Developing Large Language Models from Scratch: A Comprehensive Guide

Michael Borck

2024-05-20

Abstract

The development of large language models (LLMs) from scratch is a complex and resource-intensive endeavour, often perceived as feasible only for specialised AI research institutions. However, with the rising interest in custom LLMs for specific business and organisational applications, understanding the comprehensive process of building these models has become increasingly relevant. This paper provides a detailed guide on the critical steps involved in developing a foundational LLM, drawing insights from models such as GPT-3, LLaMA, and Falcon. Key aspects covered include data curation, where the quality and diversity of the training data are emphasised; model architecture, focusing on the transformative role of transformer networks and their various configurations; training at scale, which addresses the technical challenges and solutions for efficient large-scale model training; and evaluation, detailing methodologies for assessing model performance across different tasks. This guide aims to equip researchers and practitioners with the knowledge necessary to navigate the intricacies of LLM development and make informed decisions about when such an investment is justified.

## Introduction

While prompt engineering, Retrieval-Augmented Generation (RAG), and fine-tuning approaches can handle the majority of LLM use cases, there are situations where it may make sense to build a large language model (LLM) from scratch. This white paper reviews the key aspects of developing a foundational LLM based on the experiences with models such as GPT-3, LLaMA, Falcon, and others.

Historically, training large-scale language models (10B+ parameters) was an esoteric activity reserved for AI researchers. However, post-ChatGPT, the environment has changed, with businesses and organisations showing interest in developing their custom LLMs (“BloombergGPT,” n.d.). Although this is unnecessary for over 99% of LLM applications, understanding the process and knowing when it makes sense to build such models is beneficial.

## Financial Considerations

Before diving into the technical aspects of LLM development, it is crucial to consider the financial costs. Meta’s LLaMA 2 models required about 180,000 GPU hours to train its 7B parameter model and 1,700,000 GPU hours to train the 70B model (“Llama 2,” n.d.). Translating this into commercial cloud computing costs, an NVIDIA A100 GPU, used for training LLaMA 2 models, costs around $1–2 per GPU hour. Therefore, a ~10B parameter model costs about $150,000 to train, while a ~100B parameter model costs approximately $1,500,000.

Alternatively, one could buy the GPUs instead of renting them. The cost of training would then include the price of the A100 GPUs and the marginal energy costs for model training. An A100 GPU costs about $10,000, and a cluster of 1000 GPUs would amount to approximately $10,000,000. Additionally, with an energy cost of about $100 per megawatt hour and training requiring around 1,000 megawatt hours, the marginal energy cost for training a 100B parameter model is about $100,000 (“LLM Energy Costs,” n.d.).

These costs do not include funding a team of ML engineers, data engineers, data scientists, and others needed for model development, which can easily amount to $1,000,000. Hence, training an LLM from scratch is a massive investment, justifiable only if there is a significant potential upside not achievable via prompt engineering or fine-tuning existing models.

## Key Steps in Developing an LLM

The development process of an LLM consists of four key steps:

1. **Data Curation**
2. **Model Architecture**
3. **Training at Scale**
4. **Evaluation**

### Step 1: Data Curation

The quality of an LLM is driven by the quality of its training data. Popular base models have training sets of substantial size:

* GPT-3 (175B): 0.5T tokens (Brown et al. 2020)
* LLaMA (70B): 2T tokens (“Llama 2,” n.d.)
* Falcon (180B): 3.5T tokens (“Falcon 180b,” n.d.)

A trillion words of text equate to about a million novels or a billion news articles. The internet is the most common data source for LLMs, encompassing web pages, books, scientific articles, code bases, and conversational data. Open datasets like Common Crawl, The Pile (Gao et al. 2020), and others on Hugging Face’s platform are often used. Alternatively, researchers can have an existing LLM generate a high-quality training corpus, as done with Alpaca (“Alpaca,” n.d.).

Preparing the data involves several preprocessing steps:

* **Quality Filtering:** Removing low-quality text using classifier-based or heuristic-based approaches (Zhao et al. 2023).
* **De-duplication:** Ensuring the same text does not bias the model or disrupt the training process (Lee et al. 2021).
* **Privacy Redaction:** Removing personally identifiable information to avoid unintended exposure of sensitive data.
* **Tokenisation:** Translating text into numerical form using algorithms like byte-pair encoding (BPE) (Sennrich, Haddow, and Birch 2015).

### Step 2: Model Architecture

Transformers are the state-of-the-art approach for language modeling (Vaswani et al. 2017). They consist of two key modules: an encoder and a decoder. These can be standalone or combined, resulting in three types of transformers:

1. **Encoder-only:** Suitable for tasks requiring input understanding, such as text classification. Example: BERT (Devlin et al. 2018).
2. **Decoder-only:** Used for text generation tasks, like GPT-3, LLaMA, and Falcon (Zhao et al. 2023, huggingface\_nlp\_course).
3. **Encoder-Decoder:** Combines both modules for tasks like translation and summarisation. Example: BART (Lewis et al. 2019).

Other design choices include:

* **Residual Connections:** Improve training stability and performance by allowing intermediate values to bypass hidden layers (He et al. 2016).
* **Layer Normalisation:** Speeds up training and makes it more stable (Ba, Kiros, and Hinton 2016).
* **Activation Functions:** Introduce non-linearities to capture complex mappings (e.g., GeLU, ReLU) (Zhao et al. 2023).
* **Position Embeddings:** Capture token positions in the model’s representation of text (Vaswani et al. 2017, shaw2018self).

### Step 3: Training at Scale

LLMs are trained via self-supervised learning, predicting the final token in a sequence based on the preceding ones. Scaling up model training involves techniques like mixed precision training, 3D parallelism, and the sero Redundancy Optimiser (seRO).

**Training Techniques:**

* **Mixed Precision Training:** Uses both 32-bit and 16-bit floating-point data types to reduce computational costs (Zhao et al. 2023, nvidia\_mixed\_precision).
* **Parallelisation:** Distributes training across multiple GPUs using pipeline, model, and data parallelism (Zhao et al. 2023).
* **sero Redundancy Optimiser (seRO):** Reduces data redundancy in the optimiser state, gradient, or parameter partitioning (Zhao et al. 2023).

**Training Stability:**

* **Checkpointing:** Saves model artifacts to resume training after failures (Zhao et al. 2023).
* **Weight Decay:** Penalises large parameter values to avoid overfitting (“Weight Decay,” n.d.).
* **Gradient Clipping:** Rescales gradients to prevent the exploding gradient problem (“What Is Gradient Clipping,” n.d.).

**Hyperparameters:**

* **Batch Sise:** Number of samples processed before updating parameters.
* **Learning Rate:** Controls optimisation step size.
* **Optimiser:** Defines how to update model parameters.
* **Dropout:** zeros out a portion of parameters at random to avoid over fitting (“Andrej Karpathy Lecture,” n.d.).

### Step 4: Evaluation

Model development is iterative, involving repeated evaluation of model performance on a set of tasks. Common benchmarks include:

* **ARC:** Grade-school level multiple-choice science questions (Clark et al. 2018).
* **Hellaswag:** Commonsense natural language inference tasks (Zellers et al. 2019).
* **MMLU:** World knowledge and problem-solving tasks (Hendrycks et al. 2020).
* **TruthfulQA:** Evaluates model truthfulness against common misconceptions (Lin, Hilton, and Evans 2021).

Evaluations can be done using prompt templates, human evaluation, or NLP metrics like Perplexity, BLEU, or ROGUE scores. An auxiliary fine-tuned LLM can also be used for comparisons, as demonstrated by GPT-judge (Lin, Hilton, and Evans 2021).

## Conclusion

Building an LLM from scratch involves significant technical and financial investments. However, understanding this process can help determine when such an endeavour is justified and what the key considerations are. Whether leveraging existing models or building new ones, the goal is to develop an AI solution that effectively addresses the target application.

### References

“Alpaca.” n.d.

“Andrej Karpathy Lecture.” n.d.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. “Layer Normalization.” *arXiv Preprint arXiv:1607.06450*.

“BloombergGPT.” n.d.

Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, et al. 2020. “Language Models Are Few-Shot Learners.” *arXiv Preprint arXiv:2005.14165*.

Clark, Peter, Stephen Cowhey, Oren Etzioni, Tushar Khot, Bhavana Mishra, Kyle Richardson, Carissa Schoenick, and Oyvind Tafjord. 2018. “Think You Have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge.” *arXiv Preprint arXiv:1803.05457*.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. “BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding.” *arXiv Preprint arXiv:1810.04805*.

“Falcon 180b.” n.d.

Gao, Leo, Stella Biderman, Sidney Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, et al. 2020. “The Pile: An 800GB Dataset of Diverse Text for Language Modeling.” *arXiv Preprint arXiv:2101.00027*.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. “Deep Residual Learning for Image Recognition.” *arXiv Preprint arXiv:1603.05027*.

Hendrycks, Dan, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2020. “Measuring Massive Multitask Language Understanding.” *arXiv Preprint arXiv:2009.03300*.

Lee, Kenton, Patrick Roit, Yasaman Razeghi, Yi Xu, Sewon Min, Semih Yavuz Mukherjee, Colin Raffel, Fabio Petroni, Luke Zettlemoyer, and Mike Lewis. 2021. “Deduplicating Training Data Makes Language Models Better.” *arXiv Preprint arXiv:2112.11446*.

Lewis, Mike, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. “BART: Denoising Sequence-to-Sequence Pre-Training for Natural Language Generation, Translation, and Comprehension.” *arXiv Preprint arXiv:1910.13461*.

Lin, Stephanie, Jacob Hilton, and Owain Evans. 2021. “TruthfulQA: Measuring How Models Mimic Human Falsehoods.” *arXiv Preprint arXiv:2109.07958*.

“Llama 2.” n.d.

“LLM Energy Costs.” n.d.

Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2015. “Neural Machine Translation of Rare Words with Subword Units.” *arXiv Preprint arXiv:1508.07909*.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. “Attention Is All You Need.” *arXiv Preprint arXiv:1706.03762*.

“Weight Decay.” n.d.

“What Is Gradient Clipping.” n.d.

Zellers, Rowan, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. “HellaSwag: Can a Machine Really Finish Your Sentence?” *arXiv Preprint arXiv:1905.07830*.

Zhao, Wayne Xin, Yuan Zhang, Yichang Zou, and Jian-Yun Nie Chen. 2023. “A Survey of Large Language Models.” *arXiv Preprint arXiv:2303.18223*.