LLM: from Transformers to ChatGPT[1]

Kunpeng (KZ) Zhang

Robert H. Smith School of Business
University of Maryland, College Park

kpzhang@umd.edu

ChatGPT: Optimizing Language Models for Dialogue

"We've trained a model called ChatGPT which interacts in a conversational way. The dialogue format makes it possible for ChatGPT to answer followup questions, admit its mistakes, challenge incorrect premises, and reject inappropriate requests. ChatGPT is a sibling model to InstructGPT, which is trained to follow an instruction in a prompt and provide a detailed response." - https://openai.com/blog/chatgpt.

To understand ChatGPT, let's begin with the key development in natural language understanding – language models. A language model (LM) is a probability distribution over sequences of tokens (e.g., words). Given any sequence of tokens of length *m*, an LM assigns a probability $P(w_1, w_2, \ldots, w_m)$ to the whole sequence. To come up with these probabilities, an LM trains on text corpora in one or many languages (see details here). It can then be used to generate the next token based on past tokens $P(w_i | w_1, w_2, \ldots, w_{i-1})$. Several approaches have been implemented to tackle this, including *k*-order Markov models and recurrent neural networks (RNNs) or transformers.

Markov models have several drawbacks. They become complicated when it comes to high order. Even with high order, the long-term dependencies among tokens are still not well captured. Given the success of deep learning, researchers have developed various RNN models for language understanding; in particular, they have improved models by making them deep, bidirectional, or sequence to sequence (encoder-decoder) and/or by adding attention. But training of RNN models is usually very difficult, particularly when the sequence is long. The high computational cost during training due to the infeasibility of parallelization also prevents the wide adoption and success of RNNs in NLP. Convolutional neural networks (CNNs), with their inherent capability to capture spatial and local information, are then incorporated and employed to improve language understanding. CNNs can be implemented in parallel to make the training efficient with large data. All efforts made by researchers in the past decade have focused on how to develop a model that can

[1] Note:

- All references are embedded as hyperlinks (text highlighted in blue).
- Figures in this report are from corresponding papers.
- Some statements are extracted from the original paper.

better extract features and represent samples well. Among these newer models, the transformer model (Attention Is All You Need, 06/2017) is one of the most significant innovations in deep learning and has become the first choice in many NLP and CV applications (see the architecture in Figure 1). It adopts the mechanism of (multi-head) self-attention that, compared to alternatives like recurrent networks, provides us with a more structured memory for handling long-term dependencies in text, resulting in robust transfer performance across diverse tasks. It models and learns complex relations in text, such as local or global information, as well as different levels of knowledge (low, mid, or high/abstract). Unfortunately, compared to conventional machine learning models, the performance of these models on a series of subsequent tasks (e.g., syntactic parsing, co-reference, entailment, POS, NER, etc.) is not satisfactory, at least not by a large margin. There are two possible explanations for this. First, for a certain NLP task (e.g., supervised classification), large-scale training data may be necessary for a deep learning model to learn knowledge (patterns for that task) when the model becomes large (more parameters). Collecting high-quality labeled data is costly. Second, these feature extractors via RNN or CNN might have limited representation power. Even with large data, they may still be unable to mine informative knowledge from text.

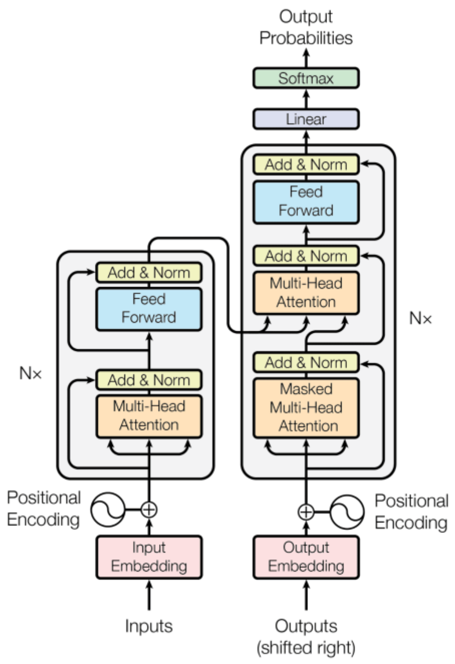

Figure 1. Transformer architecture.

The great potential of transformers has led to a boom in two different directions based on two fundamental NLP tasks: natural language understanding and natural language generation. The former includes document classification, sentiment identification, and sentence relation discrimination. The latter mainly focuses on Q&A systems (DIALOGPT, 2020), (abstractive) text summarization (PEGASUS, 2020), and machine translation (T5, 2020). One interesting trend is that transformer architecture gradually unifies all NLP applications, even those in computer vision (ViT, 2021). Most NLP studies have recently been shifting their focus to a two stage mode: pre-training (Phase I) + fine-tuning or prompt (Phase II). Pre-trained model + fine-tuning is a model that leverages the encoder component of transformers. The best representative is BERT (Bidirectional Encoder Representations from Transformers, 10/2018), which pre-trains the encoder using large-scale unlabeled data under two tasks, MLM (masked LM) and NSP (next sentence prediction), and then fine-tunes using labeled data for different downstream tasks (see Figure 2). The pioneering tech firm driving and promoting BERT is Google. On the other hand, pre-trained model + zero/few shot prompt is an autoregressive language model (unidirectionally predicting the next word from left to right) that only uses the decoder part of transformers (see the difference from BERT in Figure 3a and its architecture in Figure 3b). The most successful model of this type is the GPT (Generative Pre-Training, 06/2018) series developed by OpenAI. OpenAI argues that pre-training plus fine-tuning might create overfitting problems because models are designed to be large to absorb information during pre-training but are then fine-tuned on very narrow task distributions. This makes larger models not necessarily generalize better out-of-distribution. These two development directions seem similar in the sense that both take sequence data as input and use transformer blocks to pre-train the language model. But in fact, they have significant differences (as we will elaborate in detail later), especially GPT is likely to be towards the future of AGI (Artificial General Intelligence).

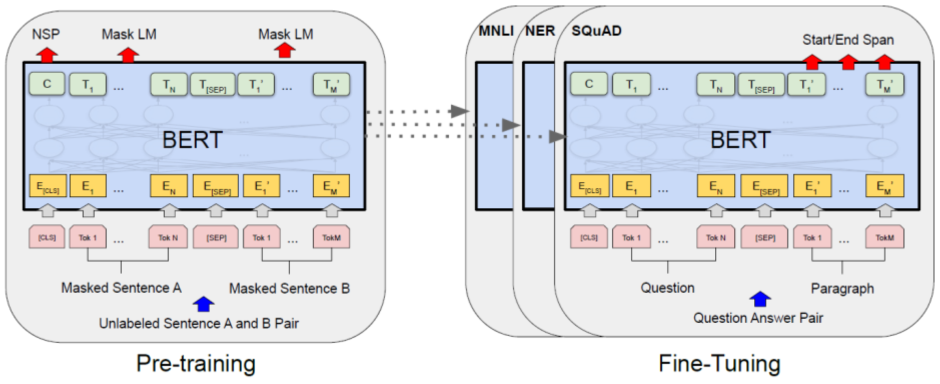

Figure 2. Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different downstream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g., separating questions/answers).

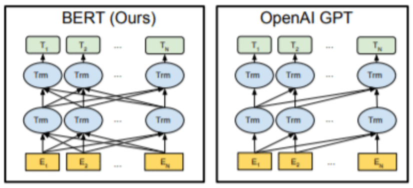

Figure 3a. Difference between BERT and GPT.

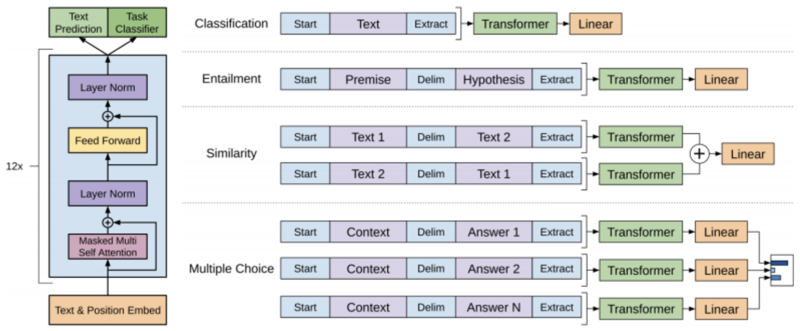

Figure 3b. (left) Transformer architecture and training objectives used in this work. (right) Input transformations for fine-tuning on different tasks.

BERT achieves superior performance over GPT on most NLP tasks, which makes BERT advocates firmly believe that this is the right path to real intelligence. Part of the reason, as seen in the experiments conducted in the BERT paper, is that the volume of data used to train BERT is approximately four times that used to train GPT. This naturally brings us to another question: what happens as the scale of these large-scale language models (LLMs) continues to increase? Based on the scaling laws, the loss of the pre-trained models on the hold-out data decreases when we expand the training set, increase the complexity of models, or train with more epochs. Therefore, OpenAI decided to make an attempt. One suspicion about the lack of generalization observed in current systems is that it is to a large extent driven by the prevalence of single task training on single domain datasets. Progress towards robust systems with current architectures is therefore likely to require training and measuring performance on a wide range of domains and tasks. "Multitask learning" and "meta-learning" might be a way to help. In GPT-2 (02/2019), OpenAI continues the architecture of GPT to pre-train a language model but performs downstream tasks in a zero-shot setting – without any parameter or architecture modification. One primary challenge in GPT-2 is that every downstream task cannot introduce new tokens that do not exist in the training set. Thus, GPT-2 leverages the idea of a prompt to formulate input for each task. For example, a translation training example can be written as the sequence (translate to French, English text, French text). Likewise, a reading comprehension training example can be written as (answer the question, document, question, answer). Why can the pre-trained models without any model/architecture adjustment understand these prompts? It could be that (i) the pre-trained model is powerful enough due to large-scale training data or (ii) the training data might contain prompt-related information that has already been learned by the model. Figure 4 shows the performance of GPT-2 models (of different sizes) on different NLP tasks. It achieves promising, competitive, or state-of-the-art results, depending on the task. In addition, this outcome further proves the scaling laws, which motivates the development of GPT-3.

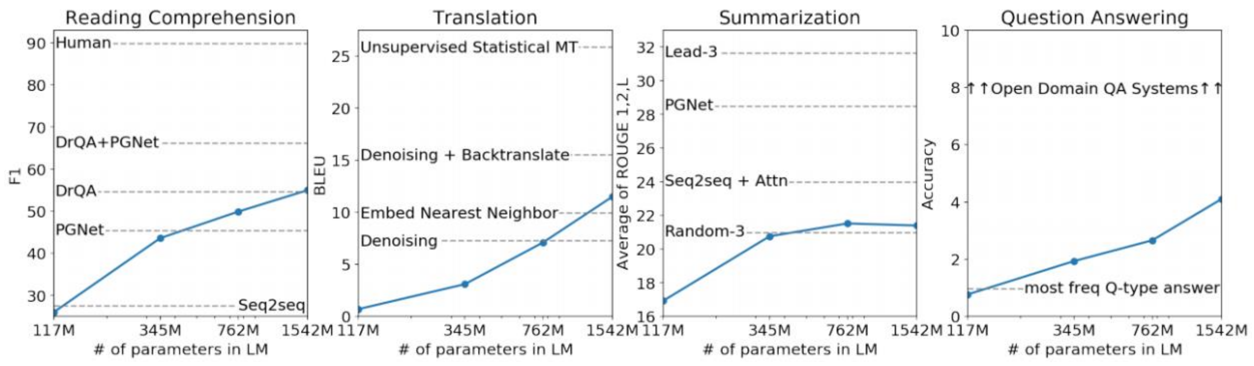

Figure 4. Zero-shot task performance of GPT-2 on four tasks.

We all know that humans do not require large supervised datasets to learn most language tasks – a brief directive in natural language or at most a tiny number of demonstrations is often sufficient to enable a human to perform a new task to at least a reasonable degree of competence. In addition, when using GPT-2, humans need to adapt to the model by providing good prompts to obtain high-quality response generation. It is extremely desirable to make models that, like a human, seamlessly mix together or switch between many tasks and skills as well as understand human instructions in general. In GPT-3, OpenAI uses what it calls "in-context learning," using the text input of a pre-trained language model as a form of task specification: the model is conditioned on a natural language instruction (i.e., prompt) and/or a few demonstrations of the task and is then expected to complete further instances of the task simply by predicting what comes next. One key difference from traditional fine-tuning of models is that no gradient updates are performed. More importantly, OpenAI trains a 175 billion parameter autoregressive language model and achieves significant improvement over state-of-the-art fine-tuned models under the zero/one/few shot setting in most NLP tasks (see Figure 5). Another surprising finding is that smaller GPT-3 (i.e., 2.7B parameters) outperforms the SOTA 17B parameter Turing-NLG in some tasks.

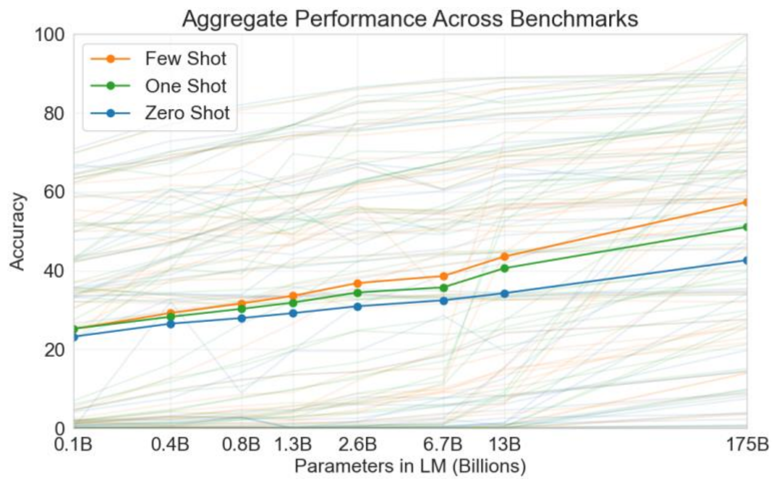

Figure 5. Aggregate performance for all 42 accuracy-denominated benchmarks.

Note that GPT-3 is still unable to understand text like humans do (e.g., instructions) but relies on prompts (this does not conform to human usage habits). This leads to a problem where humans need to rack their brains to figure out how to design an appropriate prompt. This problem has been understood and addressed for the first time by InstructGPT (2022), which intends to tackle the following major challenge: making language models bigger does not inherently make them better at following the user's intent. An LLM can generate outputs that do not align with the user's desire, e.g., responses that are untruthful, toxic, or simply not helpful. To resolve this issue, InstructGPT focuses on fine-tuning with human feedback (it's a bit of a slap in the face - go to the opposite path: fine-tuning). Specifically, this fine-tuning consists of three steps (see Figure 6).

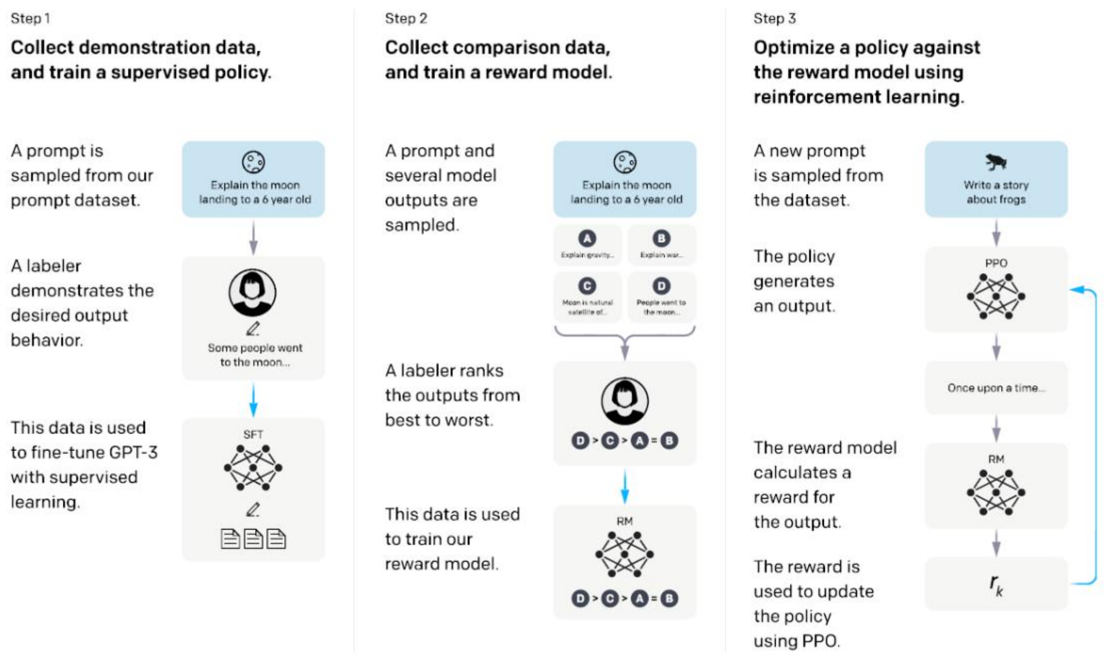
Step 1
Collect demonstration data, and train a supervised policy.
A prompt is sampled from our prompt dataset.
Explain the moon landing to a 6 year old
A labeler demonstrates the desired output behavior.
Some people went to the moon...
This data is used to fine-tune GPT-3 with supervised learning.
SFT
Step 2
Collect comparison data, and train a reward model.
A prompt and several model outputs are sampled.
Explain the moon landing to a 6 year old
A
Explain gravity...
B
Explain war...
C
Moon is natural satellite of...
D
People went to the moon...
A labeler ranks the outputs from best to worst.
D > C > A = B
This data is used to train our reward model.
RM
D > C > A = B
Step 3
Optimize a policy against the reward model using reinforcement learning.
A new prompt is sampled from the dataset.
Write a story about frogs
The policy generates an output.
PPO
Once upon a time...
The reward model calculates a reward for the output.
RM
The reward is used to update the policy using PPO.
r_k

Figure 6. A diagram illustrating the three steps of InstructGPT: (1) supervised fine-tuning (SFT), (2) reward model (RM) training, and (3) reinforcement learning via proximal policy optimization (PPO). Blue arrows indicate that this data is used to train one of our models. In Step 2, boxes A-D are samples from our models that get ranked by labelers.

(1) OpenAI collects a dataset of human-written demonstrations of the desired output behavior on (mostly English) prompts submitted to the OpenAI API and some labeler-written prompts. It then uses this dataset (13k) to train our supervised learning baselines (this is supervised fine-tuning (SFT)).

(2) OpenAI collects a dataset of human-labeled comparisons between outputs from SFT on a larger set of prompts. It then trains a reward model (RM) on this dataset (33k) to predict which output the labelers would prefer. Note that here OpenAI uses 6B SFT as its RM model. The loss function is shown below.

$$\operatorname{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta\left(x,y_w\right) - r_\theta\left(x,y_l\right)\right)\right)\right],$$

where $r_\theta(x, y)$ is the scalar output of the reward model for prompt x and completion y with parameters θ, y_w is the preferred completion out of the pair of y_w and y_l, and D is the dataset of human comparisons.

(3) InstructGPT further fine-tunes the SFT using PPO. Given the prompt and response, it produces a reward determined by the reward model and ends the episode. The PPO is trained using 31k prompts that are only from the OpenAI API. The objective is as follows.

$$\begin{aligned}\operatorname{objective}(\phi) = &E_{(x,y)\sim D_{\pi_\phi^{\mathrm{RL}}}}\left[r_\theta(x,y) - \beta\log\left(\pi_\phi^{\mathrm{RL}}(y\mid x)/\pi^{\mathrm{SFT}}(y\mid x)\right)\right] + \\ &\gamma E_{x\sim D_{\text{pretrain}}}\left[\log(\pi_\phi^{\mathrm{RL}}(x))\right]\end{aligned},$$

where $\pi_\emptyset^{RL}$ is the learned RL policy (i.e., the final InstructGPT model), π^{SFT} is the supervised trained model, and $D_{pretrain}$is the pre-training distribution. The second term is the KL penalty from the SFT model to mitigate over-optimization of the RM model. The third term ensures that the learned model does not get too away from the pre-trained one and lose the general knowledge in the pre-trained model.

It is worth noting that InstructGPT has done an amazing job in engineering implementation, addressing issues such as working with labelers (frequent interaction with contractors hired by OpenAI), picking prompts from those provided by users to the OpenAI API, choosing RM models, setting the number of responses to rank, and cleaning data (including ascertaining whether data from a labeler is normal and how to adjust it if not). However, I also have several questions about InstructGPT. (i) Why does OpenAI not obtain more labeled data for SFT, which might outperform the currently used SFT with RL and RM? (ii) OpenAI aimed to obtain improvements in InstructGPT's truthfulness, reduce toxic output generation, and increase helpfulness. But why does it never evaluate helpfulness?

Given the promise and incredible performance of ChatGPT, as well as its requirement of intensive computational resources for modeling training, it will have significant impacts on our lives and especially on NLP research. Here is a list of my personal thoughts.

- The first and most discussed is that personally I do believe that some NLP research, particularly at the intermediate level (e.g., generating reports, summarizing documents, classifying documents, etc.), can be easily dominated by ChatGPT. I am sure this will be even more clearly manifested when GPT-4 or a higher version of GPT is released later. I also expect that more NLP tasks (previously tackled in an unsatisfactory manner) will be explored and that GPT will achieve good performance.

- Training an LLM requires extremely large computational resources, which cannot be provided by any groups in universities and is even difficult for specialized institutes (e.g., OpenAI, DeepMind). I would expect to see collaborations across institutes to train and share LLMs.
- There will be debate and further investigation of two directions in LLMs: pre-trained + fine-tuning or pre-trained + in-context learning/instruct. Regarding ChatGPT, I think large-scale pre-training plus human in the loop demonstrates great potential to make language models more aligned with human language. The issue of how to involve human feedback is still unexplored: current pre-training is completely unsupervised without any human intervention, but there may be better alternatives to reward models that evaluate responses.
- Knowledge learned and stored in LLMs needs updating. Given the number of parameters in an LLM, it is challenging to precisely locate relevant parameters to update certain knowledge without changing irrelevant parameters. Simply using a fine-tuning strategy is not appropriate because the model may then update certain knowledge but meanwhile forget other knowledge, which would accordingly hurt downstream tasks.
- ChatGPT is still weak in addressing questions that require a strong capability of logic inference, such as advanced math. This is not because ChatGPT does not have this knowledge but because we do not yet know how to guide it to find and extract such knowledge. Such tasks are very dependent on the instruction (prompts) provided by a human. Lots of research has been done in this area, particularly in regard to CoT (chain of thought prompt).
- LLM is currently still a language model, but its algorithmic and engineering frameworks are fascinating. I believe these frameworks will be expanded to other types of data (structured and unstructured), e.g., image, audio, and video. We may eventually see a unified model and reach the destination of AGI. That being said, we will see many applications and algorithmic breakthroughs in non-text domains.