Memory-Augmenting Decoder-Only Language Models through Encoders (Student Abstract)

Alessio Galatolo[1, 2], **Katie Winkle**[1]

[1] Department of Information Technology, Uppsala University
[2] Department of Women's and Children's Health, Uppsala University
alessio.galatolo@it.uu.se, katie.winkle@it.uu.se

Abstract

The Transformer architecture has seen a lot of attention in recent years also thanks to its ability to scale well and allow massive parallelism during training. This has made possible the development of Language Models (LMs) of increasing size and the discovery of latent abilities that completely outclass traditional methods e.g. rule-based systems. However, they also introduced new issues, like their inability to retain the history of previous interactions due to their stateless nature or the difficulty in controlling their generation. Different attempts have been made to address these issues, e.g. a 'brute force' approach to solving the memory issue is to include the full conversation history in the context window, a solution that is limited by the quadratic scalability of Transformers. In this work, we explore computationally practical solutions to the memory problem. We propose to augment the decoder-only architecture of (most) Large LMs with a (relatively small) memory encoder. Its output is prepended to the decoder's input in a similar fashion to recent works in Adapters and the original Transformer architecture. Initial experiments show promising results, however future work is needed to compare with State-of-the-Art methods.

Introduction

Large Language Models (LLMs) have gained a lot of traction lately thanks to their content flexibility and human-like fluency. However, they also present many shortcomings, from the difficulty of controlling content output, to their - possibly harmful- hallucinations to their high computational costs both for training and inference.

LLMs are based on the Transformer architecture proposed by Vaswani et al. (2017) which originally comprised an encoder and a decoder and was aimed at seq-to-seq tasks. However, most language models that show 'interesting' emergent abilities such as in-context learning (Brown et al. 2020), instruction following (Ouyang et al. 2022), etc. follow a decoder-only architecture.

In this work, we tackle one of the problems given by the stateless nature of LLMs, that is their inability to keep a history of previous conversations. Keeping a memory of previous conversations is a common ability in humans and its expectation is also projected in LLMs. Further, memory can also provide a way for existing models to acquire and use new knowledge that e.g. was not available in the training data due to its specificity or a temporal difference.

The ever-growing size of the context window for Transformer-based models, see 32k tokens for GPT-4[1], offers a 'brute force' solution to the problem whereby any relevant memory/knowledge is included in the context window. However, given the quadratic scalability of the Transformer architecture (Vaswani et al. 2017) the need for a different solution is obvious.

Other approaches to this problem include: the use of differentiable k-NNs to augment a generic Transformer architecture (Fan et al. 2021), extension of the Transformer architecture to allow the inclusion of memory (Martins, Marinho, and Martins 2022), and implementation of memory as added tokens to the input/output (Bulatov, Kuratov, and Burtsev 2022).

Methods

Our approach consists of building on top of pretrained decoder-only models such as LLaMA (Touvron et al. 2023) and augmenting them with an encoder. This encoder will only take as input the memory (e.g. conversation history, new knowledge, etc.) and will output its encoded version. The decoder (plain LLaMA), will receive the regular input (e.g. a user query) and the output of the encoder (the memory).

We acknowledge that separately trained encoders may not be able to reach the same performance as models designed to have an encoder since the beginning of the training (Zhong, Lei, and Chen 2022), however we think this method (and our approach) represents a good trade-off between good performance and the computational costs of pretraining a model from scratch.

Our approach is also similar in concept to works that use Adapters to fine-tune language models (Zhang et al. 2023) with the difference that our encoder (replacing the adapter) receives a different input compared to the decoder. While the decoder's goal is still to answer the original query, the encoder receives the memory and encodes it to influence the decoder's output.

[1] https://platform.openai.com/docs/models/gpt-4

Architecture

We start our experiments with an encoder obtained by mirroring LLaMA's decoder architecture. From the original Transformer architecture (Vaswani et al. 2017), a decoder block and encoder block differ for (i) the cross-attention that is only present in the decoder and (ii) the masked attention in the decoder that is *not* masked in the encoder. Given that decoder-only architectures do not contain cross-attention, we obtain our encoder by only removing the masking from the attention of the decoder block.

Compared to the LLaMA decoder, in the encoder, we scale down the dimension of each block and the total number of blocks. This is done to reduce the training computational and data needs. Given both the reduced dimension and the absence of cross-attention, we inject the encoded memory by prepending it to the decoder's input. We experiment with a different number of layers (blocks), settling for eight as a trade-off between performance and computational needs. We also scale down the dimension from 4096 to 128. We provide our code in our GitHub repository[2].

Training

When training, we freeze the weights of the decoder and only train the encoder based on the output of the whole model (backpropagating from the output of the decoder). We pretrain the encoder using a linearly increasing learning rate (1e-5 to 1e-4) for two epochs with a reconstruction objective, feeding the decoder an empty string and computing the cross-entropy loss of the memory input and the decoder's output. This is followed by three epochs of fine-tuning where we feed the decoder the regular input and prepend the encoder's output at every layer, we then compute the cross-entropy loss with the target. For the learning rate, we used a cosine decay from 1e-4.

Datasets

For both training and pretraining we use four different datasets: Curiosity (Rodriguez et al. 2020) is a collection of conversations on a specific topic with separate knowledge (memory) from which they are based. Dialogue-based REAding comprehension exaMination (DREAM) (Sun et al. 2019) is a dataset that is aimed at text comprehension, we adopt the text as our memory and the question as input to the system. Schema-Guided Dialogue (SGD) (Rastogi et al. 2020) is a dataset containing task-oriented conversations where a virtual assistant answers queries based on retrieved data (memory). Wizard of Wikipedia (WoW) (Dinan et al. 2019) contains conversations grounded on knowledge from Wikipedia (memory).

Results

We compare our method to two simple baselines: (i) regularly feeding the input to the decoder with no inclusion of memory and (ii) prepending the memory to the regular input of the decoder (similar to the brute force approach we described earlier). Note how (i) is not an absurd baseline as some of the queries can be answered without the memory and just with 'common knowledge' e.g. from the WoW dataset: "Are they [armadillos] native to a Spanish-speaking part of the world?".

[2]https://github.com/alessioGalatolo/Memory-Encoder-LLaMA

Method	Cross-Entropy Loss
Memory encoder (ours)	**7.73**
No inclusion of memory	19.97
Prepending plain memory to input	21.37

Table 1: Summary of results

Preliminary results on our validation set show a better cross-entropy loss of our method compared to both baselines, summarised in Table 1.

Conclusions and Future Work

Our method is able to encode and compress a given memory and augment the generation of the decoder without the need for further training, outperforming simple approaches. With the encoder's reduced size, we improve over our baselines by prepending to the input only 16 tokens.

This method shows promising opportunities for future work. For instance, we plan to extend the pretraining phase with additional, more diverse datasets. Experimenting with a different number of encoder layers and dimensions and possibly fine-tuning the decoder after the encoder's training.

Acknowledgements

The computations and data handling were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) at Alvis, C3SE (Chalmers) partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

References

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Bulatov, A.; Kuratov, Y.; and Burtsev, M. 2022. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35: 11079–11091.

Dinan, E.; Roller, S.; Shuster, K.; Fan, A.; Auli, M.; and Weston, J. 2019. Wizard of Wikipedia: Knowledge-Powered Conversational Agents. In *International Conference on Learning Representations*.

Fan, A.; Gardent, C.; Braud, C.; and Bordes, A. 2021. Augmenting transformers with KNN-based composite memory for dialog. *Transactions of the Association for Computational Linguistics*, 9: 82–99.

Martins, P. H.; Marinho, Z.; and Martins, A. F. 2022. ∞-former: Infinite Memory Transformer-former: Infinite Memory Transformer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 5468–5485.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744.

Rastogi, A.; Zang, X.; Sunkara, S.; Gupta, R.; and Khaitan, P. 2020. Towards Scalable Multi-Domain Conversational Agents: The Schema-Guided Dialogue Dataset. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05): 8689–8696.

Rodriguez, P.; Crook, P.; Moon, S.; and Wang, Z. 2020. Information Seeking in the Spirit of Learning: a Dataset for Conversational Curiosity. In *Empirical Methods in Natural Language Processing*.

Sun, K.; Yu, D.; Chen, J.; Yu, D.; Choi, Y.; and Cardie, C. 2019. Dream: A challenge data set and models for dialogue-based reading comprehension. *Transactions of the Association for Computational Linguistics*, 7: 217–231.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Zhang, R.; Han, J.; Zhou, A.; Hu, X.; Yan, S.; Lu, P.; Li, H.; Gao, P.; and Qiao, Y. 2023. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*.

Zhong, Z.; Lei, T.; and Chen, D. 2022. Training Language Models with Memory Augmentation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 5657–5673.