Region - Educational Research and Reviews, 2024, Volume6, Issue1
DOI: 10.32629/rerr.v6i1.1619
ISSN Online: 2661-4634
ISSN Print: 2661-4626

Contextual panel conditioning and reward models in large language models

Muyuan WEN

GPT DESK PTE LTD

Abstract: Direct preference optimization (DPO) aims to match human preferences while reducing the complexity of reinforcement learning. Traditional methods such as reinforcement learning with human feedback (RLHF) first match reward models with cues and preferences, and then use reinforcement learning (RL) to find policies that maximize rewards. In contrast, DPO simplifies the process by directly optimizing the policy to satisfy preferences without explicit reward functions or RL processes. DPO is a more direct and potentially more efficient way to fine-tune a language model to remain consistent with human feedback. Additionally, OpenAI mentioned that they trained the model by imitating human ratings to help improve RLHF. The next step is to fit the model to a data set containing rich "conditions". For example, the training model generates a panel containing memories, conditions, goals, plans, and future tasks, and uses this panel for training. These conditions transform the "creative writing task" into the task of "distributing materials", reducing entropy in creative writing. Conditional reinforcement learning fine-tuning (C-RLFT) enables large language models to understand and generate human-like text, adapt to new information, and personalize responses while maintaining relevance and coherence. Future improvements include improving conditional panels using RLHF or RLAIF, iteration between datasets and models, aligning models with real-world needs, and building new base models based on 0-order optimization. These directions aim to make large language models more efficient, consistent with human preferences, and able to run in a variety of environments, including edge computing devices. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in the original language. There is no need for special content, but the length of words should match the language.

Key words: direct preference optimization; human feedback reinforcement learning; conditional panel; creative writing entropy reduction; c-RLFT training; edge computing

1 Introduction

In complex tasks such as creative writing, the use of detailed contextual panel moderated Large Language Modeling (LLM) can significantly reduce entropy. By using detailed context (e.g., character and location descriptions and the author's plans and memories) to transform high-entropy creative writing tasks of the more structured task of "writing on specified material", the model can make OpenAI use a similar technique, using manual scoring. OpenAI uses a similar technique that employs human scoring to perform reinforcement learning(RL) based on human feedback. Reinforcement Learning (RLHF) is based on human feedback. The study also incorporates the RecurrentGPT method, which uses LLM to learn from human feedback approach and to generate initial conditioning from raw text input panel. The training consists of two parts: one is a task with a global informative condition panel and the other is learning of the initial condition panel from a single task and generating these panels from a single task. The experiments show that without further reinforcement learning, the model performs well on the creative writing task and the evaluation dataset without further reinforcement learning, achieving low loss and high accuracy. This contrasts with the model trained on plain text, which did not perform as well as the model trained on plain text. If it is randomness or creativity, entropy can be intentionally introduced through human preference or other controlled processes. This balanced approach maintains the generative capacity of a large language model while significantly reducing its output.

2 Scaling method zoom

The main goal of the scaling law is to find out the computational resources, dataset size and model parameters in relation to each other. It helps to solve the questions: how big should the model be, or how much big language model requires data, or how many parameters a large language model should have, and so on. To visualize this problem, let's look at the following Figures 1 to 4 and the captions attached to them:

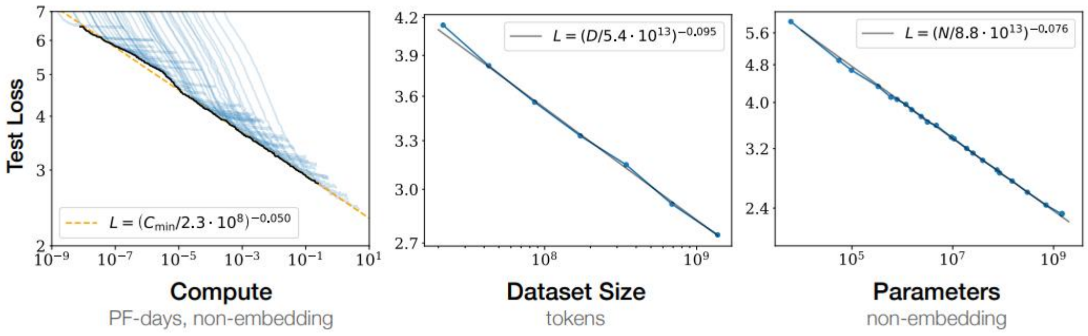

Figure 1. The performance of language modeling smoothly improves, as we increase the model size, the dataset size, and the amount of computation used for training. To achieve optimal performance, all three factors must be scaled simultaneously. When not constrained by the other two factors, experimental performance is power-law related to each individual factor. Note: The above graphic was translated from "Scaling Laws of Models". https://arxiv.org/pdf/2001.08361.pdf

The first graph (Compute): This graph shows the computational resources (measured in PF days, representing PetaFlop days, the computational power of a single bit) and the relationship between test losses. The various lines represent different model configurations or experiments, and the highlighted dashed line serves as a reference. From the graph, it can be seen that the testing loss decreases exponentially with the increase of computing resources, indicating that the improvement of computing power can significantly improve the performance of the model.

The second graph (Dataset Size): This graph illustrates how an increase in dataset size (in terms of markers) is associated with an increase in test loss. Labeled with

$$Ł = (D/5.4 \times 10^{13})^{-0.095} \qquad (1)$$

The linear line represents a specific scaling law, where "D" stands for the data set size. This trend suggests that larger datasets tend to lead to better model performance, based on a power-law relationship.

The third graph (Parameters): This graph shows the effect of the number of parameters (non-embedded) on the test loss. Once again, we observe power-law relationship between the number of parameters and the test loss: the more parameters, the test loss is lower, which means that the model performance is improved. In summary, as the model size, data set size, and the amount of computation used for training increase, the language modeling performance increases. However, for optimal performance, these factors must scale together. If any of these three factors do not increase proportionally, it can become a bottleneck, limiting the ability to scale the other factors. Most importantly is that the performance of language modeling has a power law relationship with these three key factors and balanced scaling is essential to achieve optimal results.

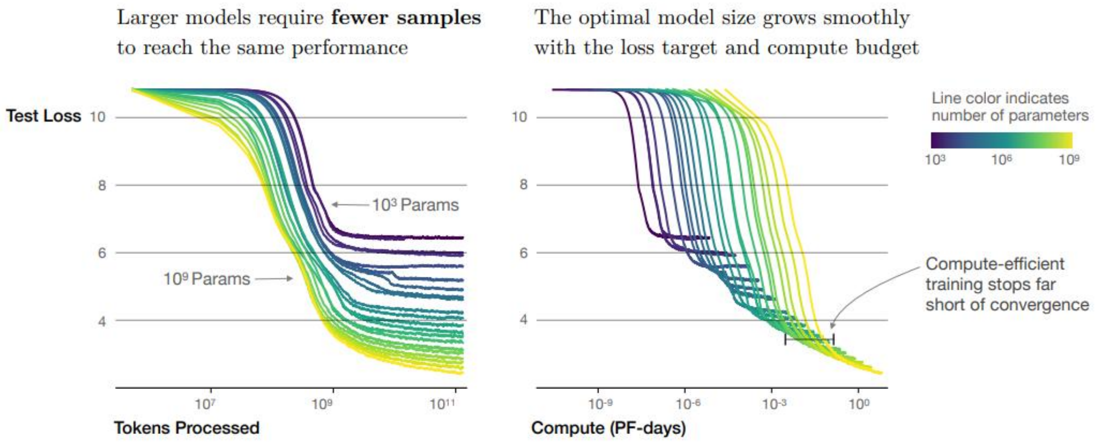

Figure 2. We show the training process for a series of language models, with model sizes range from 103 to 109 parameters (excluding embeddings). Note: The above illustrations are translated from "Scaling Laws for Models". https://arxiv.org/pdf/2001.08361.pdf

The left graph shows the relationship between the number of tokens processed (the amount of model training data metric) and the relationship between test loss. As can be seen from the figure, as the number of parameters increases (indicated by a shift of the curve to the right), the model needs to process fewer tokens to achieve a lower test loss. This suggests that larger model samples are more efficient and can use less data to achieve better performance.

The graph on the right shows the test loss for the computational resources used (in PF days). As can be seen from the figure, as the computational resources increase, the test loss decreases. It is worth noting that for each curve (model size), there is a point of diminishing returns where additional computation does not significantly reduce the test loss. This means that there is an optimal model size for the given loss target and calculation budget.

Above are the results of training runs of language models of different sizes. The larger model is more efficient, requiring fewer data samples to achieve the same level of performance. Since there exists an optimal model size with respect to the loss objective and the available computational resources, it follows that if the computational budget is limited, a larger model is not always better. The efficiency and performance of a language model is affected by the number of its parameters and the computational resources available, but needs to be balanced for optimal results.

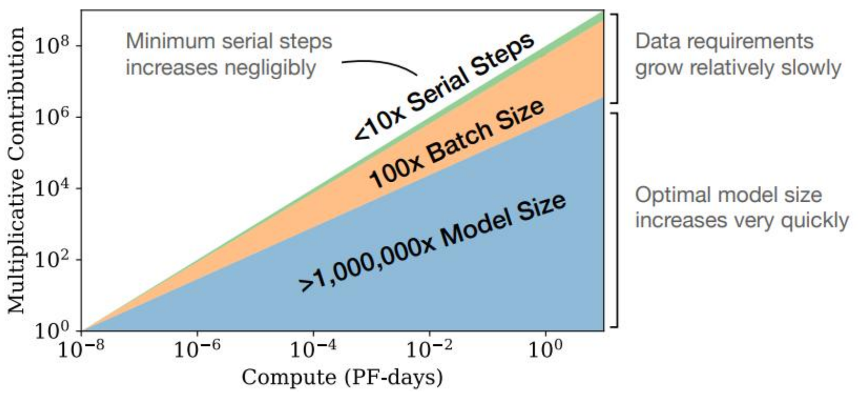

Figure 3. As the amount of available computation increases, we can choose how to divide the computational effort for training larger models, using larger batches and performing more training steps. Let's take a billion-fold increase in computation for example. For optimally computationally efficient training, most of the increase should be used to increase the model size. To avoid reusing data, only a relatively small increase in data is needed. In the increase of data, most of it can be used to increase parallelism through larger batch sizes, requiring only a very small increase in serial training time. Note: The above graphic is translated from "Scaling Laws for Models". https://arxiv.org/pdf/2001.08361.pdf

The optimal size of a model seems to increase rapidly as computational resources increase. This suggests that with more computational power, larger models can be trained more efficiently.

This double logarithmic plot shows the relationship between computational resources (in PF days) and the "multiplicative contribution", which refers to the combined benefits of the model, i.e., the impact of size, batch size, and number of serial steps on the efficiency or effectiveness of the model training process.

The impact of increasing batch size (the number of samples processed together in each step of training) grows dramatically, but it does not grow as dramatically as model size. This suggests that larger batch sizes help improve efficiency, but there are diminishing returns.

In addition, the number of serial steps (sequential processing steps) increases slightly as the amount of computation increases. This suggests that the need for serial processing will only increase slightly as computational power increases.

Finally, the relatively slow growth in data requirements means that the amount of data required does not need to grow as fast as the amount of computation or model size to keep the model efficient.

When we have more computational resources, we can choose how to allocate them: do we train larger models, use larger batch sizes, or train more steps? For more efficient training, most of the increase in resources should be used for larger models and larger batch sizes, not for training more steps, more data or longer training training time. The graphic above illustrates the strategy for a billion-fold increase in computation, emphasizing the relatively small amount of data. It is sufficient to avoid too frequent reuse of data, which can lead to excessive reuse of data and that data growth should be used primarily to increase computational power. In addition, we should increase the size of the batch by larger batch sizes rather than longer training to achieve parallelism.

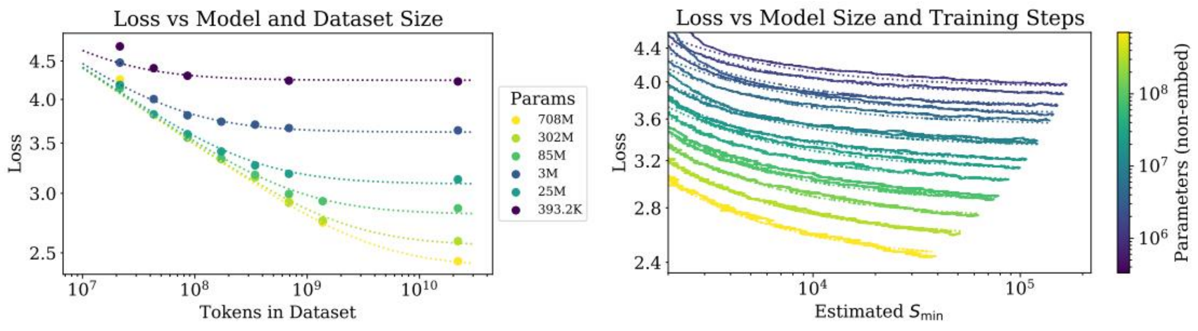

Figure 4. Left panel: Early stopping test loss according to equation (2) and loss L (N, D) varies with dataset size D and model size N. Right panel: After an initial transient phase, the learning curves for all model sizes N can be fitted with equation (3), which is parameterized by $S_{\min}$, and the number of steps when training at large batch sizes. Note: The above graphic is translated from "Scaling Laws for Models". https://arxiv.org/pdf/2001.08361.pdf

$$L(N,D) = [(\frac{N_C}{N})^{\frac{a_N}{a_D}} + \frac{D_C}{D}]^{aD} \tag{2}$$

$$L(N,S) = (\frac{N_C}{N})^{aN} + (\frac{S_C}{S_{\min}(S)})^{as} \tag{3}$$

The left panel shows the loss associated with the number of tokens in the data set loss, with different curves representing models and different parameter sizes (ranging from 393.2K to 708M parameters). As the number of tokens in the dataset increases, the loss decreases for all models. However, larger models (with more parameters) tend to have lower losses for the same dataset size, suggesting that they are more effective in utilizing the data.

The right panel then shows the loss compared to the estimated S_{min}, which may represent the minimum number of training steps required to reach a specific performance threshold. The right side the color scale indicates the number of parameters in the model. As the number of training steps increases, the loss decreases for all sizes of the model. Models with a larger number of parameters reach lower loss levels faster than smaller models.

As can be seen from the illustration, both size of the dataset and the model (in terms of the parameters) are both significant predictors of model performance (as measured by loss). Furthermore, the relationship between loss and these variables can be be described by specific equations showing how the loss decreases as the amount of data or the number of training number of steps increases, and there is a predictable pattern of how losses can be reduced. These insights come from a famous OpenAI paper that it presents some important insights into building large language models, covering how to obtain the best large language model, how to estimate the model size by calculating resource budgets and dataset size to estimate model size and loss. These rules are very useful even now and have attracted a lot of followup research, such as Deep Mind's Chinchilla.

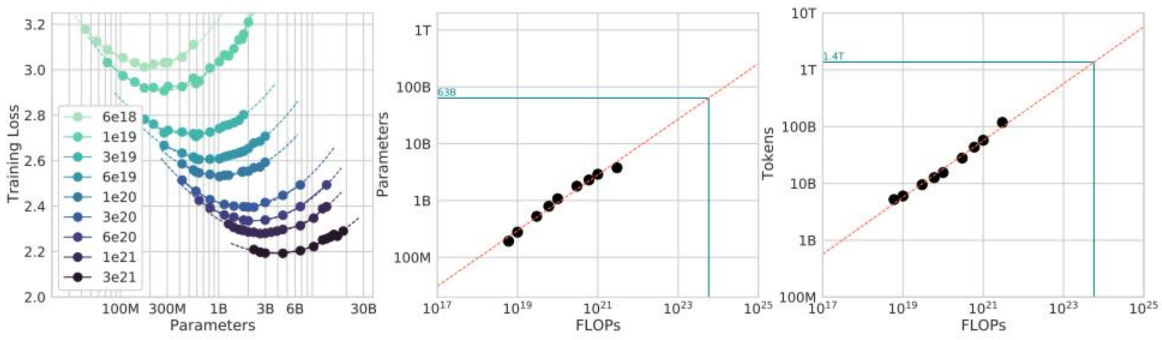

Figure 5. IsoFLOP curves. For different model sizes, we choose the number of trained tokens so that the final FLOPs are a constant number. The cosine cycle length is set to match the target FLOP counts.

We found a significant trough in the loss, meaning that for a given FLOP budget, there is an optimal model that can be train practiced (left panel). Using the locations of these troughs, we predicted the optimal model size and number of tokens for larger models (center and right graphs). In green, we show the parameters and token estimates for the optimal model practiced with Gopher's computational budget. Note: This figure is translated from "Training Computationally Optimal Large Language Models". https://arxiv.org/pdf/2203.15556.pdf

This is a paper from DeepMind where they proposed Chinchilla. This paper also makes a great contribution to the scaling law. The famous open-source LLaMa family of large language models series tracked this research and got quite positive feedback. All the benchmarks are valid, and we seem to have found the optimal values for model size, data size, and computational cost. They roughly conform to the following axiomatic formula:

$$\text{isoFLOPs} = \text{iso} - \text{FLOPs} = \text{sameFLOPs} \qquad (4)$$

3 The law of scaling does not solve all problems

Why does the scaling law not solve all problems? Scaling laws focus on training things, their goal is a single loss, and the training and validation sets are static. These things don't mean that scaling laws fail, but they make problems simpler, and sometimes even unacceptable.

In real-world requirements, we need to understand deployment costs. Our task is more than just predicting the next word. We need it to perform the task, and the task may be in the training dataset at a lower frequency. We need to perform the alignment, and we don not know the impact of the alignment on the model size/task. What effect alignment has on model size/base training samples/base on different base models. We also need to make trade offs between data size and data quality, and in some way we can either try to improve the quality of the dataset, or intentionally make more data. We need to choose the goal. In addition, once our model is deployed, the amount of real-world data will increase, and we can on top of that do things like human feedback reinforcement learning. We will also learn how these metrics differ across models of different sizes and how many problems can be solved by alignment.

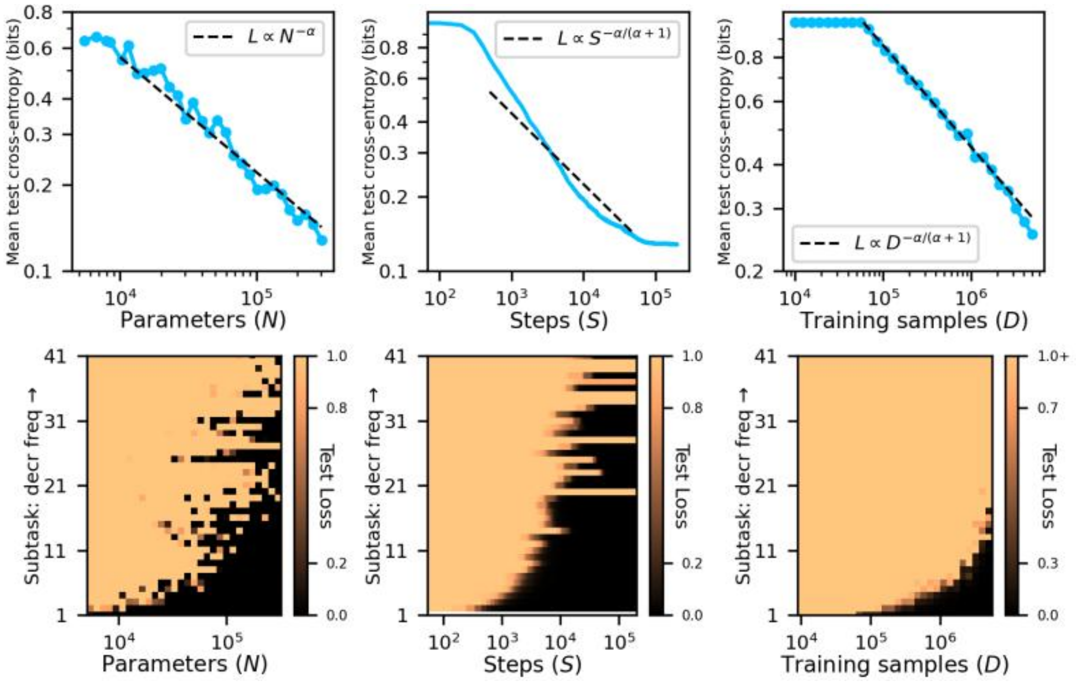

Figure 6. When trained on a multi-task sparse parity dataset, the loss of the neural network with respect to parameter N, training time S, and training samples D shows a power law scaling. D (for multiple training rounds) exhibits powerlaw scaling. Here a = 0.4, and we plot lines $\propto N^{-a}$, $S^{-a/(a+1)}$, $\propto D^{-a/(a+1)}$. Lower panel: neural scaling decomposed by subtask. The scaling behavior of a single subtask exhibits emergence, i.e., the subtasks are learned abruptly above a specific scale. The power law neural scaling of the mean test loss averages a large amount of qualitative variation in network performance (broken down by subtask decomposition), and as the loss tends to zero on more and more sub-tasks, these subtasks become more frequent. Note: This figure is translated from "Quantitative Models of Neural Scaling". https://arxiv.org/pdf/2303.13506.pdf

When the model performs poorly on low-frequency tasks but is in good agreement with the scaling law, it highlights the challenges that need to be addressed in the model alignment phase. The reason for this problem is that the base model is usually a model that is scaled at some point in time. This problem arises because the base model is often trained on datasets that are underrepresented on certain tasks. Therefore, the proficiency of the model in these areas is limited.

One possible solution is to adapt the underlying model to balance task frequency. This means intentionally modifying the training dataset to ensure a more equal representation of less frequent tasks. This approach can help the model develop a more balanced understanding and performance across a wider range of tasks.

Another approach is to train the model on a small number of tasks in addition to plain text during the basic training phase. Sample less learning involves training the model using a small number of examples for each task, teaching it to generalize from limited data. This approach may enhance the model's ability to perform well on tasks that are not heavily represented in the training data as it learns to make the best use of limited information.

Both strategies aim to enrich their training with more diverse tasks and learning scenarios, thus alleviating the limitations of the base model and improving its overall performance and adaptability.

4 Concluding remarks

The scaling law is an effective AI training method, but it has its limitations and challenges. In practical applications, we need to consider a variety of factors, such as deployment cost, task diversity, data quality and quantity, and human feedback. In order to improve the generalization ability and adaptability of the model, we need to explore different data processing and training strategies, such as adjusting the task frequency and learning with fewer samples. These strategies can help us achieve better alignment results on models of different sizes, thus improving the performance and value of AI.

Conflicts of interest

The author declares no conflicts of interest regarding the publication of this paper.

Reference

[1] Eduardo GA, Giampaolo C, Mirko DE. 2012. On the origin of long range correlations in texts. *Proceedings of the National Academy of Sciences*, 109(29): 11582-11587.

[2] Gehman S, Gururangan S, Sap M, Choi Y, Smith NA. 2020. Real toxicity prompts: evaluating neural toxic degeneration in language models. In findings of the association for computational linguistics: EMNLP2020, 3356-3369, Association for Computational Linguistics. doi.org/10.48550/arXiv.2009.11462

[3] Jared K, Sam MC, Tom H, Tom B, Benjamin C, Rewon C, Scott G, Alec R, Jeffrey W, Dario A. "Scaling laws for neural language models". ar5iv.labs.arxiv.org/html/2001.08361