Parallel Key-Value Cache Fusion for Position Invariant RAG

Anonymous ACL submission

Abstract

Recent advancements in Large Language Models (LLMs) underscore the necessity of Retrieval Augmented Generation (RAG) to leverage external information. However, LLMs are sensitive to the position of relevant information within contexts and tend to generate incorrect responses when such information is placed in the middle, known as 'Lost in the Middle' phenomenon. In this paper, we introduce a framework that generates consistent outputs for decoder-only models, irrespective of the input context order. Experimental results for three open domain question answering tasks demonstrate position invariance, where the model is not sensitive to input context order, and superior robustness to irrelevent passages compared to prevailing approaches for RAG pipelines.

1 Introduction

In Retrieval Augmented Generation (RAG) (Guu et al., 2020; Lewis et al., 2021; Izacard et al., 2022), models first extract relevant information from a knowledge base and then incorporate this extracted information with its parameteric knowledge to generate the response. This two-step approach is the de-facto approach for knowledge-intensive tasks (Lewis et al., 2021; Petroni et al., 2021).

However, decoder-only models exhibit an intrinsic positional bias, assigning more attention to tokens at the beginning or end of the input sequence while often overlooking relevant context located in the middle, a problem known as the 'Lost in the Middle' (Liu et al., 2023). Previous works to address this issue involves training with specific prompt (He et al., 2024) or data-intensive training (An et al., 2024). Other works aimed at modifying positional embeddings (Hsieh et al., 2024b) or reducing positional attention bias in LLMs (Yu et al., 2024a). Yet, none of these methods fully guarantee a solution to this intrinsic bias in LLMs for RAG.

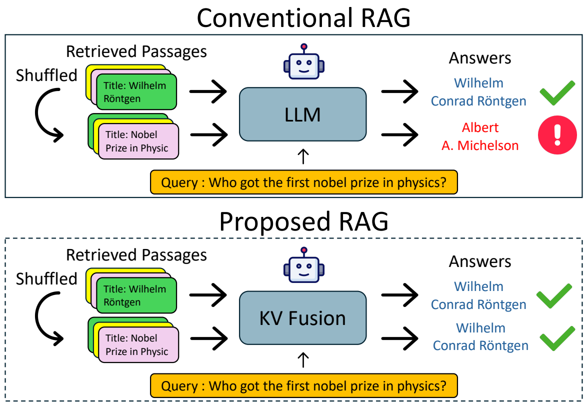

Figure 1: Illustration of the KV-Fusion model: Generated tokens remain consistent even when the retrieved passages are shuffled.

In this paper, we introduce a framework for decoder-only models, called **K**ey **V**alue **Fusion** (**KV Fusion**), to generate consistent outcomes regardless of input order as illustrated in Figure 1. **KV Fusion** consists of two components: a *prefill* decoder that extract key-values caches in parallel and a *trainable* decoder that utilizes extracted key value caches to produce consistent outcome. This architecture injects uniform positional information to each input contexts, ensuring consistent output generation even when the input order varies. Experiments on open domain question answering datasets, including NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and POPQA (Mallen et al., 2023), demonstrate KV-Fusion's position-invariant nature, achieving accuracy improvements of 21.4%, 6.4%, and 6.6% over baseline models in shuffled settings. Furthermore, KV-Fusion models exhibit robust and stable accuracies even with additional contexts compared to other approaches.

2 Method

Notation Our KV-Fusion architecture is illustrated in Figure 2. For clarity, we refer to this prefill decoder as $\mathcal{D}_{\mathrm{p}}$, which is characterized by the

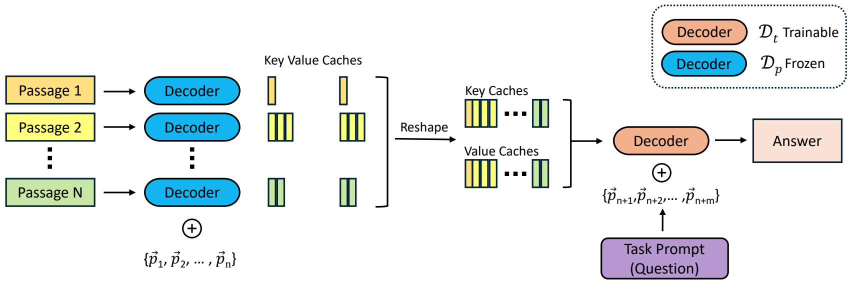

Figure 2: Overview of KV-Fusion Architecture. $\mathcal{D}_\text{p}$ denotes Prefill decoder and $\mathcal{D}_\text{t}$ represents Trainable decoder. We employ the off-the-shelf LLM to extract the key and value states of the retrieved contexts independently. Then reshaping these caches to train the LLM with task instructions along with questions to generate answers.

number of key and value heads $|H|$, each with a dimension of d_h. We denote the trainable decoder as $\mathcal{D}_\text{t}$, and represent the set of input passages as $\mathcal{C} = \{c_1, c_2, \dots, c_N\}$ with fixed token length n for each c_i. This set of passages represents smaller chunks of a long document or retrieved contexts. Lastly, let L represent the total number of layers in $\mathcal{D}_\text{p}$ and $\mathcal{D}_\text{t}$, and let l denote the lth layer.

Prefill Decoder ($\mathcal{D}_\text{p}$) extracts the KV cache from multiple input passages in parallel, resulting in the injection of identical local positional embeddings $\{\vec{p_1}, \vec{p_2}, \dots, \vec{p_n}\}$. The layer-wise cache representation for each c_i is as follows:

$$\{k_i^l, v_i^l\}_{l=1}^L = \mathcal{D}_\text{p}(c_i), \quad k_i^l, v_i^l \in \mathbb{R}^{|H| \times n \times d_h}$$

Next, we reshape layer-wise KV-caches by concatenating along the token axis over N contexts, forming a single cache for each layer l:

$$K^l = \text{RES}(\{k_i^l\}_{i=1}^N) \quad V^l = \text{RES}(\{v_i^l\}_{i=1}^N)$$

Here, $K^l, V^l \in \mathbb{R}^{|H| \times (N \times n) \times d_h}$ are reshaped KV-cache for the corresonding layer over input passages. These caches prefills and serve as grounding knowledge for training $\mathcal{D}_\text{t}$.

Trainable Decoder ($\mathcal{D}_\text{t}$) takes two inputs: (1) reshaped KV-caches ($\{K^l, V^l\}_{l=1}^L$) and (2) target tokens, which contain instruction queries, and answers with a length of m tokens. To ensure sequential alignment of positional information with the KV-caches, position information starting from $\vec{p}_{n+1}$ to $\vec{p}_{n+m}$ are assigned. We then train $\mathcal{D}_\text{t}$ using next-token prediction, conditioning on the reshaped KV-caches rather than previous tokens:

$$\mathcal{D}_\text{t}(y|q, \mathcal{C}') \triangleq \mathcal{D}_\text{t}(y|q, \{K^l, V^l\}_{l=1}^L)$$

Here, q denotes the instruction with query tokens and y is answer tokens. $\mathcal{C}'$ represents the set of input passages tokens, and $\{K^l, V^l\}_{l=1}^L$ is the reshaped KV-cache corresponding to $\mathcal{C}'$. We illustrate the details of KV-Fusion in Appendix A.1

3 Experiment Setup

3.1 Datasets

We consider three open domain question answering datasets: Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and POPQA (Mallen et al., 2023) .[1] For the base retrieval corpus, we utilize a December 2018 Wikipedia snapshot consisting of 21 million passages, following (Yen et al., 2024; Yu et al., 2024b). Lastly, we use the DPR (Karpukhin et al., 2020) as our baseline retriever to extract the top-40 passages for each dataset. [2]

Dataset Construction To enhance the robustness in RAG, we train models with irrelevant contexts (Fang et al., 2024; Yoran et al., 2024a). To this end, we draw the best gold context and extract key phrases among candidate passages by prompting gpt-4o API with a fine-grained template. If all responses are negative, the instance is discarded. Otherwise, we retain the extracted key phrases as evidence, which is later used for training. Negative contexts are sampled from DPR-retrieved passages that do not contain any answer. Each training instance consists of one gold context and 19 negative contexts. The prompt for this process and statistics of all datasets are desribed in Appendix A.2.

Metric and Evaluations Exact Match (EM) Accuracy is used for evaluation. (Asai et al., 2023; Mallen et al., 2023). However, we observe that as more documents are added to the input, baseline models tend to generate instrinsic knowledge or hallucinated responses (Hsieh et al., 2024a). To address this, we incorporate answerability into the prompt, requiring responses to be concise, and limited to a single sentence. Lastly, we set a 48-token

[1]Note that NQ and TriviaQA are filtered version from DPR (Karpukhin et al., 2020)

[2]Note that we used DPR trained from scratch with its own hard negatives on December 2018 Wikipedia snapshot

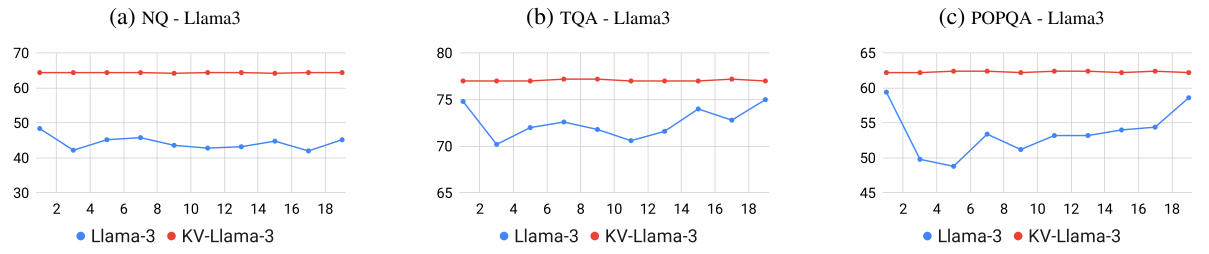

Figure 3: Comparison of EM Accuracy between KV-Llama3 and Llama3 across different gold context positions. KV-Llama3 maintains its accuracy, while Llama3 shows a tendency for the 'lost in the middle' problem.

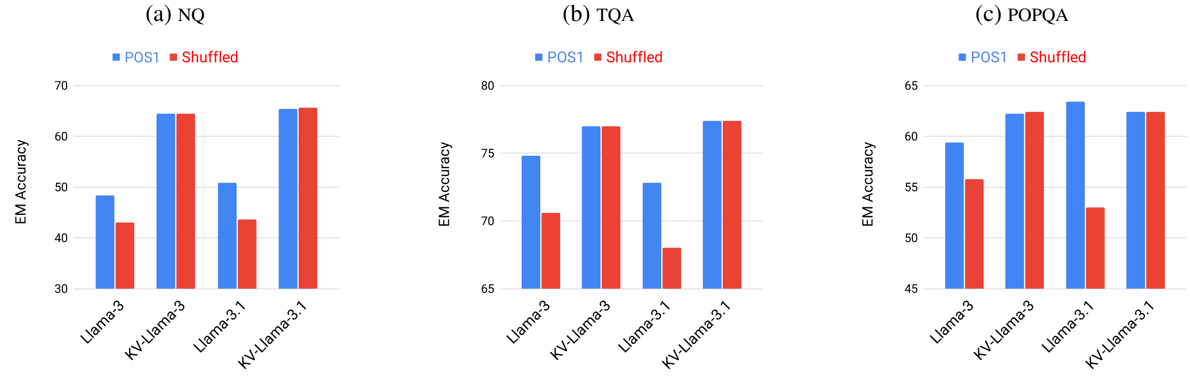

Figure 4: Accuracies of baseline and KV models in two scenarios: 1) POS1, where the gold context is positioned first, and 2) Shuffled, where contexts are randomly ordered. KV models maintain their accuracy on both cases, while baseline models struggle in shuffled setting, leading in a wider accuracy gap between the baseline and KV-models.

limit and use greedy-decoding (Huang et al., 2023a) for baselines and KV-Fusions. The template and example for baseline are illustrated in Appendix A.3.

3.2 Training

Input Formatting Each input passage is formatted with '`Title:`{title}' and '`Context:`{text}', followed by a document boundary, '`====`'. For target tokens, we preprend a signal token, <|question_answering|>, to guide the model's behavior during inference (Asai et al., 2023). Next, we append instruction and '`Question:`{question}'. Finally, we add answer tokens, which contain both answer string and a key phrase as evidence, as described in Section 3.1. We hypothesize that appending key phrases enhances the the model's robustness (Thoppilan et al., 2022; Menick et al., 2022). Format examples are provided in Appendix A.4.

Technical Details We initialize both $\mathcal{D}_\text{p}$ and $\mathcal{D}_\text{t}$ with the Llama3-8B model (Dubey et al., 2024). We fine-tune on each dataset with a maximum learning rate of 2×10^{-5} using the AdamW. Across all dataset, we use a batch size of 64 on four A100(80G) GPUs. For the NQ and TQA datasets, models are trained for 2 epochs. For the POPQA dataset, we fine-tune it on top of TQA fine-tuned model due to its small training size. The same procedure is applied to the Llama3.1-8B. Detail hyperparameters are reported in Appendix A.5.

4 Results

Position Invariant RAG To demonstrate the position-agnostic property, we test models with the gold context placed at varying positions. For each dev dataset, we construct 10 versions by inserting gold context at every alternate location (1st, 3rd, etc.), along with an additional dev set where all 20 contexts are randomly shuffled. To manage the increased inference time, we evaluate the first 500 instances. As shown in Figure 3, KV-Llama3 maintains consistent accuracy across all datasets, regardless of the position of the gold context, while conventional Llama3 shows varying accuracy. A similar pattern is observed with KV-Llama3.1 and Llama3.1 as shown in Appendix A.6. Figure 4 emphasizes this difference: the accuracy of the baseline model drops considerably with shuffled contexts, while the KV models maintain stable performance. In the shuffled scenario, KV-Llama3 achieves higher accuracy than baselines on the NQ, TQA, and POPQA datasets, with similar trends observed for KV-Llama3.1. These findings sug-

Dataset	NQ				TQA				POPQA			
Top-K	5	10	20	40	5	10	20	40	5	10	20	40
Llama3	34.1	37.2	40.4	38.8	62.4	66.5	67.2	64.7	31.7	33.7	33.7	31.6
Llama3.1	43.2	41.5	42.7	42.4	64.0	64.8	65.9	67.1	31.1	33.4	32.8	33.3
REPLUG-LLAMA3	35.6	34.0	33.6	32.2	57.7	56.4	55.8	56.3	30.1	28.1	26.0	26.7
REPLUG-LLAMA3.1	38.6	36.4	35.3	34.1	65.1	64.4	62.8	60.6	35.4	33.8	30.7	29.4
PAM QA	**51.9**	46.5	40.7	19.7	65.9	61.2	52.3	29.0	37.0	35.9	34.9	15.7
KV-LLAMA3	51.6	51.7	**51.4**	**49.8**	67.5	**68.8**	**69.3**	**69.3**	44.5	46.7	**48.3**	**46.7**
KV-LLAMA3.1	51.7	**51.8**	50.8	49.0	**68.6**	68.3	69.3	68.7	**44.7**	**47.6**	47.4	45.3

Table 1: Accuracy comparison with differnet position-invariant readers. Across top-k results, KV-fusion maintains stable and the strong accuracies, while other models either degrade or exhibit relatively low accuracies.

gest that KV-Fusion improves performance in RAG pipelines.

Comparison with Recent Methods We evaluate KV models alongside other position-agnostic methods: PAM QA (He et al., 2024), which employs multi-step reasoning to reduce position bias, and REPLUG (Shi et al., 2024b), which predicts the next token based on a weighted score for each context. Across the test sets, we utilize up to 40 DPR-retrieved passages, using default settings for PAM QA and the same configurations as Llama3 for REPLUG. As shown in Table 1, KV-models achieve the highest accuracy across datasets except top-5 NQ case. Notably, KV-models, originally trained with 20 passages, demonstrate strong robustness even with top-40 passages. PAM QA performs well with up to 20 passages but shows an average accuracy decline of 50.3% when scaled to the top 40. REPLUG follows the similar pattern as the baselines but also experiences performance degradation. Comparable results are observed with contriever passages as shown in Appendix A.7. These results indicate that KV-Fusion enhances robustness even with large input passages within the RAG pipeline.

5 Related Works

Retrieval Augmented Generation (RAG) With recent advancements in LLMs(Team et al., 2024; OpenAI, 2024), Retrieval Augmented Generation (RAG) have proven to be effective in complementing LLMs across various tasks: managing long-tail information (Mallen et al., 2023), reducing hallucinations (Huang et al., 2023b; Shi et al., 2024a), and improving interpretability (Borgeaud et al., 2022; Rudin et al., 2021). The idea of utilizing external knowledge has become prevalent, particularly in knowledge-intensive (Thorne et al., 2018; Lewis et al., 2021; Petroni et al., 2021), where retrievers like DPR and Contriever (Karpukhin et al., 2020; Izacard et al., 2021) first retrieve relevant information, and readers like FiD, ATLAS (Izacard and Grave, 2020; Izacard et al., 2022) incorporate the retrieved information to make predictions.

Robustness and Bias in RAG Pipeline Despite the promising capabilities of the RAG system, one major challenge is the notable drop in performance when irrelevant contexts exist during inference. (Shi et al., 2023; Oh and Thorne, 2023), along with incorrect responses even when the gold context appears in the middle (Liu et al., 2023). To address these issues, Xu et al. 2023 trained an auxiliary LLM to summarize and extract relevant contexts, while Yoran et al. 2024b proposed a simple Natural Language Inference (NLI) model to eliminate unnecessary passages. Also, He et al. 2024 suggests decomposing inference into multi-step resasoning, enabling the model to generate accurate response regardless of the context order. Other methods focus on internal features, such as adjusting position hidden states or calibrating attention biases (Hsieh et al., 2024b; Yu et al., 2024a). However, none of these approaches fully resolve a complete solution for 'Lost in the Middle' problem.

6 Conclusion

This paper presents KV-Fusion, a lightweight training scheme aimed at addressing positional bias and improving robustness of decoder-only models in RAG pipeline. KV-Fusion trains language models to be context-order invariant by extracting and prefilling KV caches with identical positional information, then training decoder-only models using these caches. The results not only highlight the robustness of KV-Fusion in handling a large number of input passages but also its position-invariant property. Our empirical evaluations on three open-domain datasets indicate that KV-Fusion can improve performance and reliability of the RAG system.

7 Limitations

One limitation of this work is its focus on question answering. Although the most common dataset for evaluating LLMs' understanding with large context would be the needle-in-a-haystack (NIAH) dataset (Kamradt, 2023), our experiments are centered around question-answering, which is more challenging than NIAH (Hsieh et al., 2024a).

Second limitation is that our experiments are limited to single-hop question answering, where multi-step reasoning is not required. For example, datasets like HotpotQA (Yang et al., 2018) and MuSiQue (Trivedi et al., 2022) require multiple passages to derive answers. This work, however, focuses on single-hop question-answering datasets, making it difficult to assess the impact of KV-fusion in multi-hop datasets.

Third limitation is that this work does not fully explore the use of KV-cache for training LLMs. Recently, training LLMs by conditioning key-value caches has gained attention (Sun et al., 2024), though our approach remains underexplored in terms of language modeling. However, we present strong empirical results to solve 'Lost in the middle' problem. We hope our work can facilitate future studies on utilizing key-value cache for training LLMs.

References

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, and Jian-Guang Lou. 2024. Make your llm fully utilize the context. *Preprint*, arXiv:2404.16811.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *Preprint*, arXiv:2310.11511.

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. 2022. Improving language models by retrieving from trillions of tokens. *Preprint*, arXiv:2112.04426.

Abhimanyu Dubey, others 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, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Feiteng Fang, Yuelin Bai, Shiwen Ni, Min Yang, Xiaojun Chen, and Ruifeng Xu. 2024. Enhancing noise robustness of retrieval-augmented language models with adaptive adversarial training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10028–10039, Bangkok, Thailand. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *Preprint*, arXiv:2002.08909.

Junqing He, Kunhao Pan, Xiaoqun Dong, Zhuoyang Song, LiuYiBo LiuYiBo, Qianguosun Qianguosun, Yuxin Liang, Hao Wang, Enming Zhang, and Jiaxing Zhang. 2024. Never lost in the middle: Mastering long-context question answering with position-agnostic decompositional training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13628–13642, Bangkok, Thailand. Association for Computational Linguistics.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024a. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*.

Cheng-Yu Hsieh, Yung-Sung Chuang, Chun-Liang Li, Zifeng Wang, Long Le, Abhishek Kumar, James Glass, Alexander Ratner, Chen-Yu Lee, Ranjay Krishna, and Tomas Pfister. 2024b. Found in the middle: Calibrating positional attention bias improves long context utilization. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 14982–14995, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023a. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *Preprint*, arXiv:2311.05232.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023b. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *Preprint*, arXiv:2311.05232.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning.

Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint*.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Atlas: Few-shot learning with retrieval augmented language models. *Preprint*, arXiv:2208.03299.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Gregory Kamradt. 2023. Needle in a haystack - pressure testing llms.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

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

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *Preprint*, arXiv:2212.10511.

Jacob Menick, Maja Trebacz, Vladimir Mikulik, John Aslanides, Francis Song, Martin Chadwick, Mia Glaese, Susannah Young, Lucy Campbell-Gillingham, Geoffrey Irving, and Nat McAleese. 2022. Teaching language models to support answers with verified quotes. *Preprint*, arXiv:2203.11147.

Philhoon Oh and James Thorne. 2023. Detrimental contexts in open-domain question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 11589–11605, Singapore. Association for Computational Linguistics.

OpenAI. 2024. Openai. 2024. gpt-4o.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.

Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. 2021. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Preprint*, arXiv:2103.11251.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. *Preprint*, arXiv:2302.00093.

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih. 2024a. Trusting your evidence: Hallucinate less with context-aware decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 783–791, Mexico City, Mexico. Association for Computational Linguistics.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024b. 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.

Yutao Sun, Li Dong, Yi Zhu, Shaohan Huang, Wenhui Wang, Shuming Ma, Quanlu Zhang, Jianyong Wang, and Furu Wei. 2024. You only cache once: Decoder-decoder architectures for language models. *Preprint*, arXiv:2405.05254.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, Soroosh Mariooryad, Yifan Ding, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Preprint*, arXiv:2403.05530.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Vincent Zhao, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Pranesh Srinivasan, Laichee Man, Kathleen Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed Chi, and Quoc Le. 2022. Lamda: Language models for dialog applications. *Preprint*, arXiv:2201.08239.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing. *Preprint*, arXiv:1910.03771.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. Recomp: Improving retrieval-augmented lms with compression and selective augmentation. *Preprint*, arXiv:2310.04408.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Howard Yen, Tianyu Gao, and Danqi Chen. 2024. Long-context language modeling with parallel context encoding. *Preprint*, arXiv:2402.16617.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2024a. Making retrieval-augmented language models robust to irrelevant context. In *The Twelfth International Conference on Learning Representations*.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2024b. Making retrieval-augmented language models robust to irrelevant context. *Preprint*, arXiv:2310.01558.

Yijiong Yu, Huiqiang Jiang, Xufang Luo, Qianhui Wu, Chin-Yew Lin, Dongsheng Li, Yuqing Yang, Yongfeng Huang, and Lili Qiu. 2024a. Mitigate position bias in large language models via scaling a single dimension. *Preprint*, arXiv:2406.02536.

Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, and Bryan Catanzaro. 2024b. Rankrag: Unifying context ranking with retrieval-augmented generation in llms. *Preprint*, arXiv:2407.02485.

A Appendix

A.1 Details for KV-Fusion Implementation

This section describes the pseudocode for KV-Fusion and Python implementation of the RES function.

A.1.1 Algorithm for KV-Fusion

This section further elaborates on the KV-Fusion algorithm, which can be implemented using standard language modeling. For clarification, we provide the pseudocode with a single-instance example. As explained in Section 2, KV-Fusion is built upon two decoders. First, $\mathcal{D}_{\text{p}}$ processes a set of input passages retrieved by retrievers, $C = \{c_1, c_2, \ldots, c_N\}$, and generates Key-Value (KV) caches in parallel. These KV caches are reshaped by the RES function into the form $\{K^l, V^l\}_{l=1}^{L}$ to prefill the cache in $\mathcal{D}_{\text{t}}$. Next, $\mathcal{D}_{\text{t}}$ processes target tokens, $t = \{t_1, t_2, \ldots, t_m\}$, along with their positional information, $p = \{p_{n+1}, p_{n+2}, \ldots, p_{n+m}\}$. Specifically, the target tokens consist of two parts: the query part, which includes instructions, $q = \{t_1, t_2, \ldots, t_k\}$, and the answer part, $y = \{t_{k+1}, t_{k+2}, \ldots, t_m\}$. Along with the prefilled KV cache, we train $\mathcal{D}_{\text{t}}$ by prompting it with q and using the standard language model loss to generate y. For implementation, we use `huggingface transformers`(Wolf et al., 2020) and `PyTorch`(Paszke et al., 2017) libraries.

Algorithm 1 Key Value Fusion(KV Fusion)

Input: $\mathcal{D}_{\text{t}}$, $\mathcal{D}_{\text{p}}$, Training Data $\mathcal{C}^{\texttt{train}} = \{C_1, C_2, \ldots, C_L\}$ where $C_i = \{c_1^i, c_2^i, \ldots, c_N^i\}$, Corresponding query tokens $q_i = \{t_1^i, t_2^i, \ldots, t_k^i\}$, Corresponding answer tokens $y_i = \{t_{k+1}^i, t_{k+2}^i, \ldots, t_m^i\}$

Initialize $\mathcal{D}_{\text{t}}$, $\mathcal{D}_{\text{p}}$ and freeze $\mathcal{D}_{\text{p}}$
for $i = 1, 2, ..., L$ **do**
    `# Extract KV-caches in parallel`
    $KV_{\texttt{cache}} = \mathcal{D}_{\text{p}}(C_i)$

    `# Reshape KV-caches`
    $\{K^l, V^l\}_{l=1}^{L} = \text{RES}(KV_{\texttt{cache}})$

    `# Compute loss and Optimize` $\mathcal{D}_{\text{t}}$
    $Loss = \mathcal{LM}_{\texttt{loss}}(y_i, \mathcal{D}_{\text{t}}(q_i; \{K^l, V^l\}_{l=1}^{L}))$
    Update parameters of $\mathcal{D}_{\text{t}}$ with repect to $Loss$ via gradient descent
end for

A.1.2 RES Implementation

To train $\mathcal{D}_{\text{t}}$ seaminglessly with `huggingface transformers`(Wolf et al., 2020) and `PyTorch`(Paszke et al., 2017), extracted KV-cache need to be reshaped to prefill the caches in $\mathcal{D}_{\text{t}}$. To this end, we implement RES function down below, which can also process batch of instances.

```
def reshape_key_value_batches(cur_past_key_values, n_psgs):
    """
    Reshapes key-value pairs in batches
    """
    new_key_cache = []

    # Iterate through each key-value pair
    for k, v in cur_past_key_values:
        # Split keys and values into splits (split by instance)
        k_splits, v_splits = torch.split(k, n_psgs, dim=0), torch.split(v, n_psgs, dim=0)

        # Reshape and concatenate splits (reshape by instance)
        k_re = torch.cat([torch.cat(torch.split(k_val, 1, dim=0), dim=2) for k_val in k_splits], dim=0)
        v_re = torch.cat([torch.cat(torch.split(v_val, 1, dim=0), dim=2) for v_val in v_splits], dim=0)

        # Append processed key-value pair
        new_key_cache.append((k_re, v_re))

    return tuple(new_key_cache)
```

A.2 Dataset Construction

A.2.1 Prompt Template

Prompt Template for Verifying Gold Passages and Supporting Evidence

Your task is to find Evidence from a given Document based on a Question and its corresponding Answer. Specifically, the Document contains the Answer for the given Question. Your job is to extract the Evidence from the document.

Here are the Question, Document, and Answer.

Question:
{QUESTION}

Document:
{PASSAGE}

Answer:
{ANSWER}

Here is how the evidence should be presented:

* Evidence
- The Evidence should only consist of sentences or paraphrases taken from the given Document.
- The Evidence should retain the same format as in the given Document.
- The Evidence should inlcude enough information to derive the given Answer from the given Question.
- If the provided Document does not contain sufficient information, generate NONE.

* Format
- DO NOT WRITE ANY GREETING MESSAGES, just write the evidence only.
- In front of the evidence, append the word “Evidence:".
- Write [END] after you are done.
- Here is the Example Format:
“
Evidence: evidence sentences [END]
“
- Do not include “ in the response.

Data Generation:

A.2.2 Dataset Statistics

Dataset	Training	Dev	Test
NQ	47,633	3,036	3,610
TQA	34,648	4,288	1,768
POPQA	6,833	1,190	1,267

Table 2: Dataset Statistics for NQ, TQA, and PQA

NQ and TriviaQA are filtered versions provided by Karpukhin et al. 2020 under the CC BY-NC 4.0.

We also use POPQA, available from the HuggingFace datasets[3] under MIT License. These datasets are English open domain question answering datasets based on December 2018 Wikipedia snapshot, preprocessed by Karpukhin et al. 2020.

- NQ dataset: Following the procedure outlined in Section 3.1, we obtained 47,633, 3,036, and 3,610 instances for the training, dev, and test sets, respectively.
- TQA dataset: To manage the GPT-4o API budget, the original TQA dev set was split in a 2:1 ratio, resulting in newly defined dev and test sets. This produced 34,648, 4,288, and 1,768 instances for the training, dev, and test sets, respectively.
- POPQA dataset: The original dataset did not include pre-defined training or dev sets. We split the data in an 8:1:1 ratio. After processing with the GPT-4 API, this resulted in 6,833, 1,190, and 1,267 instances for the training, dev, and test sets, respectively

A.3 Baseline Model Template and Example

Baseline Template

Title: {TITLE} Context: {TEXT}
=====
Title: {TITLE} Context: {TEXT}
=====
.
.
.
=====
Title: {TITLE} Context: {TEXT}
=====
Strictly based on listed documents (titles and contexts) above, answer the given question clearly and concisely in a single sentence. If none of the documents provide a valid answer, respond with "Unanswerable". Question: {QUESTION}? ANSWER:

Baseline Example

Title: Nobel Prize in Physics Context: Nobel Prize in Physics The Nobel Prize in Physics () is a yearly award given by the Royal Swedish Academy of Sciences for...
=====
Title: Nobel Prize Context: His son, George Paget Thomson, received the same prize in 1937 for showing that they also have the properties of waves...
=====
.
.
.
=====
Title: Nobel Prize Context: Wilhelm Röntgen's discovery of X-rays and Philipp Lenard's work on cathode rays. The Academy of Sciences selected Röntgen...
=====
Strictly based on listed documents (titles and contexts) above, answer the given question clearly and concisely in a single sentence. If none of the documents provide a valid answer, respond with "Unanswerable". Question: who got the first nobel prize in physics? ANSWER:

[3] https://huggingface.co/datasets/akariasai/PopQA

A.4 KV-Model Input Format Template and Example

The following examples outline the input format template along with a concrete example for $\mathcal{D}_{\text{p}}$.

Input Template for $\mathcal{D}_{\text{p}}$

Title: {TITLE} Context: {TEXT}
=====

Input Example for $\mathcal{D}_{\text{p}}$

Title: Does He Love You Context: Does He Love You "Does He Love You" is a song written by Sandy Knox and Billy Stritch, and recorded as a duet by American country music artists Reba McEntire and Linda Davis. It was released in August 1993 as the first single from Reba's album "Greatest Hits Volume Two". It is one of country music's several songs about a love triangle. "Does He Love You" was written in 1982 by Billy Stritch. He recorded it with a trio in which he performed at the time, because he wanted a song that could be sung by the other two members
=====

The following example outlines the input format template along with a concrete example for $\mathcal{D}_{\text{t}}$.

Input Template for $\mathcal{D}_{\text{t}}$

<|question_answering|> Using the provided titles and contexts, answer the given question briefly and provide the supporting sentences as evidence.
Question: {QUESTION}?
Answer: {ANSWER} [RESULT]
Evidence: {EVIDENCE} [END]

Input Example for $\mathcal{D}_{\text{t}}$

<|question_answering|> Using the provided titles and contexts, answer the given question briefly and provide the supporting sentences as evidence.
Question: who sings does he love me with reba?
Answer: Linda Davis [RESULT]
Evidence: "Does He Love You" is a song written by Sandy Knox and Billy Stritch, and recorded as a duet by American country music artists Reba McEntire and Linda Davis. [END]

605 A.5 Hyperparameters for training

Training Hyperparameters	
flash attention 2	True
target token max length	192
number of input contexts	20
input token max length	192
epochs	2
batch size per gpu	2
gradient accumulation	8
learning rate	2e-5
warmup ratio	0.05
scheduler	cosine
optimizer	adamW

Table 3: Across all datasets, we utilize four A100 80 GB GPUs, with a batch size of 2 per device and a gradient accumulation of 8, consuming approximately 12 hours (48 GPU hours). The number of training passages is 20 consisting of one gold context and 19 negative contexts. Contexts are tokenized with a maximum length of 192 tokens using left-padding; if a context exceeds this limit, tokens are truncated from the left. A random sample of 10,000 contexts from the NQ training set showed that 99.0% of contexts fit within this token limit.(Avg: 145 tokens, Std: 14 tokens) The maximum learning rate is set to 2×10^{-5}, using a linear warmup and cosine decay. The warmup ratio is set to 5%. The AdamW optimizer(paged_adamw_32bit) is used with $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

606 A.6 KV-Llama3.1 and Llama3.1

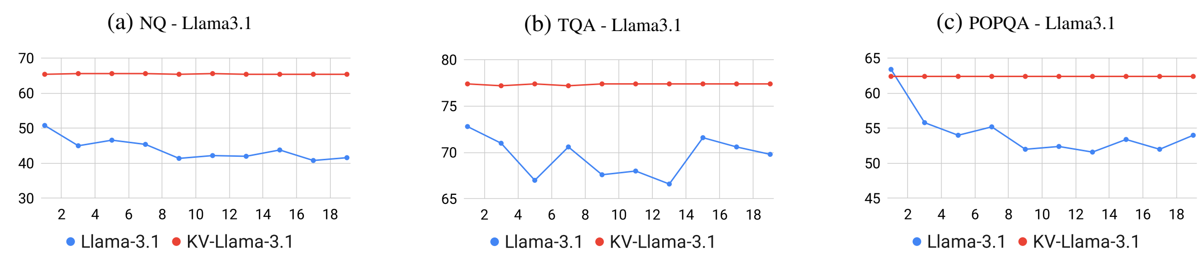

Figure 5: Comparison of EM Accuracy between KV-Llama3.1 and Llama3.1 across different gold context positions. With varying gold context positions, KV-Llama3.1 illustrates consistent accuracies across datasets. However, Llama3.1 suffers from 'lost in the middle' problem, which can be resolved by KV-Fusion models.

607 A.7 Position Agnostic reader evaluation on Contriever retrieved passages

Dataset	NQ				TQA				POPQA			
Top-K	5	10	20	40	5	10	20	40	5	10	20	40
Llama3	29.6	32.0	36.6	36.1	55.1	59.7	61.8	58.7	37.5	39.1	37.7	36.9
Llama3.1	38.8	36.2	37.7	38.4	58.4	57.6	59.4	60.7	38.8	38.8	38.4	38.8
REPLUG-LLAMA3	33.4	33.1	31.3	31.0	54.2	54.5	55.2	55.8	32.4	28.0	25.8	23.9
REPLUG-LLAMA3.1	35.3	34.9	33.4	32.7	61.0	60.9	60.1	60.4	39.9	34.3	30.8	28.8
PAM QA	**49.9**	44.1	38.1	18.9	64.4	57.7	52.7	28.5	51.0	48.8	44.0	24.2
KV-LLAMA3	49.1	50.6	**50.3**	**49.1**	65.1	**67.1**	68.4	**68.3**	**53.9**	**56.6**	**54.5**	**51.9**
KV-LLAMA3.1	48.9	**50.9**	**50.3**	48.9	**65.7**	66.3	**68.8**	67.8	53.0	54.8	54.1	51.3

Table 4: Accuracy comparison with other position-invariant methods on contriever-retrieved passages. KV-Fusion models achieve the highest accuracies across datasets except NQ top-5 case. Consistent with the results observed for DPR-retrieved passages in Table 1, KV-Fusion models show strong robustness to the inclusion of additional passages, while other methods experience a decline in performance as more passages are added. Notably, the strong performance on POPQA datasets highlights Contriever's ability to excel on unseen datasets, which KV-Fusion models effectively leverage. This demonstrates that KV-Fusion models can achieve strong performance on different retrievers.