

Hybrid Retrieval-Augmented Generation for Real-time Composition Assistance

Xuchao Zhang*
Guoqing Zheng

Menglin Xia*
Saravan Rajmohan

Camille Couturier
Victor Rühle

Microsoft

{xuchaozhang, mollyxia, cacoutur, zheng, saravar, viruh}@microsoft.com

Abstract

Retrieval augmented models show promise in enhancing traditional language models by improving their contextual understanding, integrating private data, and reducing hallucination. However, the processing time required for retrieval augmented large language models poses a challenge when applying them to tasks that require real-time responses, such as composition assistance. To overcome this limitation, we propose the Hybrid Retrieval-Augmented Generation (HybridRAG) framework that leverages a hybrid setting that combines both client and cloud models. HybridRAG incorporates retrieval-augmented memory generated asynchronously by a Large Language Model (LLM) in the cloud. By integrating this retrieval augmented memory, the client model acquires the capability to generate highly effective responses, benefiting from the LLM’s capabilities. Furthermore, through asynchronous memory integration, the client model is capable of delivering real-time responses to user requests without the need to wait for memory synchronization from the cloud. Our experiments on Wikitext and Pile subsets show that HybridRAG achieves lower latency than a cloud-based retrieval-augmented LLM, while outperforming client-only models in utility.

run due to the size of the model and the extra retrieval step they require, which can cause latency and limit its application in tasks requiring real-time responses, such as composition assistance.

Real-time composition tools are designed to promptly suggest next words or sentences, and therefore operate within tight latency budgets (typically in the order of 100ms or less). To avoid latency overheads for sending inference requests to the cloud, these models are usually deployed on users’ edge devices. This imposes strict constraints on the model’s size and capabilities, limiting the effectiveness of composition assistance. While recent advancements have enabled LLMs like LLAMA (Touvron et al., 2023) to generate 5 tokens per second¹, they still fall short in terms of achieving real-time response time for completing a sentence within a few hundred milliseconds. In addition, embedding a retrieval-augmentation module into the edge model may not be practical because relevant documents are often stored in the cloud, such as a company’s centralized document storage, and the retrieval step can introduce additional latency overhead.

To address the challenges previously outlined, we propose the Hybrid Retrieval-Augmented Generation (HybridRAG) framework. This framework leverages cloud-generated memory augmentation to boost the performance of small language models on edge devices, while operating *in an asynchronous manner*. The HybridRAG framework consists of a retriever model and memory generator residing on the cloud server, as well as an augmentation coordinator and memory-augmented client model deployed on client devices. The cloud model creates the retrieval-augmented memory and sends it asynchronously to the client model. This allows the client model to respond to user requests for suggestions in real-time without waiting for the cloud memory. This asynchronous communication also

1 Introduction

Retrieval-augmented approaches (Lewis et al., 2020; Liu et al., 2022) have emerged as a powerful tool to boost Large Language Model (LLM) performance by incorporating external documents (Lewis et al., 2020; Liu et al., 2022). This integration enables models like GPT3 (Brown et al., 2020) to leverage external contextual information, resulting in improved contextual understanding, streamlined integration of private data, and reduced occurrence of hallucinations. However, the retrieval-augmented large language models can be slow to

*These authors contributed equally to this work.

¹<https://news.ycombinator.com/item?id=35171116>

reduces the computational cost of the cloud model, as it does not need to process every user request.

In summary, the contributions in this work can be summarized as follows:

- *Hybrid retrieval-augmentation enables real-time generation:* A novel hybrid framework is proposed to enable real-time text generation on client devices, utilizing retrieval augmentation in the cloud server. Our approach leverages asynchronous client-cloud communication to achieve prompt responses while mitigating the effects of network latency and avoiding slow inference inherent to cloud-based retrieval-augmented LLMs.
- *Competitive utility compared to cloud-based language models:* We introduce an LLM-augmented memory approach to enhance the utility of the client language model, using GPT3-generated labels for instruction-tuning the client model. Our model effectively utilizes the LLM-augmented memory, resulting in substantial improvement in client model performance.
- *Reduced client-to-cloud communication:* Our augmentation coordinator module enables asynchronous memory augmentation, minimizing client-to-cloud communication by requesting augmented memory only when existing memory becomes stale. Additionally, utilizing GPT3-generated bullet points as memory further minimizes data transfer volume.

To assess the effectiveness of our proposed approach, we conducted experiments using 5 benchmark datasets spanning various domains. Our model demonstrated superior performance compared to the best hybrid baseline, achieving a notable average improvement of 48.6% in GLEU score through LLM-generated memory augmentation, and an additional 9.5% improvement through instruction tuning on client model. In addition, our asynchronous framework demonstrated an impressive 138-fold speed improvement compared to a synchronous approach within the same experimental setup. We plan to make our code and datasets public at a later time.

2 Related Work

The concept of hybrid computing between edge and cloud devices originated outside the realm of ma-

chine learning literature. Hybrid edge-cloud computing involves the collaboration between edge and cloud devices for computation. This approach typically divides processing tasks between the edge and the cloud, effectively addressing the limited computation capabilities of low-power edge devices and enabling real-time responses of critical services such as autonomous driving (Loghin et al., 2019; Wang et al., 2020). However, literature on hybrid computing for machine learning models is relatively scarce.

To our knowledge, the most relevant topic in the literature regarding hybrid inference for machine learning is split computing, which involves partitioning modules of machine learning pipelines or layers of neural network models between edge and cloud devices (Matsubara et al., 2022). For example, in the work of Osia et al. (2020), a deep neural network model for image classification is split into head layers (acting as a feature extractor) and tail layers (serving as a classifier). The feature extractor runs on an edge device, while the classifier runs on the cloud. Apart from balancing overall computation cost and efficiency, split computing also offers privacy benefits. Communication between the edge and the cloud in split computing is inherently synchronized, as both devices contribute to completing one inference run.

Another notable paradigm for hybrid computing in machine learning is federated learning, which leverages multiple computing devices for training machine learning models for safety or efficiency purposes (Bonawitz et al., 2019). However, this technique is less commonly used for inference. Cloud service providers such as AWS also have developed patterns for hosting machine learning pipelines across local and cloud devices (AWS-Whitepaper, 2021). However, the design usually involves splitting the components of an entire machine learning pipeline, with the core machine learning models still hosted in the cloud.

In addition to hybrid computing, there is also literature on improving the efficiency of models deployed on edge devices (Tambe et al., 2021) as well as methods focused on reducing the size of large models for deployment on smaller devices (Hoeffler et al., 2021). However these methods are not directly comparable to our work.

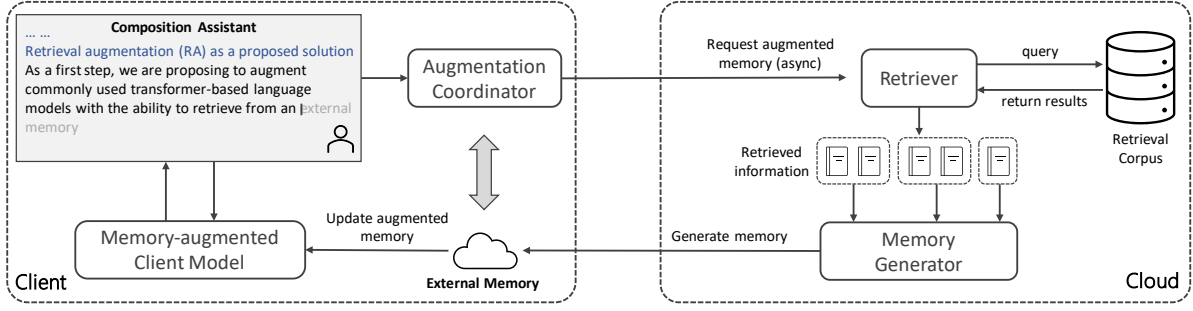


Figure 1: Overview of our HybridRAG Framework

3 Model

We present our HybridRAG approach that utilizes cloud-generated memory to enhance the utility of client-based language model. First, we provide an overview of our approach in Section 3.1. Then we delve into the details of cloud-based memory generation in Section 3.2 and Section 3.3. Furthermore, we describe the client model in Section 3.4.

3.1 Overall Architecture

The HybridRAG framework consists of four main components: an augmentation coordinator (client), a memory-augmented client model (client), a retriever model (cloud), a memory generator (cloud). Figure 1 illustrates the model architecture. The augmentation coordinator monitors the writing context and determines when to request an augmented memory from the cloud. The retriever model on the cloud server then searches the retrieval corpus to find relevant data. Subsequently, the memory generator employs the GPT3 model to construct an augmented memory that includes all essential information from the retrieved data, optimizing its usefulness. Finally, the augmented memory is transmitted to the client and seamlessly integrated into the client model, thereby enhancing its overall performance.

3.2 Augmentation Coordinator

The augmentation coordinator component is responsible for managing the augmented memory \mathcal{M} by monitoring changes to the writing context. The entire process of the augmentation coordinator is depicted in Figure 2. To determine whether a memory update is necessary, the coordinator takes into account both the current context c_t and the context c_{t-1} from the previous step and calculates the context edit distance $ED(c_t, c_{t-1})$. Once the distance exceeds a pre-determined threshold τ , the coordinator initiates a request to the cloud server

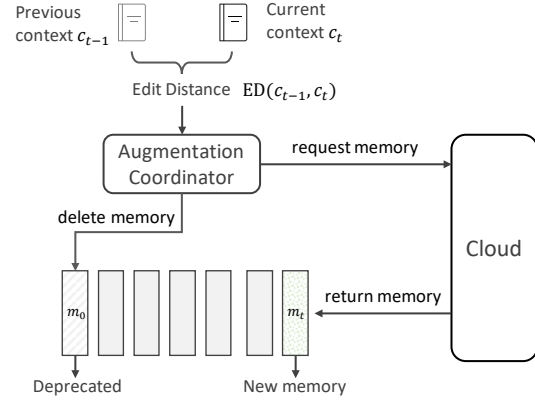


Figure 2: Process of the Augmentation Coordinator

for augmented memory. We employ the Levenshtein distance (Yujian and Bo, 2007) to measure the token-level distance. To avoid redundant memory requests, we adopt an incremental memory update approach, where only the newly updated context is used as the query input to generate the new memory m_t . When the augmented memory reaches its maximum capacity of M , the earliest stored memory is replaced with the new one. This process is depicted in Figure 2, where we observe that upon reaching the maximum memory capacity, the oldest memory m_0 is swapped out and replaced by the new memory m_t .

3.3 Retrieval-Augmented Memory Generator

Upon receiving a request from the augmentation coordinator, the memory generator on the cloud initiates the preparation of the augmented memory, which is then returned to the client. The memory preparation process consists of two steps: document retrieval and memory generation.

3.3.1 Document Retrieval

Given an input context x , the goal of the retriever is to select the top-k most relevant documents

Prompt	<p>Generate a list of key takeaways in complete sentences for each paragraph one by one, containing all the details such as entities, persons, organizations, numbers, years, and locations.</p> <p>P1: <i>Generative Pre-trained Transformer 3 (GPT-3) is an autoregressive language model released in 2020 that uses deep learning to produce human-like text. Given an initial text as prompt, it will produce text that continues the prompt.</i></p> <p>P2: <i>The architecture is a decoder-only transformer network with a 2048-token-long context and then-unprecedented size of 175 billion parameters, requiring 800GB to store. ... This eliminated the need for human supervision and for time-intensive hand-labeling.</i></p> <p>Key Takeaways: ### P1:</p>
Output	<p>- <i>Generative Pre-trained Transformer 3 (GPT-3) is a deep learning-based autoregressive language model released in 2020 that produces human-like text.</i></p> <p>- <i>It takes an initial text as prompt and produces text that continues the prompt.</i></p> <p>### P2:</p> <p>- <i>GPT-3 has a decoder-only transformer network with a 2048-token-long context and 175 billion parameters, requiring 800GB to store.</i></p> <p>- <i>It is trained to predict what the next token is based on previous tokens using generative pre-training. ...</i></p>

Table 1: Example of Augmented Memory Generated by GPT3 Model.

$\mathcal{D}_r = \{d_1, \dots, d_k\}$ from a large retrieval corpus \mathcal{D} , where $\mathcal{D}_r \subseteq \mathcal{D}$. Following prior work (Lewis et al., 2020; Ni et al., 2021), we use the Dense Passage Retrieval (DPR) (Karpukhin et al., 2020) method, which is based on a dual encoder architecture pre-trained for question and answering task. DPR encodes the document and the query with a context encoder and a query encoder respectively, and calculates the cosine similarity of the encoded embeddings to retrieve the top-k most relevant documents. We use the FAISS library (Johnson et al., 2019) for efficient retrieval.

3.3.2 Memory Generation

After retrieving the relevant documents \mathcal{D}_r , instead of directly using them as in Lewis et al. (2020), we employ a LLM such as GPT3/3.5 to generate concise bullet points that capture the essential information from the retrieved documents. These bullet points serve as the augmented memory, significantly reducing its size by including only the key takeaways instead of the full documents. This reduces both the communication cost to the client and the inference cost of the client model.

To generate summary bullet points from a retrieved document $d \in \mathcal{D}_r$, we first split the whole document into paragraphs $d = \{p_1, \dots, p_l\}$, where l is the number of paragraphs. We choose an appropriate paragraph size that maintains sentence integrity, avoiding breaking sentences in the middle. Once the paragraphs are created, we utilize the GPT3 Davinci model to extract the key takeaways from each paragraph p_i , as outlined in Table 1. To minimize the frequency of GPT3 call requests, we consolidate multiple paragraphs within a document. Table 1 illustrates an example where bullet points are generated for two paragraphs. Subsequently, all the generated bullet points from the retrieval docu-

ments are merged to form the memory m_t for the current t -th memory request. This memory is then combined with the existing memory to construct the new \mathcal{M} by the augmentation coordinator, as elucidated in Section 3.2.

3.4 Memory-Augmented Client Model

While most previous work on memory-augmented networks (Weston et al., 2014; Sukhbaatar et al., 2015; Yogatama et al., 2021; Wu et al., 2022) focus on language modeling task performance, our client model aims to effectively leverage the augmented memory generated by the LLM. We hypothesize that although a small language model may lack the capacity to handle complex tasks like its larger counterpart, it can still be trained effectively to leverage augmented memory to accomplish simpler tasks, such as paraphrasing content. To this end, we propose an instruction-finetuning approach that aims to bolster the client model’s ability to effectively leverage augmented memory.

Training such a client model is a challenging task due to the scarcity of training data. Specifically, it requires data that includes an input prompt \mathcal{I} , an augmented memory \mathcal{M} , and a ground truth text for completion \mathcal{C} . However, obtaining the latter two can be difficult. To address this, we propose a novel method that leverages a LLM to generate the necessary training data.

Given a document d , our initial step involves randomly selecting a percentage of the document (ranging from 12.5% to 50%) to serve as the input prompt $\mathcal{I}(d)$. Subsequently, we generate the augmented memory $\mathcal{M}(d)$ with the steps outlined in Section 3.3. As for the text completion labels, an alternative approach is to directly utilize the remaining portion of the document d , excluding the prompt. However, this method is suboptimal

Prompt	<i>Reference:</i> In 2020, Generative Pre-trained Transformer 3 (GPT-3) was unveiled, a deep learning-based autoregressive language model that can produce human-like text. When provided an initial text as a prompt, it can then generate text that follows on from it. ... This process has eliminated the need for laborious manual labeling and human supervision. <i>Complete the following paragraph based on the above reference.</i> Generative Pre-trained Transformer 3 (GPT-3) is an autoregressive language model released in 2020 that
Output	is capable of producing human-like text when prompted with an initial text. GPT-3 has a 2048-token-long context, a record-breaking 175 billion parameters and a storage capacity of 800GB.

Table 2: Example of Constructing an Instruction-enhanced Prompt for Pseudo Label Generation

since the original text may not encompass the information contained in the augmented memory. The discrepancy between the completion and the augmented memory used as a reference can negatively impact the performance of the client model.

To address this issue, we employ the GPT3 Davinci model to generate the text completion labels. We structure the input prompt and augmented memory into an instruction-based prompt following the format specified in Table 2. This enables us to instruct the GPT3 model to complete the input text based on the provided reference memory. The completion label can be expressed as $\mathcal{C}(d) = \mathcal{G}_c(\mathcal{I}(d); \mathcal{M}(d))$, with $\mathcal{G}_c(\cdot)$ referring to the GPT3-generated completion model. Additional details can be found in Appendix D, where we present empirical evidence demonstrating that GPT3-completed labels outperform ground truth labels from the original text.

After preparing the training data, we proceed to finetune our client model using the instruction-enhanced prompt along with the GPT3-completed labels, as demonstrated in Table 2 (Output). By incorporating the instruction-tuning method, the client model learns to effectively utilize the augmented memory generated by the GPT3 model. To minimize the discrepancy between our model’s predictions and the GPT3-completed pseudo labels, we employ the cross-entropy loss, as defined in Equation 1.

$$\mathcal{L}_d = - \sum_{i=1}^l \mathcal{G}_c(\mathcal{I}(d); \mathcal{M})_i \log \left(\mathcal{H}(\mathcal{I}(d); \mathcal{M})_i \right), \quad (1)$$

where l is the length of completion label and $\mathcal{H}(\cdot)$ refers to the probability of tokens generated by our client model.

4 Experiments

In this section, we present the evaluation results of our proposed HybridRAG approach on multiple benchmark datasets. We introduce the experiment setup in Section 4.1 and detail the performance

of the proposed method compared against several baseline models in terms of both composition assistance utility and inference latency in Section 4.2. Furthermore, we present a case study in Section 4.3. Additionally, we provide results from further experiments examining various components of our model in the appendix.

4.1 Experimental Setup

4.1.1 Datasets and Labels

We use the WikiText-103 dataset² and four datasets from the Pile benchmark (Gao et al., 2020) for our model training and evaluation. For instruction-tuning the client model, we utilize the training set of WikiText-103, which consists of 15,220 wiki pages. Specifically, we employ the abstract section of each page for text completion and use the remaining sections of the wiki pages as retrieval texts for memory generation, as outlined in Section 3.4.

For evaluation, we use the WikiText-103 test set and four subsets from the Pile benchmark: Enron Emails, HackerNews, NIH ExPorter, and Youtube Subtitles. These datasets encompass a diverse range of domains, including news, emails, medical documents and video subtitles. We compare the predictions of our models against the GPT3 completion label.

4.1.2 Implementation Details

For our client model, we select two small OPT language models (Zhang et al., 2022) with 125 million (OPT-125M) and 350 million model parameters (OPT-350M) for text prediction. These language models are pre-trained to predict text and have a small enough size to be well-suited for real-time composition assistance. We use greedy decoding for generation.

During the training phase, these models are trained on machines equipped with one Tesla V100 GPU with 16GB memory. For latency evaluation, we deploy our client model on two different machines: a GPU machine equipped with an 11GB

²<https://huggingface.co/datasets/wikitext>

Nvidia Tesla K80 GPU, and a laptop without a GPU, specifically a Surface 4 laptop featuring an Intel i7 CPU @3.00GHz with 4 cores and 16GB of physical memory. For bullet point generation, we utilize the GPT3 Davinci model in the OpenAI API with $temperature = 0$, $top_p = 1$.

4.1.3 Evaluation Metrics

We employ various automated metrics for evaluating the model performance. First, we use Perplexity³, a measure of the language model’s level of surprise or uncertainty when processing given text. Lower Perplexity values indicate higher model confidence and better comprehension of the input text. In addition, we present the results of lexical and semantic similarity metrics, including GLEU (Wu et al., 2016), BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), METEOR (Banerjee and Lavie, 2005), and BERTScore (Zhang* et al., 2020), to evaluate the degree of similarity between the model’s predictions and the provided labels.

To evaluate the inference latency of our system, we measure the running time required for three steps in our task: document retrieval, memory generation, and text prediction. This allows us to quantify the time cost associated with each of these steps and analyze the overall efficiency of our system.

4.1.4 Baseline Methods

We compare our approach against the following baselines: (i) *Vanilla OPT*: We employ a vanilla client OPT model for text completion, which does not use any additional memory. (ii) *RAG*: The Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) approach can be easily turned into a hybrid model with our framework. In this setting, we use the DPR model to retrieve relevant data from the cloud and feed the full retrieved text to the client model for generation. (iii) *HybridRAG without finetuning (HybridRAG w/o FT)*: To assess the efficacy of our instruction-tuned client model, we examine a HybridRAG model without applying finetuning to the client model for text prediction. (iv) *GPT3 zero-shot*: We use the GPT3 Davinci model in a zero-shot manner for text completion. However, it’s important to note that the GPT3 model cannot be deployed on client devices for real-time composition assistance.

When evaluating the baseline models, we ensure a fair comparison by regenerating reference labels using the GPT3 model, based on the memory used

by each baseline. Specifically, for the Vanilla OPT baseline, reference labels are generated with GPT3 without additional memory. For RAG, reference labels are generated by GPT3 with full text. In the case of GPT3-zero-shot baseline, since there is no ideal reference label for comparison, we used the same label as our HybridRAG approach.

4.2 Experimental Results

4.2.1 Utility

Table 3 presents the performance comparison between our models and the baselines on the Wikitext-103 dataset, using the OPT-125M and OPT-350M models. The results demonstrate that our approach outperforms all client/hybrid baselines across all evaluated metrics. Compared to the vanilla client models, our approach exhibited remarkable performance improvements. On average, the HybridRAG approach achieves an improvement of 67.6% in Perplexity and 171.7% in GLEU when comparing OPT-125M and OPT-350M. In comparison to the RAG approach (where we use the top 3 documents for memory augmentation due to prompt size limitations), our model still achieved substantial performance improvement of 50.4% on Perplexity and 113.1% on GLEU. Furthermore, when comparing our full approach to the variant without finetuning, our model outperformed it by 4.7% and 9.5% in the respective metrics. Comparing RAG and HybridRAG w/o FT, we can see that the utilization of GPT3-compressed memory results in a significant average performance gain of 48.6% in GLEU. Finally, the zero-shot performance from the GPT Davinci model showcased impressive generation capability without any finetuning or additional context, surpassing the OPT-125M model albeit still falling short of the OPT-350M model.

The results obtained from the OPT-350M model on the four Pile datasets are presented in Table 4. Consistent with the findings on the Wikitext-103 dataset, our model demonstrates superior performance compared to the baseline models across all four datasets. It is important to note that we did not finetune the client model specifically on the Pile datasets, further highlighting the model’s generalization capabilities. Lastly, we have also observed a significantly high perplexity when using the GPT3 Davinci model. This is due to substantial variance in probability between the model’s predictions (generated by GPT3 zero-shot) and the ground truth labels (generated by GPT3 with aug-

³<https://en.wikipedia.org/wiki/Perplexity>

mented memory).

4.2.2 Inference Latency

We performed a latency evaluation for both the OPT-125M and OPT-350M models on the two hardware setups, as described in Section 4.1.2. Figure 5a illustrates that the OPT-125M model exhibits a 49.3% faster inference time compared to the OPT-350M model. This finding emphasizes that the size of the client model plays a crucial role in the inference time. Figure 5b presents the running time for the retrieval and memory generation steps. The results indicate that memory generation utilizing a large language model consumes the majority of the memory preparation time. Figure 3c compares synchronous inference with retrieval augmentation using the GPT3 model and our asynchronous HybridRAG approach. Notably, our approach showcases an impressive speed enhancement, achieving a remarkable 138.3 times faster performance compared to the synchronous approach. Lastly, we conducted a comparison of the running time between the GPU machine and the laptop in Figure 3d. The results indicate that our approach can be deployed on user edge devices without GPUs, although the inference time is approximately 1.45 times slower compared to a GPU machine. It should be noted that we didn’t optimize the client model for decoding speed with established methods such as caching and quantization. These methods are orthogonal to our work and could be used in conjunction with our approach to further reduce the inference latency.

4.2.3 Asynchronous Memory Update

Figure 4 illustrates the impact of asynchronous memory update on model performance. The results indicate that as the edit distance increases, the memory becomes less up-to-date, resulting in a decline in utility. Notably, when the edit distance threshold is raised from 5 to 10, there is only a slight decrease of 2.9% in the GLEU score. However, as the threshold reaches 20, we observe a notable drop of 13.4% in the GLEU score. More detailed information can be found in Appendix A. These results imply that that setting the edit distance threshold below 10 yields the optimal utility for asynchronous memory update.

4.3 Case Study

To better understand the strengths and limitations of HybridRAG, we manually examined the completions of different models: GPT3 zero-shot, GPT3-

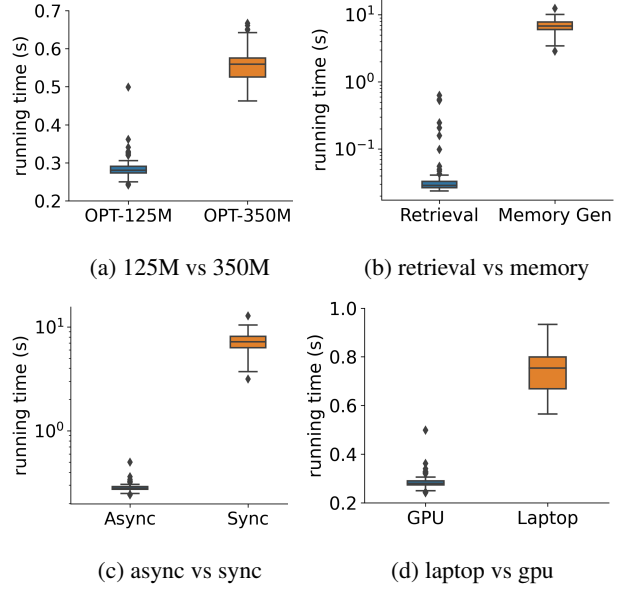


Figure 3: Inference Latency for client inference time, retrieval time and memory generation time on multiple devices

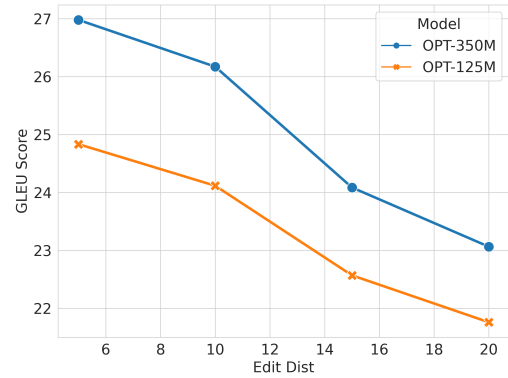


Figure 4: Utility on Asynchronous Memory Update

generated ground truth, vanilla OPT-350M and HybridRAG models with OPT-125M and OPT-350M.

We evaluated the completions of the models based on several criteria: factualness, repetitions and fluency. We manually annotated the completions of the models with these criteria, using a binary score for each criterion. Tables 5 shows an example of completions of the prompt by different models that illustrates the main observations and learnings. If we focus on the restricted task of completing the sentence, both OPT-125M and OPT-350M HybridRAG models are able to generate fluent, factual texts. When looking at longer completions, the OPT-125M HybridRAG model seems to perform better than the OPT-350M counterpart, despite having a smaller size. This might

		PPL	GLEU	BLEU-4	ROUGE-1	ROUGE-L	METEOR	BERTScore
	GPT3 zero-shot	4.9	32.4	31.3	50.0	44.3	43.6	89.4
OPT-125M	Vanilla OPT	10.3	12.5	8.6	28.9	23.7	22.0	84.2
	RAG	6.7	15.5	12.6	32.4	27.0	25.9	84.9
	HybridRAG w/o FT	3.3	31.1	31.1	45.8	40.7	40.5	87.7
	HybridRAG	3.3	34.1	33.9	48.4	42.9	43.4	88.4
OPT-350M	Vanilla OPT	8.1	14.0	10.2	31.3	25.5	24.3	84.9
	RAG	5.3	18.3	15.8	35.5	29.4	29.1	85.8
	HybridRAG w/o FT	2.9	34.6	35.0	49.0	43.8	44.0	88.5
	HybridRAG	2.7	37.8	37.8	52.4	47.2	47.5	89.3

Table 3: Utility Performance on the Wikitext-103 dataset

	Enron Emails		NIH ExPorter		Hacker News		Youtube Subtitles	
	Perplexity	GLEU	Perplexity	GLEU	Perplexity	GLEU	Perplexity	GLEU
GPT3 zero-shot	106.9	12.3	12.2	18.5	65.1	15.3	36.6	13.7
Vanilla OPT	5.4	8.7	6.3	13.1	4.8	12.7	5.5	9.7
	RAG	3.4	13.6	3.9	17.9	3.3	13.0	3.8
	HybridRAG w/o FT	2.9	19.9	3.3	22.1	2.5	23.6	2.8
	HybridRAG	2.9	23.6	3.0	24.0	2.7	25.7	21.7

Table 4: Utility Performance on Pile dataset

prompt	Graham Arthur Chapman (8 January 1941 - 4 October 1989
GPT3 zero-shot) was an English cricketer. Chapman was a right-handed batsman who bowled right-arm medium pace. He was born at Leicester, Leicestershire. Chapman made his first
GPT3-generated label) was an English comedian, writer, actor, author, and one of the six members of the British surreal comedy group Monty Python. Chapman and Cleeve went on to collaborate on different projects after the success of Mont
HybridRAG OPT-125M) was a British actor and comedian. He was originally cast in the Red Dwarf episode "Timeslides ". He met his long term partner, David Sherlock in 1966. Chapman and Sherlock moved to Belsize Park in
Vanilla OPT-350M) was an English cricketer. He played first-class cricket for Surrey in the 1960s and 1970s. References External links
HybridRAG OPT-350M) was a British comedian and actor. He was born in London, the son of a British diplomat. He was educated at Emmanuel College, Cambridge, where he studied medicine. He also joined the Cambridge Footlights, where

Table 5: Completions of the prompt by different models, showcasing a working case for HybridRAG OPT-125M; this is also a working case for HybridRAG OPT-350M if we only consider the first sentence.

indicate that the HybridRAG OPT-125M has a better balance between the retrieval and generation mechanisms.

However, we also observe cases where the HybridRAG model fails (refer to Appendix F). Improving the memory generator by reducing duplicate information, and enhancing the reasoning abilities of the client model or encouraging it to stick to the memories content would be some of the ways to address these failing cases and limitations.

5 Conclusion

In this paper, we propose HybridRAG, a novel hybrid retrieval-augmented generation approach for real-time composition assistance. By integrating

LLM-enhanced memory into our instruction-tuned client model with asynchronous update, we show with experiment results on multiple datasets that our hybrid retrieval approach enables substantial utility improvements over smaller language models while maintaining inference efficiency, making it a valuable solution for real-time tasks.

In our work, we employ retrieval-based memory augmentation as the solution to combine the powerful LLM on the cloud and the more agile client model. Naturally, the performance of the system relies on the quality of the memory that the cloud provides to the client and how the memory is integrated into the client model. The quality of the memory is influenced by multiple factors: the rep-

resentation of the memory (e.g. original text chunk vs condensed information snippets), the relevance of the retrieved data, and the freshness of information compared to the current input context. In future work, we will continue to investigate more effective ways of representing memory according to the tasks, and explore alternative memory augmentation approaches, such as RETRO (Borgeaud et al., 2021) and Knowledge Infused Decoding (Liu et al., 2022), to further enhance our model performance.

References

- AWS-Whitepaper. 2021. [Hybrid machine learning](#).
- Satanjeev Banerjee and Alon Lavie. 2005. [METEOR: An automatic metric for MT evaluation with improved correlation with human judgments](#). In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.
- Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, et al. 2019. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1:374–388.
- Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. 2021. [Improving language models by retrieving from trillions of tokens](#). *CoRR*, abs/2112.04426.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.
- Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.
- Torsten Hoeffler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. 2021. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *J. Mach. Learn. Res.*, 22(1).
- Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.
- Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.
- Chin-Yew Lin. 2004. [ROUGE: A package for automatic evaluation of summaries](#). In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.
- Ruibo Liu, Guoqing Zheng, Shashank Gupta, Radhika Gaonkar, Chongyang Gao, Soroush Vosoughi, Milad Shokouhi, and Ahmed Hassan Awadallah. 2022. Knowledge infused decoding. *arXiv preprint arXiv:2204.03084*.
- Dumitrel Loghin, Lavanya Ramapantulu, and Yong Meng Teo. 2019. [Towards analyzing the performance of hybrid edge-cloud processing](#). In *2019 IEEE International Conference on Edge Computing (EDGE)*, pages 87–94.
- Yoshitomo Matsubara, Marco Levorato, and Francesco Restuccia. 2022. [Split computing and early exiting for deep learning applications: Survey and research challenges](#). *ACM Comput. Surv.*, 55(5).
- Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y Zhao, Yi Luan, Keith B Hall, Ming-Wei Chang, et al. 2021. Large dual encoders are generalizable retrievers. *arXiv preprint arXiv:2112.07899*.
- Seyed Ali Osia, Ali Shahin Shamsabadi, Sina Sajadmanesh, Ali Taheri, Kleomenis Katevas, Hamid R. Rabiee, Nicholas D. Lane, and Hamed Haddadi. 2020. [A hybrid deep learning architecture for privacy-preserving mobile analytics](#). *IEEE Internet of Things Journal*, 7(5):4505–4518.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. pages 311–318.
- Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. *Advances in neural information processing systems*, 28.
- Thierry Tamba, Coleman Hooper, Lillian Pentecost, Tianyu Jia, En-Yu Yang, Marco Donato, Victor Sanh, Paul Whatmough, Alexander M. Rush, David Brooks,

- and Gu-Yeon Wei. 2021. [Edgebert: Sentence-level energy optimizations for latency-aware multi-task nlp inference](#). In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '21, page 830–844, New York, NY, USA. Association for Computing Machinery.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.
- Bo Wang, Changhai Wang, Wanwei Huang, Ying Song, and Xiaoyun Qin. 2020. [A survey and taxonomy on task offloading for edge-cloud computing](#). *IEEE Access*, 8:186080–186101.
- Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.
- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. [Google’s neural machine translation system: Bridging the gap between human and machine translation](#).
- Yuhuai Wu, Markus N Rabe, DeLesley Hutchins, and Christian Szegedy. 2022. Memorizing transformers. *arXiv preprint arXiv:2203.08913*.
- Dani Yogatama, Cyprien de Masson d’Autume, and Lingpeng Kong. 2021. Adaptive semiparametric language models. *Transactions of the Association for Computational Linguistics*, 9:362–373.
- Li Yujian and Liu Bo. 2007. A normalized levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1091–1095.
- Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.
- Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. [Bertscore: Evaluating text generation with bert](#). In *International Conference on Learning Representations*.

A Asynchronous Memory Update

Figure 5 shows the changes in GLEU score and perplexity as the edit distance increases in asynchronous memory update.

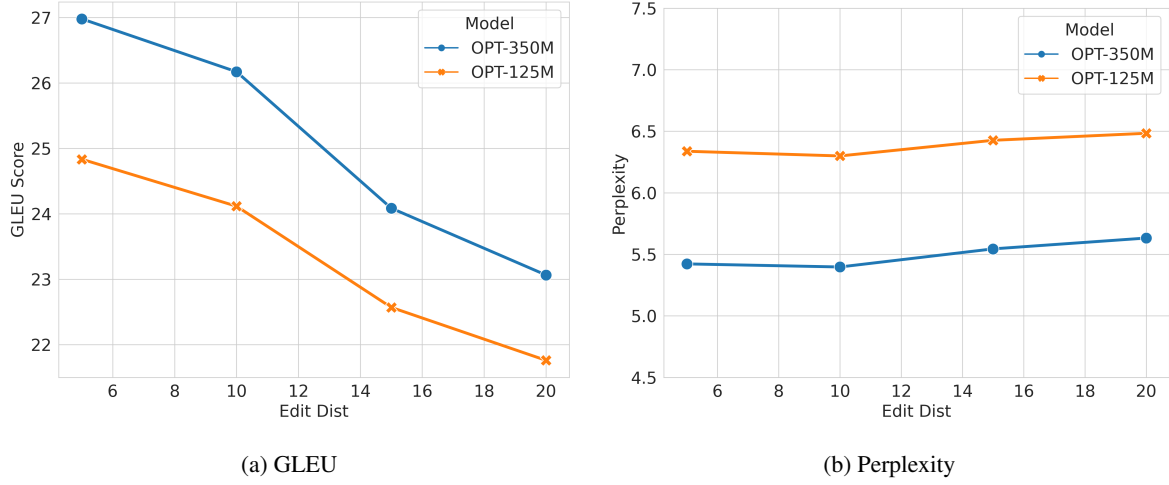


Figure 5: Changes in GLEU and Perplexity with frequency in Asynchronous Memory Update

B Local vs. Global Retrieval Corpus

Figure 6 displays a performance comparison between local and global retrieval corpus on the Wikitext-103 dataset. In the local setting, the retrieval corpus consists of the texts contained within the same document, while in the global setting, the entire dataset is utilized as the retrieval corpus. The results indicate that the local retrieval corpus yields superior GLEU and Perplexity scores. This can be attributed to the smaller pool of documents in the local corpus, making it easier to obtain the relevant documents for retrieval.

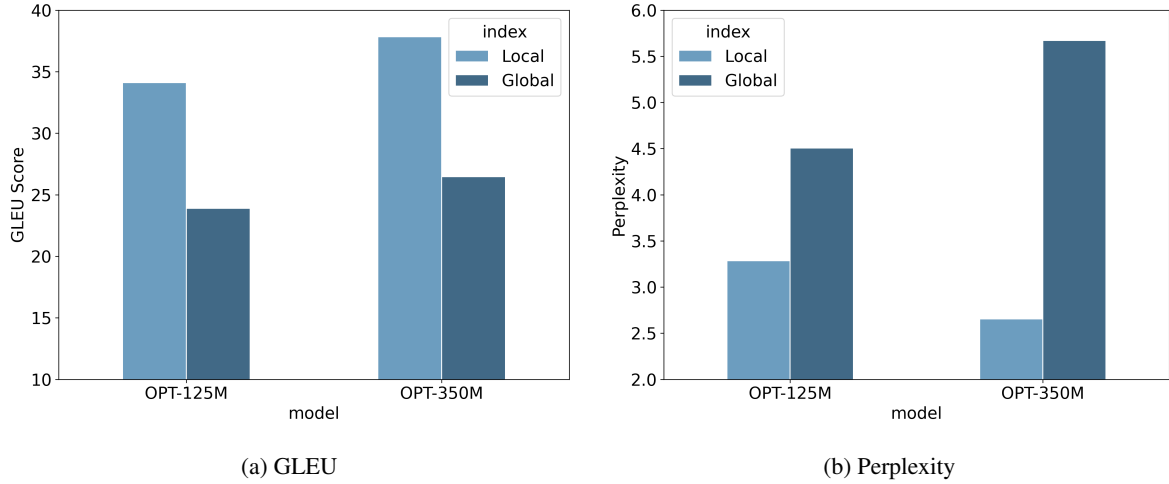


Figure 6: Comparison between local and global retrieval corpus

C Number of Retrieval Documents

Figure 7 depicts the model performance for various numbers of retrieval documents on Wikitext-103 dataset. Based on the results, we can deduce that both the OPT-125M and OPT-350M models exhibit optimal performance when four retrieval documents are used. As more documents are included beyond four, the performance remains consistent as the top four documents already encompass the majority of relevant information for our task.

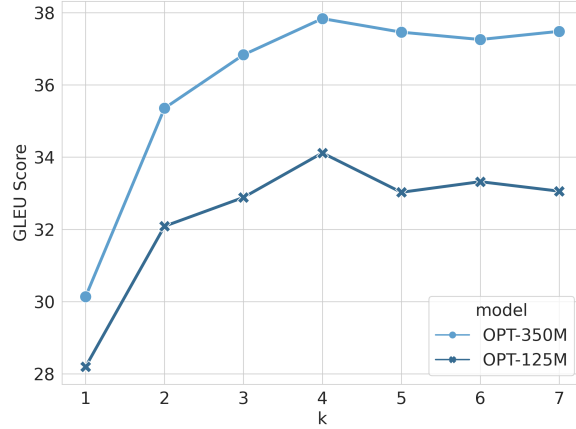


Figure 7: Performance analysis on number of retrieval documents

D GPT Completion Labels

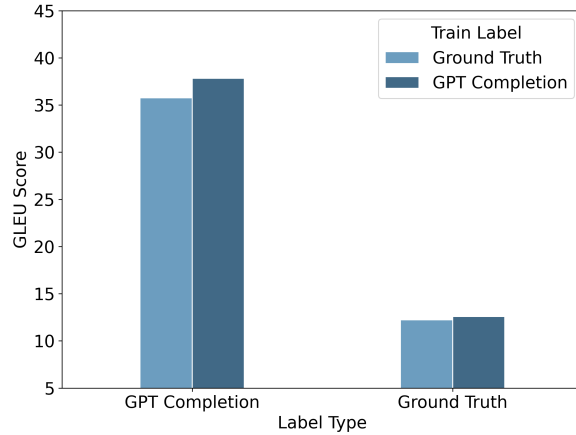


Figure 8: Comparison between ground truth labels derived from GPT3 completion and the original text

We investigate the discrepancy between using ground-truth labels from GPT3 completion and the original text for both model training and evaluation. As shown in Figure 8, fine-tuning our client model on GPT3 completion labels consistently yields consistent better results. Notably, even when evaluating with the original text as ground truth, the model fine-tuned with GPT3-generated labels outperforms the alternative.

E More experiments on Pile datasets

The results of the 5 Pile datasets on all seven metrics are presented in Tables 6 and 7. We can observe that our model consistently outperforms all the other baselines, demonstrating the superior performance. However, it is worth noting that the perplexity of the OPT-125M model does not consistently surpass that of the HybridRAG w/o FT model.

F More examples from human evaluation

Table 8 shows another working example for HybridRAG models and Table 9 shows an example of a failing case for both OPT-125M and OPT-350M HybridRAG models. The OPT-125M model generates a repetitive and factually incorrect text, while the OPT-350M model generates a text that is factually incorrect. GPT3 Davinci, however, can still use the same retrieved memories to provide a factual and useful completion for this prompt. The memories are bullet points generated from several document chunks;

		PPL	GLEU	BLEU-4	ROUGE-1	ROUGE-L	METEOR	BERTScore
Enron Emails	GPT3 zero-shot	106.9	12.3	10.4	26.1	23.3	21.6	83.6
	Vanilla OPT	5.4	8.7	6.5	21.1	18.6	17.1	80.2
	RAG	3.4	13.6	12.2	26.8	24.0	22.9	81.2
	HybridRAG w/o FT	2.9	19.9	19.4	32.7	30.2	28.8	83.0
	HybridRAG	2.9	23.6	23.7	35.0	32.3	31.9	84.7
NIH ExPorter	GPT3 zero-shot	12.2	18.5	16.2	36.6	31.7	29.2	86.7
	Vanilla OPT	6.3	13.1	8.9	31.9	26.7	23.9	85.3
	RAG	3.9	17.9	15.2	36.7	31.1	29.7	86.5
	HybridRAG w/o FT	3.3	22.1	20.2	40.0	35.1	33.5	87.1
	HybridRAG	3.0	24.0	22.9	41.5	37.1	35.4	87.1
Hacker News	GPT3 zero-shot	65.1	15.3	14.3	30.2	27.7	20.4	85.8
	Vanilla OPT	4.8	12.7	11.3	29.3	26.4	22.4	84.9
	RAG	3.3	13.0	12.2	26.8	24.5	20.5	84.3
	HybridRAG w/o FT	2.5	23.6	24.4	37.9	35.7	31.5	86.6
	HybridRAG	2.7	25.7	25.9	39.5	36.8	34.4	86.5
Youtube Subtitles	GPT3 zero-shot	36.6	13.7	11.8	27.1	24.5	23.3	84.4
	Vanilla OPT	5.5	9.7	7.2	22.2	20.0	19.1	82.4
	RAG	3.8	14.8	12.9	28.4	25.3	25.1	84.7
	HybridRAG w/o FT	2.8	20.1	19.9	32.9	30.4	29.7	85.1
	HybridRAG	2.8	21.7	21.1	34.6	31.8	31.9	85.9

Table 6: Utility Performance of OPT-350M Model on Pile datasets

		PPL	GLEU	BLEU-4	ROUGE-1	ROUGE-L	METEOR	BERTScore
Enron Emails	GPT3 zero-shot	106.9	12.3	10.4	26.1	23.3	21.6	83.6
	Vanilla OPT	6.0	10.5	9.1	21.5	19.3	18.0	80.3
	RAG	3.7	12.7	11.9	25.2	22.9	21.6	80.4
	HybridRAG w/o FT	3.2	20.3	19.9	31.0	28.6	27.4	82.7
	HybridRAG	3.7	18.9	18.9	31.6	28.3	28.2	83.8
NIH ExPorter	GPT3 zero-shot	12.2	18.5	16.2	36.6	31.7	29.2	86.7
	Vanilla OPT	5.4	12.0	10.8	27.9	25.3	21.4	84.3
	RAG	3.8	11.5	10.5	25.4	23.3	18.6	83.8
	HybridRAG w/o FT	2.9	19.5	19.8	33.5	31.4	27.2	85.6
	HybridRAG	3.3	23.0	23.5	36.2	33.2	30.8	85.9
Hacker News	GPT3 zero-shot	65.1	15.3	14.3	30.2	27.7	20.4	85.8
	Vanilla OPT	7.8	11.8	7.5	29.7	25.2	22.2	84.7
	RAG	4.7	15.8	12.8	34.1	29.1	27.2	85.8
	HybridRAG w/o FT	3.8	19.5	17.5	37.3	32.6	30.8	86.4
	HybridRAG	4.0	21.7	20.5	38.0	32.8	32.2	86.5
Youtube Subtitles	GPT3 zero-shot	36.6	13.7	11.8	27.1	24.5	23.3	84.4
	Vanilla OPT	6.4	9.4	6.9	23.2	20.7	19.6	82.9
	RAG	4.5	12.2	9.6	26.1	23.2	23.4	84.1
	HybridRAG w/o FT	3.1	15.9	15.0	28.5	26.1	25.1	84.2
	HybridRAG	3.5	17.1	16.2	29.4	26.5	25.7	84.7

Table 7: Utility Performance of OPT-125M Model on Pile datasets

OPT-125M and OPT-350M limited reasoning abilities does not allow them to process the memories extensively and reorganize them. The working cases correspond to situations where the client model only has to paste an existing chunk of text as is; we’ve noticed that when the small OPT client models combine information from different bits of the memories, they usually end up fabricating wrong facts. On top of this, the completions from the current finetuned OPT-350M model seem to put more emphasis on the (limited) parametric memory than on the retrieved memories, leading to more "creative" – but factually wrong – generated text.

Improving the memory generator by reducing duplicate information, and enhancing the reasoning abilities of the client model or encouraging it to stick to the memories content would be some of the ways to address these failing cases and limitations.

prompt	Gears of War 2 is a 2008 military science fiction third-person shooter video
GPT3 zero-shot	game developed by Epic Games and published by Microsoft Game Studios for the Xbox 360. It is the second installment of the Gears of War series and the sequel to 2006's Gears of War. The game was released worldwide in
GPT3-generated label	game. The title and spoken words of the trailer are taken from the World War I poem "I Have a Rendezvous with Death" by Alan Seeger. In Gears of War 2, players are able to carry a
HybridRAG OPT-125M	game. The title and spoken words of the trailer were taken from Alan Seeger's World War I poem "I Have a Rendezvous with Death". On October 10, 2008, Microsoft and Epic Games announced the game had
Vanilla OPT-350M	game developed by Electronic Arts and published by Electronic Arts for the PlayStation 2 and Xbox 360. The game was released on November 26, 2008 for the PlayStation 2 and Xbox 360. Gameplay Gears of War
HybridRAG OPT-350M	game developed and published by Epic Games and Microsoft. The game is set 18 months after the events of Gears of War 2. The game is set in the fictional world of the United States of America. The player controls a

Table 8: Completions of the prompt by different models, showcasing a working case for OPT-125M HybridRAG; this is also a working case for OPT-350M HybridRAG if we only consider the first 2 sentences.

prompt	The Bengali language movement (Bengali: Bhasha Andolon), was a political movement in former East Bengal (today Bangladesh) advocating the recognition of the Bengali language as an
GPT3 zero-shot	official language of Pakistan. The movement was a major part of the Bengali Language Movement and led to the establishment of Bengali as one of the state languages of then East Pakistan in 1956. The movement was
GPT3-generated label	official language of the then-Dominion of Pakistan. That sparked many similar movements in other parts of the country, especially in East Pakistan, where many people felt subjugated by the central government dominated by West Pakistani elite.
HybridRAG OPT-125M	official language. The movement was founded by the Bengali community in the early 20th century. The movement was founded by the Bengali community in the early 20th century. The movement was founded by the Bengali community
Vanilla OPT-350M	official language of the state. The movement was founded by the Bengali-language activist and writer, Bhasha Andolôn. The movement was founded in the year 2000. The movement was led by the
HybridRAG OPT-350M	official language of Bangladesh. The movement was founded in the early 1990s by Fazlur Rahman, a Bengali-language journalist and activist. The movement was founded in the early 1990s by Fazlur

Table 9: Completions of the prompt by different models corresponding to failing cases for OPT-125M HybridRAG and OPT-350M HybridRAG.