains unclear. Additionally, we have restricted EWE to only accept textual real-time feedback, and exploring multimodal feedback could be intriguing. Moreover, our human evaluation experiments have been conducted on a relatively small scale, and it would be beneficial to validate our findings in a larger-scale setting.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2024. Chain-of-verification reduces hallucination in large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 3563–3578,

Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, and Archie Sravankumar et. al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori Hashimoto. 2023. AlpacaFarm: A simulation framework for methods that learn from human feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yann Dubois, Percy Liang, and Tatsunori Hashimoto. 2024. Length-controlled alpacaeval: A simple debiasing of automatic evaluators. In *First Conference on Language Modeling*.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, and Kelvin Guu. 2023. RARR: Researching and revising what language models say, using language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16477–16508, Toronto, Canada. Association for Computational Linguistics.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proc. INLG*, pages 124–133.

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth 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*.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–

7992, Singapore. Association for Computational Linguistics.

Haoqiang Kang, Juntong Ni, and Huaxiu Yao. 2023. Ever: Mitigating hallucination in large language models through real-time verification and rectification. *arXiv preprint arXiv:2311.09114*.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. 2023. Openassistant conversations - democratizing large language model alignment. In *Advances in Neural Information Processing Systems*, volume 36, pages 47669–47681. Curran Associates, Inc.

Jinhyuk Lee, Anthony Chen, Zhuyun Dai, Dheeru Dua, Devendra Singh Sachan, Michael Boratko, Yi Luan, Sébastien M. R. Arnold, Vincent Perot, Siddharth Dalmia, Hexiang Hu, Xudong Lin, Panupong Pasupat, Aida Amini, Jeremy R. Cole, Sebastian Riedel, Iftekhar Naim, Ming-Wei Chang, and Kelvin Guu. 2024. Can long-context language models subsume retrieval, rag, sql, and more? *Preprint*, arXiv:2406.13121.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wentau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Changmao Li and Jeffrey Flanigan. 2024. Rac: Efficient llm factuality correction with retrieval augmentation. *arXiv preprint arXiv:2410.15667*.

Minghan Li, Xilun Chen, Ari Holtzman, Beidi Chen, Jimmy Lin, Wen-tau Yih, and Xi Victoria Lin. 2024. Nearest neighbor speculative decoding for llm generation and attribution. *arXiv preprint arXiv:2405.19325*.

Sheng-Chieh Lin, Luyu Gao, Barlas Oguz, Wenhan Xiong, Jimmy Lin, Wen-tau Yih, and Xilun Chen. 2024. Flame: Factuality-aware alignment for large language models. *arXiv preprint arXiv:2405.01525*.

Songshuo Lu, Hua Wang, Yutian Rong, Zhi Chen, and Yaohua Tang. 2024. TurboRAG: Accelerating retrieval-augmented generation with pre-computed kv caches for chunked text. *arXiv preprint arXiv:2410.07590*.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore. Association for Computational Linguistics.

Abhika Mishra, Akari Asai, Vidhisha Balachandran, Yizhong Wang, Graham Neubig, Yulia Tsvetkov, and Hannaneh Hajishirzi. 2024. Fine-grained hallucination detection and editing for language models. In *First Conference on Language Modeling*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Nazneen Rajani, Lewis Tunstall, Edward Beeching, Nathan Lambert, Alexander M. Rush, and Thomas Wolf. 2023. No robots. *Hugging Face repository*.

Rulin Shao, Jacqueline He, Akari Asai, Weijia Shi, Tim Dettmers, Sewon Min, Luke Zettlemoyer, and Pang Wei Koh. 2024. Scaling retrieval-based language models with a trillion-token datastore. *arXiv preprint arXiv:2407.12854*.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274, Singapore. Association for Computational Linguistics.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. REPLUG: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8371–8384, Mexico City, Mexico. Association for Computational Linguistics.

Yixiao Song, Yekyung Kim, and Mohit Iyyer. 2024. VeriScore: Evaluating the factuality of verifiable claims in long-form text generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9447–9474, Miami, Florida, USA. Association for Computational Linguistics.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. 2024. DRAGIN: Dynamic retrieval augmented generation based on the real-time information needs of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12991–13013, Bangkok, Thailand. Association for Computational Linguistics.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023. Augmenting language models with long-term memory. In *Advances in Neural Information Processing Systems*, volume 36, pages 74530–74543. Curran Associates, Inc.

Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Zixia Hu, Jie Huang, Dustin Tran, Daiyi Peng, Ruibo Liu, Da Huang, Cosmo Du, and Quoc V Le. 2024. Long-form factuality in large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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

Hongkang Yang, Zehao Lin, Wenjin Wang, Hao Wu, Zhiyu Li, Bo Tang, Wenqiang Wei, Jinbo Wang, Zeyun Tang, Shichao Song, and 1 others. 2024. Memory[3]: Language modeling with explicit memory. *arXiv preprint arXiv:2407.01178*.

A Evaluation Datasets

LongFact Designed to probe the factuality of a model of which response consists of at least several paragraphs, LongFact was created by prompting GPT-4 to generate questions regarding a specific concept or object within a given topic. In our experiments, we use the 250 prompts from the LongFact-Objects dataset, selected by Wei et al. (2024).

Fava As a new fine-grained hallucination benchmark, FAVA constructed 200 information-seeking queries that require factual knowledge to give accurate long-form answers from multiple sources, including Open Assistant (Köpf et al., 2023), No Robots (Rajani et al., 2023), WebNLG (Gardent et al., 2017) and instructions written by the authors (Mishra et al., 2024). Following Lin et al. (2024), we selected 141 prompts from this collection for our experiments.

AlpacaFact Originally collected from real-world interactions with various users, the 805

instructions in AlpacaFarm (Dubois et al., 2023) was used for evaluating the instruction-following ability of different LLMs. To focus our evaluation on factuality, we used a subset of 241 fact-seeking instructions selected by Lin et al. (2024) in this work.

Biography To demonstrate the effectiveness of the factuality metric FActScore, Min et al. (2023) selected 183 names of famous people found in Wikipedia, and applied the "*Tell me a bio of* [Person Name]" template to create a collection of prompts called Biography. As this set of prompts have been used extensively in several recent papers, we include it in our study as well.

When using these prompts, we appended the instruction "*Provide as many specific details and examples as possible (such as names of people, numbers, events, locations, dates, times, etc.)*" to encourage models to generate more detailed responses that cover multiple factoids, following Wei et al. (2024).

B   Analysis on Model Confidence

One important question remains is when to refresh the working memory. To study it, we conducted a comparative analysis of different criteria for refreshing working memory and regeneration. Since the working memory consists of the retrieval memory and fact-checking memory, which can have interacting effects, we first investigate when to trigger the retriever alone (without fact-checking memory) and then investigate when to trigger the fact-checker (when retrieval interval T_r is set to 1).

Fixed intervals for refreshing working memory As shown in Figure 6, when using a fixed retrieval interval, an smaller interval seems to perform better in general with a few spikes. This may be due to the fact that overly frequent retrieval can add irrelevant and conflicting information to the memory. On the other hand, when using a fixed verification interval, shorter interval always leads to better performance.

Model confidence for refreshing working memory In practice, fixed retrieval and verification intervals may be unnecessary and lead to sub-optimal performance. We explore whether model-confidence can serve as a signal for refreshing working memory. Specifically, we compare two different metrics for model confidence: (1) **Entropy**: average entropy of generated tokens in a sentence, and (2) **Min-prob**: minimum probability of tokens in a sentence. A higher threshold for entropy results in less frequent memory update, and a higher threshold for min-prob results in more frequent memory update. Since external fact-checkers can be computationally expensive, we first examine if we can use model confidence as a signal for retrieval and regeneration, without using an auxiliary fact-checking model to provide feedback. As shown in Figure 7 and 8 (blue line), we observe empirically intermediate thresholds for retrieval perform well, leading to to better F_1 when compared to the settings in Figure 6, where we use different fixed intervals for retrieval. With external fact-checkers, we investigate if we can use model confidence as a signal to trigger verification and regeneration to improve generation efficiency. In Figure 7 and 8 (green line), when chosen at an appropriate threshold, both entropy and min-prob can outperform the baseline (using fixed verification interval $T_v = 8$ with the same memory configuration) despite with less frequent verification.

C   Knowledge from Retrieval

We present the results of using different retrieval corpora in Table 3, including Wikipedia, C4, or both of them together. Interestingly, different datasets exhibit distinct preferences. For LongFact and Fava, C4 proves more effective than Wikipedia in helping the model to generate more factual responses. Conversely, Biography and AlpacaFact show a preference for Wikipedia. Combining C4 with Wikipedia further improves factual accuracy for LongFact and Biography, but the same trend is not observed for AlpacaFact and Fava. This is likely because the first two datasets consist of domains where C4 and Wikipedia provide complementary knowledge, whereas the latter two do not.

In Figure 10, we explore the impact of using different sizes of retrieval datastores. Both LongFact and Biography exhibit similar patterns, where increasing the datastore size gen-

Datastore	LongFact	Biography	AlpacaFact	Fava
Wiki	67.9	46.1	**55.5**	52.5
C4	70.8	44.6	53.7	**53.3**
C4 + Wiki	**74.8**	**48.4**	53.3	52.3

Table 3: VeriScore F_1 over 50 prompts from LongFact, AlpacaFact, Fava and Biography with different retrieval datastores.

erally improves results. In contrast, Fava and AlpacaFact differ, as Ewe tends to maintain similar level of VeriScore F_1 across various setups. We hypothesize that real-time feedback from fact verifiers helps offset the lack of information from retrieval datastores in these two datasets.

D Feedback Forms

In this analysis, we explore various feedback formats utilized by fact-checkers. The models in VeriScore offers 2 types of information: a list of both factual and nonfactual claims, along with relevant passages that support these factual and nonfactual judgments. To examine the impact of these feedback formats, we conduct experiments using different combinations of these information types in the working memory. For the supporting passages, we combine them using new line symbols. For the list of claims, we apply an instruction template as follows to encode nonfactual claims:

> Please refrain from including the following imprecise statements: (1) nonfactual claim$_1$ (2) nonfactual claim$_2$ ...

Our results are shown in Table 4. Overall, fact-checking feedback is beneficial compared to the base model with and without retrieval augmentation. The specific types of feedback also play a crucial role. Incorporating all feedback forms does not enhance model performance, with supporting passages proving more effective than instructions. We notice that instructing models not to generate specific details often results in misunderstanding. Models might rephrase the instruction, include the nonfactual statement in their response, and then add a clarification indicating the previous statement is nonfactual, such as *"(Note: This is a nonfactual claim and may not be accurate.)"*. We leave a better design of

feedback forms to future work. Interestingly, when we exclude all the textual feedback from fact-checkers and only pause and regenerate in the presence of nonfactual sentences, performance still slightly improves.

E Instructions for Human Annotators

Below is the instruction we provide to each annotator.

> You will be given a sentence extracted from a model response, along with the full model response and the original prompt. Your task is to assess the factuality of the provided sentence. You may use Google search to help in your evaluation. A sentence should be considered as factual only if you can locate sources that corroborate all the claims made within it. If the sentence contains no claims, you may select "unable to decide."

Passages determining a claim is incorrect	Passages determining a claim is correct	Instructions for nonfactual claims	Precision	Recall	F$_1$
✓	✓	✓	78.1	72.5	73.2
✓			77.1	73.5	72.2
	✓		77.5	72.7	73.9
		✓	71.1	76.1	72.5
✓	✓		**72.1**	**79.4**	**74.8**
			70.8	76.0	73.0
		Llama-3.1$_{70B}$	65.8	67.1	65.5
		Llama-3.1$_{70B}$ + RA	70.7	75.6	72.5

Table 4: Comparing different feedback forms for fact-checkers. We report VeriScore over 50 prompts from LongFact.
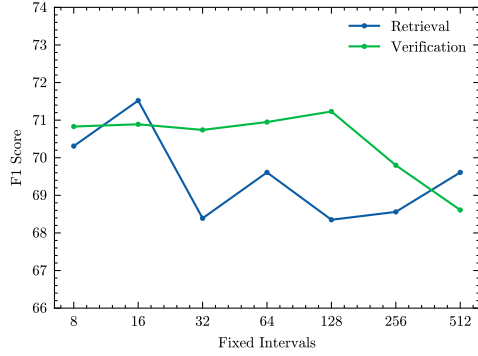
Figure 6: VeriScore F_1 when using fixed retrieval and verification intervals.
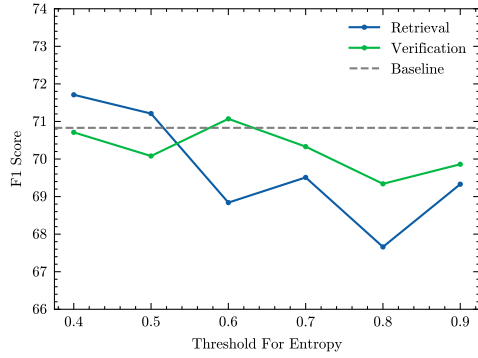

Figure 7: F_1 when using entropy thresholds for triggering retrieval and verification.
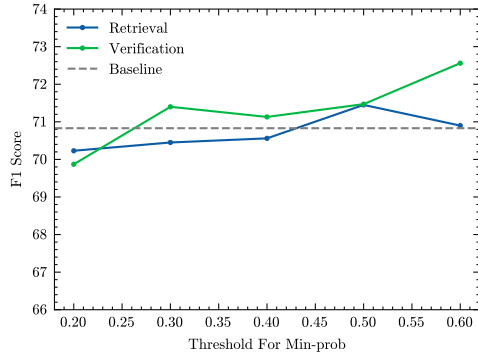

Figure 8: F_1 when using min-prob thresholds for triggering retrieval and verification.

Figure 9: Comparison of different criteria for refreshing working memory over 50 prompts from LongFact. The baseline uses retrieval interval $T_r = 1$ and verification interval $T_v = 8$.
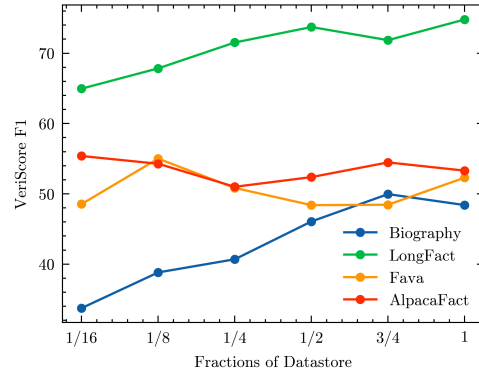

Figure 10: VeriScore F_1 when using varying fractions of datastore (C4+wiki).