Building LLMs: Key Insights from Stanford's CS229 Lecture

Kshitij · Follow

5 min read · Oct 18, 2024

A recent Stanford University lecture (CS229) on building LLMs sheds light on the core concepts and challenges behind these advanced models. Here's a summary of my key takeaways from the lecture, focusing on the practical aspects of training and optimizing LLMs.

The 5 Major Components of Building LLMs

1. **Architecture**
2. **Training Algorithm/Loss**
3. **Data**
4. **Evaluation**

5. **System**

While architecture and training algorithms are mostly used in academia, the focus of this lecture was on the last three components — data, evaluation, and system — which are critical when deploying LLMs in the real world.

Understanding Language Modeling

Language models predict the probability of sequences of words, essentially determining how likely a given sentence is to be naturally uttered by humans or found online. For example, sentences like "The mouse ate the cheese" would have a much higher probability than a sentence like "The cheese ate the mouse."

Autoregressive Models

Autoregressive language models predict each word in a sequence by considering the previous ones, using formulas to break down the probabilities step by step. This process includes tokenization (breaking the input into tokens), passing it through a model, predicting the next word, sampling and then de-tokenizing the output for interpretation.

Cross-Entropy Loss and Tokenization

One of the core concepts in LLM training is **cross-entropy loss**, which measures how well the model's predictions align with the actual data. Lower cross-entropy means better performance.

Tokenization

One important aspect of LLMs is how they break down text into tokens. The lecture explains that tokenization is more general than simply splitting words; it involves creating smaller sequences, sometimes even down to characters. Tokens represent common sub-sequences, which leads to more efficient model training and inference.

Byte-Pair Encoding (BPE) is a popular tokenization technique used in LLMs. The steps involved in BPE are:

1. Take a large corpus of text.
2. Start with one token per character.
3. Merge the most common pairs of tokens into a single token.

4. Repeat the merging process until the desired vocabulary size is reached

LLM Evaluation: Perplexity and Benchmarks

Evaluating LLMs is critical to understanding their performance. **Perplexity** measures how confident a model is in predicting the next word. If a model has low perplexity, it means it is more certain about its predictions. For example, if the model is certain about the next token, its perplexity would be 1. If it hesitates between multiple options, the perplexity increases.

In addition to perplexity, frameworks like **HELM (Holistic Evaluation of Language Models)** and the **Hugging Face Open LLM Leaderboard** aggregate performance across various NLP tasks, offering a comprehensive view of how models stack up against one another.

Data for Training LLMs

Training large language models requires massive amounts of data, and much of this data is sourced from the web. When talking about extracting internet data, Google has a collection of around 250 billion pages, and web crawlers like Common Crawl are used to crawl these pages, resulting in a dataset that exceeds 1 petabyte in size. This raw data then undergoes some filtering processes to ensure high-quality input for the models.

Scaling Laws and Optimizing Model Training

The lecture also touched on **scaling laws**, which describe how increasing compute power, data size, and model parameters can reduce training loss. This means that with more resources, we can expect better performance. However, these decisions need to be carefully calibrated. For instance, with 10,000 GPUs available for a month, what's the best model to train?

Let's answer this question with an example: training **Transformers** and **LSTMs** at different scales, researchers can fit the scaling law to predict which model would perform best with increased compute. From the below graph, we can see that the Transformers outperform LSTMs as compute scales up.

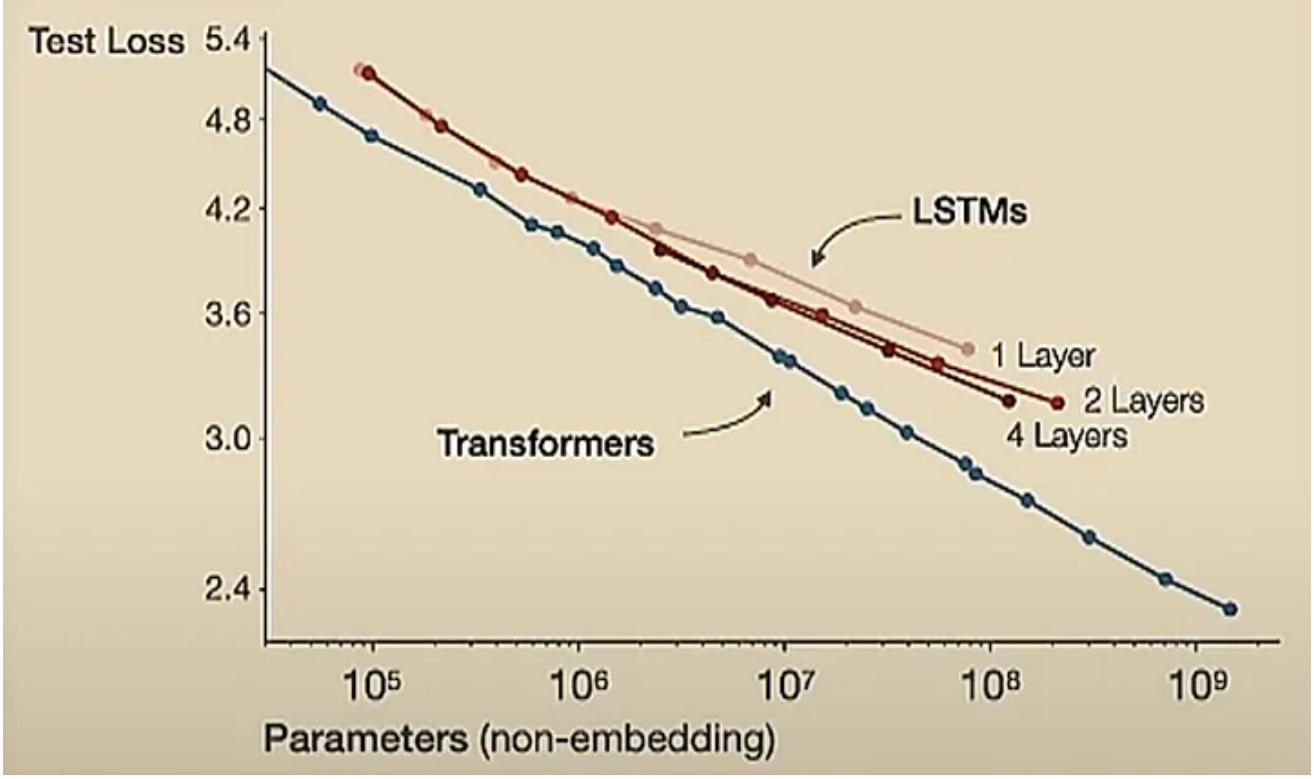

Image taken from the lecture slide

Training State-of-the-Art (SOTA) Models

The lecture discussed the challenges of training **state-of-the-art (SOTA) models** like **LLaMA3 400B**. Training such a model requires enormous compute resources and is incredibly costly. The lecturer provides specific benchmarks and cost figures to illustrate just how expensive it is to train these large-scale models.

Post-Training and Fine-Tuning

In the lecture, the concept of **post-training** is introduced, which involves fine-tuning a pre-trained large language model (LLM) to tailor it to specific tasks or user needs. This is particularly relevant for AI assistance, where models need to generate more accurate and context-specific responses.

Supervised Fine-Tuning (SFT)

The idea of SFT is to take a pre-trained LLM and fine-tune it with data that contains the desired answers for specific tasks. To collect this fine-tuning data, humans are often asked to provide answers, which was a crucial part of the development from **GPT-3** to **ChatGPT**. However, human data collection is both slow and expensive, which presents a challenge when scaling up these models.

ALPACA 7B: Scaling Data Collection with LLMs

The lecturer presents **ALPACA** as an innovative solution to this problem of slow and expensive human data collection. The key idea behind ALPACA was to use LLMs themselves to generate large-scale, human-like data for supervised fine-tuning, reducing the reliance on manual human input.

Here's how it worked:

1. **Initial Data Collection**: They started by collecting 175 question-answer pairs provided by humans.
2. **LLM-Generated Data**: Using **TextDavinci003**, they generated 52,000 similar question-answer pairs.
3. **Fine-Tuning**: The **LLAMA 7-billion** model was then fine-tuned using these 52,000 LLM-generated question-answer pairs, resulting in the **ALPACA 7-billion** model.

Reinforcement Learning from Human Feedback (RLHF)

Another fine-tuning method is Reinforcement Learning from Human Feedback (RLHF), which overcomes some limitations of Supervised Fine-Tuning (SFT). While SFT involves cloning human behavior, RLHF focuses on optimizing model outputs based on human preferences.
RLHF uses a process where a model generates two possible answers, and human labelers choose which one they prefer. The model is then fine-tuned using reinforcement learning algorithms.

Conclusion

Building large language models is a complex process that requires a balance of compute, data, and evaluation metrics. As LLMs continue to evolve, these insights from Stanford's lecture provide a glimpse into the future of AI and language modeling.

References

1. https://www.youtube.com/watch?v=9vM4p9NN0Ts

Llm Training Stanford Evaluation