Llama-UL2: Emerging New Capabilities with Continued Pretraining using UL2

Stanford CS224N Custom Project

Jason Wang
Department of Computer Science
Stanford University
jsywang@stanford.edu

Abstract

Causal language modeling has long been the dominant pretraining objective for large decoder-only transformer models. However, it is not the only objective when pretraining language models. In this paper, we explored the possibility to steer language models and bake in new capabilities by continue pretraining an existing causal LM, Llama2-7B (Touvron et al., 2023), using a different objective called UL2 (Tay et al., 2023). The resulting model, Llama-UL2-7B, showed emergent task performance and acquires new capabilities introduced by the UL2 objective.

1 Key Information to include

- Mentor: Kaylee Burns

2 Introduction

Language modeling is the task of predicting the next token in a given sentence. It is one of the most important task to tackle with in natural language processing as it is widely used in applications such as machine translation, text auto-completion, and virtual assistant (chatbot). Given the nature of this task, the most straightforward way is to use the causal language modeling objective, which constraints that the next token is only conditioned on the previous tokens. However, such naive objective has several limitations which suggest that it may not be the best idea to pretrain a language model. For example, LMs pretrained with causal objective is not able to do text infilling, and it doesn't utilize bidirectional information in the prompt when generating text. Because of this, researchers have proposed newer and more sophisticated pretraining objectives to unlock more capabilities of the language model. One example is UL2, which combines prefix language modeling with span corruption.

However, pretraining language models from scratch is computationally expensive. With the vast amount of open-source causalLM checkpoint available online, it would be nice if there is a way to use small amount of compute to bake these missing abilities into existing LMs while preserving its original learned abilities.

In this project, we take an existing Llama2-7B checkpoint and continue pretraining it using the UL2 objective for around 0.2% additional data to get Llama-UL2-7B. We show that adapting Llama2-7B with UL2 objective gives it the capability of doing text infilling despite only trained for a relatively small amount of data. Llama-UL2-7B also showed emergent task performance such that it even outperformed Llama2-13B (2x of its size) on some challenging tasks.

3 Related Work

3.1 Large Language Models

Language Models, nowadays usually built using the Transformer architecture (Vaswani et al., 2023), has gained big success in the NLP area. Researchers have found that these transformer models scales up easily, providing promising task performance while maintaining reasonable hardware utilization. As a result, we are seeing the size of these transformers blowing up from 117M GPT-1 (Radford et al., 2018) to (rumored) 1.8T GPT-4 (OpenAI, 2024). While some of the largest and most powerful LLMs remain closed-source, some organizations have decided to open-source their LLMs in various sizes, such as Llama and Mixtral (Jiang et al., 2024).

3.2 Emergent Abilities

Researchers have found that new behaviors arises while scaling up language models. For example, the GPT-3 (Brown et al., 2020) paper found that when scaling LLMs to 175B, the model gains the ability to do few-shot learning by leveraging examples from the input without any weight updates. These phenomenon is later referred to as emergent abilities, which Wei et al. (2022b) defined it as "abilities that are not present in smaller models but as present in larger models." Usually, the classical way to get such behavior is by 1) increase its size, 2) increase the amount of data, 3) increase the amount of compute. Here we show that a fourth way to get emergent abilities is by continue training the model with a different objective.

3.3 Causal Language Modeling

Causal Language Modeling, denoted as the causal objective in this paper, is the most common way to do language modeling nowadays. It models the dependencies of the tokens by only allowing tokens to attend to tokens on their left, resulting in a lower triangle attention mask, and every position in the transformer outputs a conditional distribution of $P(x_t|x_{<t})$. Pretraining causal language models works by maximizing the log likelihood of the token sequences in the dataset, which are possibly curated raw text in the wild. As mentioned in the introduction, even though this objective fits the definition of the language modeling task (predicting the next word from previous words), it still has several limitations. For example, in most use cases of language models, the model is usually given a prompt and is expected to generate text conditioned on the prompt. Since the prompt is given but not generated by the model, the model could acutually utilize bidirectional information when encoding the prompt. This is not possible in causal LMs because of the lower triangular attention mask. Instead, it can only utilize unidirectional information in the prompt as if it is generating the prompt as well.

3.4 Prefix Language Modeling

We know that an encoder-decoder transformer architecture such as T5 (Raffel et al., 2023) is an easy solution to the above limitation, where the encoder transformer considers bidirectional information of the input prompt and the decoder transformer considers unidirectional information when generating the response. But can we achieve a similar effect on decoder-only architectures? The answer is to use prefix language modeling, which emulates an encoder-decoder architecture using a decoder-only transformer. Prefix language modeling combines a fully visible attention mask (encoder) with a causal attention mask (decoder) to ensure that the prompt tokens can attend to any other prompt tokens and the generated tokens can only attend to the previous tokens (See Figure 1).

3.5 Continued Training

Continue training a language model is uaually referred to as finetuning. Prior works have shown that finetuning a language model can substantially improve the model's performance on downstream tasks. For example, FLAN (Wei et al., 2022a) performs instruction finetuning on an LM using a collection of tasks and found that it improves the model's performance on unseen tasks. Another method called LoRA (Hu et al., 2021) freezes the base model and inject trainable adapters to finetune the model, resulting in reasonable performance with less memory footprint. However, these finetuning methods usually require additional dedicated dataset that are relevant to downstream tasks. We denote our

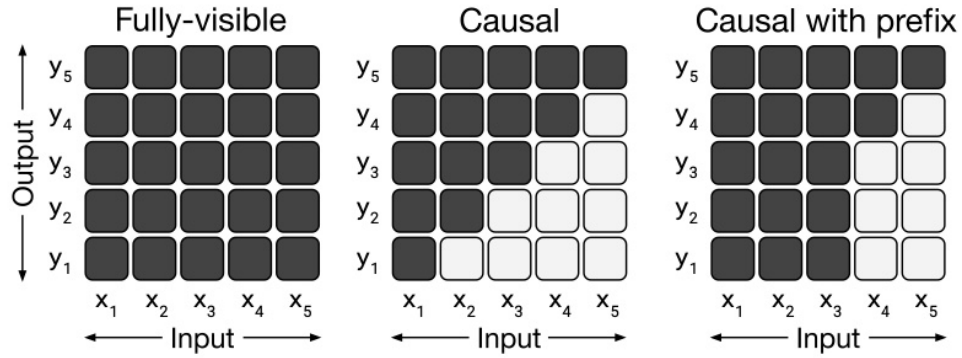

Figure 1: Attention masks of different model types. The input and output of the self-attention mechanism are denoted x and y respectively. A dark cell at row i and column j indicates that the self-attention mechanism is allowed to attend to input element j at output timestep i. A light cell indicates that the self-attention mechanism is not allowed to attend to the corresponding i and j combination. (Raffel et al., 2023)

method as continued **pretraining** so as to differ from the general continued training, because our method works on any pretraining dataset and can therefore be seen as second-stage pretraining.

4 Approach

4.1 Model Architecture

We use the standard Llama2-7B architecture. It is a decoder-only transformer model with RoPE embeddings (Su et al., 2023), RMSNorm (Zhang and Sennrich, 2019), and swiGLU activation (Shazeer, 2020). More details can be found in Table 1

hidden size	4096
intermediate size	11008
sequence length	4096
# attention heads	32
vocab size	32000

Table 1: The Llama2-7B architecture details.

4.2 UL2 Objective

The UL2 objective builds upon the prefix language modeling idea. It preprocesses data by adding noise and asking the model to denoise. To make the objective more robust, UL2 defines a mixture of denoisers such that each denoiser has a different characteristic, ranging from span corruption to simple causal generation. Our approach follows the UL2R (Tay et al., 2022) setup, using the following 3 types of denoisers. See Appendix A.1 for more details for each denoiser's configuration.

- **R-Denoiser** The regular denoiser is the standard span corruption task introduced in (Raffel et al., 2023).
- **X-Denoiser** The extreme denoiser performs span corruption in a more aggressive manner, either by corrupting longer spans or by corrupting short spans with higher probability.
- **S-Denoiser** The sesquential denoiser resembles the simple causal generation by only corrupting one span that always ends in the end of the sequence.

For every input data, we randomly sample a denoiser based on the mixture weights and use that denoiser to corrupt the data. To enable different prompting mode, we add a task token corresponding

to the denoiser type to the beginning of the sequence. At the end of the sequence, we also add a special sentinel token "<sentinel_0> to signal the end of prompt and start of generation. A good mathematical intuition in this approach is that given n raw tokens from the dataset, assume μ is the average span length, r is the corruption rate, the number of tokens after UL2 preprocessing is a fixed numbeer of $n(1 + \frac{2r}{\mu})$ (see Appendix A.2 for derivation of this equation).

Figure 2 has some visualizations on how the data is preprocessed by the mixture of denoisers. By using a mixture of these 3 denoisers, we prevent the model from only learning to do text infilling, and ensures that it preserves its ability to do causal generation.

As it is a prefix language modeling task, in addition to the prefix attention mask, we use a loss mask to mask out the input token loss and only use the cross entropy loss in the output portion to do gradient descent.

We chose to use the UL2 objective to do continue pretraining because: 1) It is a prefixLM objective which considers bidirectional information in the prompt, which is better than unidirectional information in causalLM, 2) it seamlessly bakes in text infilling capability to the model during training, 3) it preserves the ability to do causal text generation and 4) the model's previous knowledge learned from causal pretraining could potentially help UL2 continue pretraining because the objectives are similar (text generation).

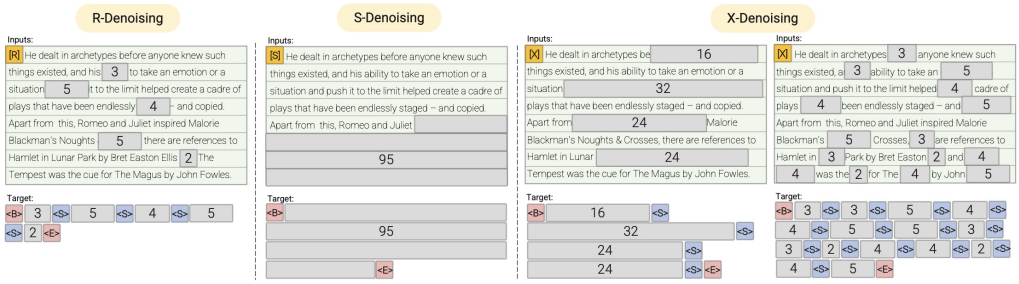

Figure 2: Mixture of denoisers for training UL2. The i-th greyed out token span replaced by a single "<sentinel_i>" token and the true corrupted tokens are shifted to the output following their corresponding sentinel token. (Tay et al., 2023)

5 Experiments

5.1 Data

We chose to use the OpenWebText dataset (Gokaslan and Cohen, 2019), which is an open-source corpus that contains web contents from URLs shared on Reddit with at least 3 upvotes. We'll use this dataset to perform causal generation and span corruption tasks. When using UL2, we will preprocess the dataset using the mixture of denoiser to get inputs and outputs as in Figure 2.

5.2 Baseline

For baseline, we use the Llama2-7B architecture with its pretrained (non-finetuned) checkpoint released by Meta in Huggingface[1].

5.3 Evaluation method

For evaluation we mainly evaluate its natural language understanding ability as well as some emergent abilities.

To evaluate natural language understanding, we used two common benchmarks:

[1] `https://huggingface.co/meta-llama/Llama-2-7b-hf`

- **Boolq** (Clark et al., 2019): a yes/no question answering task to evaluate reading comprehension. Each example is a triplet of (question, passage, answer (yes/no)).
- **Piqa** (Bisk et al., 2019): a question answering task with binary options to evaluate physical commonsense reasoning.

To evaluate emergent abilities, we used four tasks in the BigBench emergent suite (BBES) (bench authors, 2023):

- **geometric shapes**: This task evaluates visual reasoning by testing the model's ability to infer the shape given an SVG path.
("<path d="M 38.35,49.41 L 31.18,9.15"/>" $\Rightarrow$ "triangle")
- **navigate**: This task evaluates spatial reasoning by testing the model's ability to predict if an agent is back in its original position after a series of moving commands.
("Take 1 step. Take 2 steps. Take 3 steps. Turn around. Take 6 steps. Turn left." $\Rightarrow$ "True")
- **snarks**: This task tests if the model can detect which one of the two statements is sarcastic.
("(a) Concrete plates. Very beautiful. (b) Stained-glass plates. Very beautiful." $\Rightarrow$ "(a)")
- **understanding fables**: This task tests if the model can understand the moral of stories.
("Some storks chose a field newly sown with wheat as their new feeding ground. The owner of the field, for a while, scared them away by waving an empty sling. Once the storks found out that the sling was empty, they started to ignore the farmer's threats. In response, the farmer charged the sling with stones, killing dozens of the birds in quick succession. What is the moral of this story?" $\Rightarrow$ "If words suffice not, blows must follow.")

Due to limitations in compute, all of the above benchmarks are zero-shot multiple choice questions and the performance are measured by accuracy. We determine the chosen option by finding the option with the largest likelihood given the question

$$\mathcal{L}(option^{(i)}|question) = \prod_{k=0}^{len(option^{(i)})} P(option_k^{(i)}|option_{<k}^{(i)}, question)$$

$$chosen\ option = \arg\max_i \mathcal{L}(option^{(i)}|question)$$

Although in practice this is usually done in log-scale (picking the option with the maximum log likelihood, which can be easily calculated by $F.log\_softmax(model(inputs).logits).sum())$

5.4 Experimental details

We take the official Llama2-7B checkpoint (trained with 2T tokens using causal objective) and continue training it using UL2 objective. We used a cosine annealing learning rate from 10^{-4} to 10^{-6}. We use a batch size of 256 and sequence length of 4096 and trained the model for an additional 4000 steps, resulting in a total of $256 \times 4096 \times 4000 \approx 4$ billion tokens, which is roughly 0.2% of the 2T pretraining tokens reported by Meta. It took around one day to finish the training. The training runs FSDP (Zhao et al., 2023) using 64 TPUv3 chips. The resulting model is called Llama-UL2-7B.

To make comparisons, we also continued training the official checkpoint using causal objective on the same dataset for the same number of tokens (2B).

To understand the synergy between the causal objective and the UL2 objective, we trained a Llama2-7B from scratch using causal objective for 500M tokens, taking 5 evenly-spaced checkpoints during training and continue training them using UL2 objective for 50M tokens (extra 1% compute).

5.5 Results

5.5.1 Training perplexity

We first look at the synergy between causal pretraining and UL2 continued training. In Figure 3, we can see that pretraining using causal objective helps future UL2 training. The more tokens the model is pretrained using causal objective, the lower its training and validation loss is when continue

training using UL2 objective. Note that here the training loss is the UL2 loss, which decreases during training. The validation loss is the causal loss, so there is a slight increase during training.

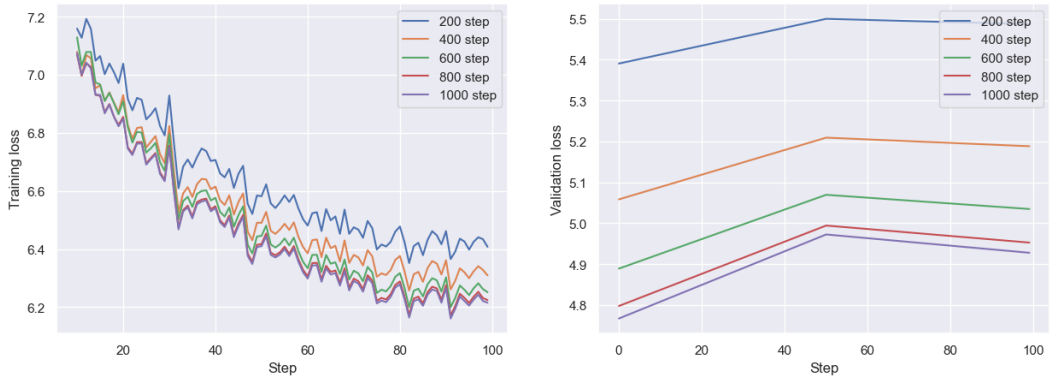

Figure 3: Training and validation loss of UL2 continue training from 5 evenly-spaced checkpoint after causal pretraining. Model was trained for 100 steps using a batch size of 256 and a sequence length of 2048 (consumed ∼50M tokens).

However, if we train the model for longer, the causal validation loss would start to decrease as well. In Figure 4, when continue training from the official checkpoint for 4000 steps, we can see that the UL2 validation losss decreases together with the UL2 training loss, and the causal validation loss also goes down in the course of training. This suggests that the UL2 objective can also benefit causal generation performance.

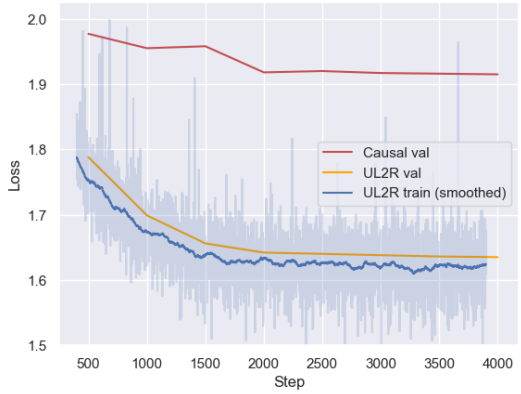

Figure 4: Training and validation loss of UL2R continue training from meta-llama/Llama-2-7b-hf. Model was trained for 4000 steps using a batch size of 256 and a sequence length of 4096 (consumed ∼4B tokens).

5.5.2 Big-Bench results

We show the results in the 4 chosen BBES tasks comparing multiple models in Figure 5. We marked the performance of random guessing for each task, which is simply $(1/\#\ options)$. In Appendix A.3, we also include the accuracy scores from Llama2-13B just to see how much does model scale matters in these tasks.

We observe that Llama-UL2-7B has gained emergent abilities by getting a significantly higher score than other models in some of the chosen tasks (e.g. geometric shapes & snarks). Some of the scores even beats Llama2-13B whose size is almost two times larger. We also see that emergent abilities can be gained by scaling up the model, as Llama2-13B has significantly outperformed the 7B models in the understanding fables task. We suspect that this is because the task has a much longer input and output sequence length, and a larger model is simply better at handling longer sequence even though they were all trained using the same sequence length (4096) during pretraining phase.

Although emergent abilities are usually obtained by scaling up model size, the BBES results show that with negligible amount of compute, UL2 continued pretraining can help the model achieve similar effect.

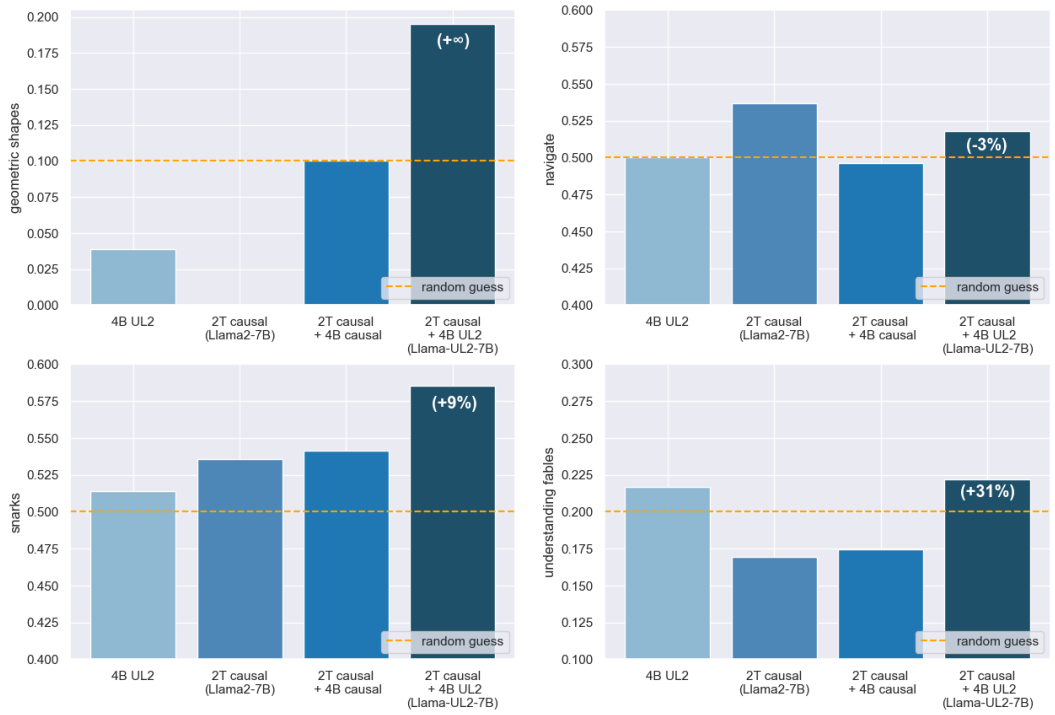

Figure 5: BBES results on Llama2-7B with reference to random performance (orange line).

5.5.3 NLU results

We also show results on standard natural language understanding (NLU) tasks Boolq and Piqa in Figure 6. Here we observe that the performance of Llama-UL2-7B is slightly worse than the baseline (-2%). We suspect that this is probably because of data quality as the causal continued pretraining run also has performance degrades similar to Llama-UL2-7B. Since Meta didn't release the pretraining dataset for Llama2 models, we cannot run UL2 on the same dataset as what it has been trained on. OpenWebText is a smaller and possibly lower-quality dataset. Therefore, the evaluation results of these NLU tasks slightly dropped.

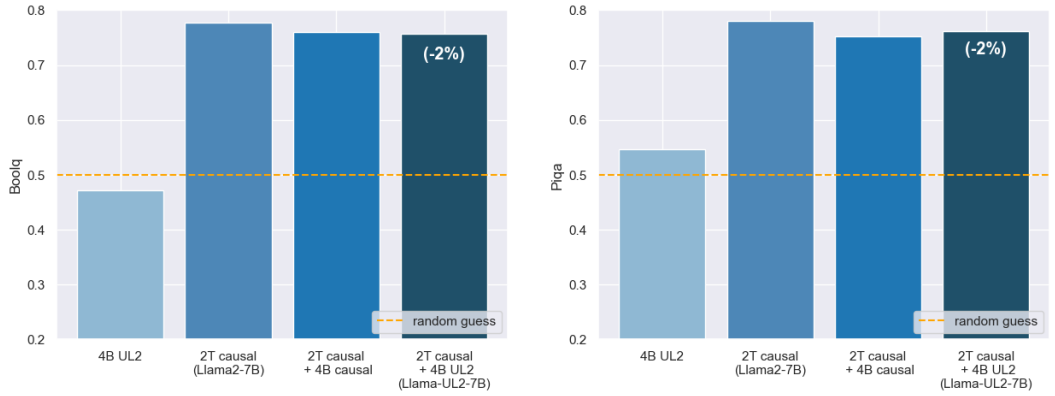

Figure 6: NLU tasks results on Llama2-7B with reference to random performance (orange line)

6 Analysis and Limitations

One of the key benefits of using UL2 objective is that it not only preserves the model's ability to do causal generation, it also injects the ability of doing text infilling to the model. In this section we show that the model has successfully acquired this new skill.

Prompt:

Steps to make a cake:
1. Prepare Ingredients
<1> with a <2>
3. Preheat oven to <3> for <4>
4. Put the cake into <5>
5. Take the cake out of the oven

Model Output:

<1> 2. Mix ingredients
<2> spoon
<3> 450
<4> 15 minutes
<5> the oven

From this example, we can clearly observe that the model has acquired the skill of text infilling. This is a very useful ability for use cases similar to the above example. When a user forgets some detail in a document (e.g. a recipe), it can directly ask the model to fill out the details instead of asking "What is the second step of making a cake", which could be ambiguous can would require more sophisticated prompting technique to let the model know the context of the recipe.

We also analyzed certain failure modes and limitations of UL2 continued pretraining. One limitation is that we don't have precise control on how many tokens the model will fill in the blank. In Appendix A.4, we prompt the model with some simple sentences expecting it to only fill in one word, but the model fills in a reasonable longer phrase instead. In future work, one could potentially modify the UL2 preprocessing algorithm to somehow include the number of tokens to fill in the prompt.

Another limitation of our approach is the inefficient utilization of data. When running any kinds of prefixLM objectives, the prompt tokens must be masked out when computing the loss. This is especially bad for UL2 because the input has length $(1 - r)$ where r is the corruption rate, which is usually smaller than 0.5. This means more than half of the tokens were wasted because the prompt logits weren't used to calculate loss for gradient descent.

In earlier stage of this project, we extract a fixed number of raw tokens from the dataset so that the resulting number of tokens after UL2 preprocessing is equal to the maximum sequence length (4096) following Equation A.2 in order to maximize FLOP utilization. This turned out to be problematic because we observe that the model's cannot fill in blanks with short prompts (e.g. the cake prompt above). The only way to let the model fill in blanks is to append around 3000 tokens with around 180 more blanks after our prompt. This is quite surprising and funny to see because this means the model overfits to the fact that during training it always sees around 186 blanks in the prompt, so it won't start doing its job unless there were 186 blanks to fill. We solved this issue by randomizing the extracted raw tokens length so that the model is able to deal with any prompt length during evaluation.

However, this results in even more waste in FLOP utilization: not only the prompt tokens were masked out, the pad tokens were also masked out. This could be solved by a technique called sequence packing which is standard in PyTorch, but not an easy implementation in our training platform in JAX[2]. We will leave this for future work as it doesn't affect model quality (despite being inefficient)

7 Conclusion

In this project, we show that it is possible to steer a language model by using UL2 objective. Our approach successfully converted a causalLM into a prefixLM, enabling bidirectional attention when processing input prompts. We further verify that with negligible additional compute and data, UL2 continued pretraining injects text infilling ability to a causally pretrained Llama2-7B and observe that the resulting model, Llama-UL2-7B, has emergent abilities on several challenging tasks in BigBench. Finally, we analyzed two limitations to our approach and proposed potential ways mitigate the issue. We also left future works to optimize token efficiency using sequence packing.

[2] `https://github.com/stanford-crfm/levanter`

References

BIG bench authors. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts.

OpenAI. 2024. Gpt-4 technical report.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer.

Noam Shazeer. 2020. Glu variants improve transformer.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2023. Roformer: Enhanced transformer with rotary position embedding.

Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. 2023. Ul2: Unifying language learning paradigms.

Yi Tay, Jason Wei, Hyung Won Chung, Vinh Q. Tran, David R. So, Siamak Shakeri, Xavier Garcia, Huaixiu Steven Zheng, Jinfeng Rao, Aakanksha Chowdhery, Denny Zhou, Donald Metzler, Slav Petrov, Neil Houlsby, Quoc V. Le, and Mostafa Dehghani. 2022.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022a. Finetuned language models are zero-shot learners.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022b. Emergent abilities of large language models.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. 2023. Pytorch fsdp: Experiences on scaling fully sharded data parallel.

A Appendix (optional)

A.1 Mixture of Denoisers Configuration

Denoiser	Mixture	Setting
R	0.25	$(\mu = 3, r = 0.15, n)$
S	0.5	$(\mu = \frac{L}{4}, r = 0.25, 1)$
X	0.25	$(\mu = 3, r = 0.5, n) \cup (\mu = 32, r = 0.15, n)$

Table 2: UL2 denoisers configurations. The last parameter denotes how many spans should be corrupted.

A.2 Derivation of number of tokens after UL2 preprocessing

$$\text{\# corrupted tokens} = nr$$

$$\text{\# spans} = \frac{\#\ corrupted\ tokens}{\mu} = \frac{nr}{\mu}$$

$$\text{\# tokens after UL2} = n + 2(\text{\# spans}) = n(1 + \frac{2r}{\mu})$$

A.3 Additional details for BBES results

Model / Pretrained tokens & objective	geometric shapes	navigate	snarks	understanding fables
Llama2-7B				
4B UL2	0.0390	0.5000	0.5138	0.2169
2T causal	0.0000	0.5370	0.5359	0.1693
2T causal + 4B causal	0.1003	0.4960	0.5414	0.1746
2T causal + 4B UL2	**0.1950**	0.5180	**0.5856**	**0.2222**
Llama2-13B				
2T causal	0.0080	0.5000	0.4807	0.3175

Table 3: List of selected BigBench emergent suite (BBES) and corresponding scores of every model. All results are reported with zero-shot prompting following format "Q:<question><choices>A:<answer>". For UL2 models, we add special token "[S2S]" at the front and "<sentinel_0>" right before the answer.

A.4 More text infilling examples from Llama-UL2-7B

Prompt:

He was absent <1> he was sick

Mr. John is a rich person,<1>he is not happy.

You did<1>in this text infilling task.

Model Output:

<1> from the game because he said

<1> but he is not happy because he thinks that even though he is rich,

<1> not find any errors. You did not participate in this text infilling task. Participate