Lecture 22: Scaling Up to GPT-3 and Beyond

The Era of Few-Shot Learning

Models of Language and Conversation

Week 7

Today's Journey

1. **GPT-2** - 10x scale-up, zero-shot multitask learning
2. **GPT-3** - 100x scale-up, few-shot learning emerges
3. **Scaling Laws** - Why bigger is (predictably) better
4. **RLHF & ChatGPT** - Aligning LLMs with human preferences
5. **Modern Landscape** - Open vs closed models
6. **Hands-on Examples** - Prompting techniques in practice

GPT-2: The Unexpected Leap

Discussion

What happens when we scale up GPT by 10x?

GPT (2018):

- 117M parameters
- 5B tokens training
- BooksCorpus
- Requires fine-tuning

GPT-2 (2019):

- 1.5B parameters (13x larger)
- 40GB text (8x more data)
- WebText dataset

The WebText Dataset

How GPT-2 was trained:

WebText Creation

1. Scrape all outbound links from Reddit with >=3 karma
2. Filter for quality and diversity
3. Remove Wikipedia (to avoid test set contamination)
4. Result: 40GB of text, 8 million documents

Why Reddit links?

- Community curation (karma = quality signal)
- Diverse topics and writing styles
- Web-scale variety
- Human-filtered content

WebText: Concrete Examples

What kinds of documents were included:

```
Reddit post (3+ karma): "Check out this great article about climate science"
 -> Linked article scraped and included in training

Reddit post (3+ karma): "Here's an amazing tutorial on machine learning"
 -> Tutorial content included in training

Reddit post (2 karma): "Random blog post"
 -> EXCLUDED (below karma threshold)
```

Sample document types in WebText:

Source Type	Example	Why Included
News articles	NYT, BBC	High quality journalism

GPT-2 Model Sizes

Four model sizes released progressively:

Model	Parameters	Layers	Hidden Size
Small	117M	12	768
Medium	345M	24	1024
Large	762M	36	1280
XL	1.5B	48	1600

Staged release strategy:

- Feb 2019: Released small model (117M)
- May 2019: Medium model (345M)
- Aug 2019: Large model (762M)

Zero-Shot Task Transfer

The surprising finding: GPT-2 can perform tasks without fine-tuning!

Zero-Shot Prompting Examples

Translation:

```
English: I love machine learning
French:
```

GPT-2 completes: "J'aime l'apprentissage automatique"

Question Answering:

```
Answer the question:
Q: What is the capital of France?
A:
```

Zero-Shot: How It Works

GPT-2 saw similar patterns during pre-training:

```
Example from training data (hypothetical):

"...The meeting was held in Berlin. The German chancellor...
Later that day in Tokyo, Japanese officials...

Quick Summary: Leaders from Germany and Japan met to discuss..."
```

At inference time:

```
User prompt: "[Article about climate conference]

TL;DR:"

GPT-2 thinks: "I've seen 'TL;DR:' followed by summaries thousands
```

GPT-2 Performance

Zero-shot results on various benchmarks:

Task	Metric	Fine-tuned SOTA	GPT-2 Zero-shot
Translation (En->Fr)	BLEU	45.6	11.5
Summarization	ROUGE	40.2	29.3
Question Answering	Accuracy	89.4	63.1
Reading Comprehension	F1	91.8	55.0

Promising:

- Works without fine-tuning
- Generalizes across tasks
- Improves with scale

Text Generation Quality

GPT-2 Generated Text Sample

Prompt: "In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains."

GPT-2 continues:

"Even more surprising to the researchers was the fact that the unicorns spoke perfect English. The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science..."

Observations:

- Coherent and fluent
- Maintains context and style
- Completely fabricated "facts"

GPT-3: The 175B Parameter Model

Discussion

What happens when we scale up another 100x?

Model size comparison:

Model	Parameters	Relative Size
GPT-1	117M	1x
GPT-2	1.5B	13x
GPT-3	175B	1,500x

Key insight: GPT-3 is so large that new capabilities *emerge* that weren't present in smaller models!

Reference: Brown et al. (2020) - "Language Models are Few-Shot Learners"

GPT-3 Model Specifications

Component	Value
Layers	96
Hidden size (d_model)	12,288
Attention heads	96
Context window	2048 tokens
Training tokens	300 billion
Training data	570GB (filtered)
Training compute	~3,640 petaflop-days
Estimated training cost	~$4.6M

Scale

GPT-3 Training Data

Training corpus composition:

Dataset	Tokens	Weight in Training
Common Crawl (filtered)	410B	60%
WebText2	19B	22%
Books1	12B	8%
Books2	55B	8%
Wikipedia	3B	3%

Key differences from GPT-2:

- Much larger and more diverse
- Includes Common Crawl (with quality filtering)

Few-Shot Learning

GPT-3's key capability: In-context learning

Learning Paradigms Compared

1. **Zero-shot:** Task description only

```
Translate to French: I love AI ->
```

2. **One-shot:** One example

```
sea otter -> loutre de mer
I love AI ->
```

3. **Few-shot:** Multiple examples (typically 10-100)

Few-Shot: Worked Example

Sentiment Classification with 3 examples:

```
prompt = """
Classify the sentiment of each review as Positive or Negative.

Review: "This movie was amazing, I loved every minute!"
Sentiment: Positive

Review: "Terrible film, complete waste of time."
Sentiment: Negative

Review: "A masterpiece of modern cinema."
Sentiment: Positive

Review: "The acting was wooden and the plot made no sense."
Sentiment:"""

# GPT-3 output: "Negative"
```

In-Context Learning: How It Works

The mechanism behind few-shot learning:

```
Prompt structure:
[Example 1] [Example 2] [Example 3] [New Input]

What GPT-3 "sees":
1. Pattern: "Review: X" followed by "Sentiment: Y"
2. Mapping: positive language -> "Positive"
3. Mapping: negative language -> "Negative"
4. Task: Apply this mapping to new input
```

Key observations:

- Model recognizes pattern in examples
- Continues the pattern for new input
- No weight updates - pure inference!

GPT-3's Emergent Abilities

Discussion

What can GPT-3 do that smaller models can't?

Emergent capabilities:

- **Arithmetic**: 2-3 digit addition/subtraction
- **Reasoning**: Simple logical deduction
- **Code generation**: Write simple programs
- **Knowledge synthesis**: Combine facts
- **Style transfer**: Mimic writing styles
- **Task composition**: Multi-step procedures

Scaling Hypothesis

These abilities weren't explicitly trained - they *emerged* from scale!

Emergent Abilities: Concrete Examples

Arithmetic (emerges around 10B parameters):

```
Q: What is 47 + 58?
A: 105
```

Code Generation:

```
# Write a function to check if a number is prime
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True
```

The Scaling Laws Hypothesis

Kaplan et al. (2020) - "Scaling Laws for Neural Language Models"

Model performance scales as a **power law** with:

- Model size (parameters)
- Dataset size (tokens)
- Compute (FLOPs)

Key findings:

1. Performance depends strongly on scale
2. Very weak dependence on model shape (depth vs width)
3. Smooth, predictable improvements
4. Optimal compute allocation: Grow model and data together

Think about it!

The Scaling Law Formula

Loss as a function of scale:

$$L(N) = \left(\frac{N_c}{N}\right)^{\alpha_N}$$

where:

- L = Cross-entropy loss
- N = Number of parameters
- N_c = Scaling constant
- $\alpha_N \approx 0.076$ (empirically determined)

In practice, this means:

10x more parameters	**->**	**~15% lower loss**

Scaling Laws: Visual Understanding

How loss decreases with scale:

```
Loss
  ^
  |
3.5|  *
   |    *
3.0|      *
   |        *
2.5|          *
   |            *
2.0|               *  *  *  *  * (diminishing returns)
   +------------------------------------------>
       10M   100M   1B    10B   100B   1T
                   Parameters
```

Key observation: Returns diminish but never stop - every 10x increase helps!

Implications of Scaling Laws

Good news:

- Predictable improvements
- Clear path to better models
- Can plan compute budgets
- Smooth progress curve

Challenges:

- Diminishing returns
- Exponential cost increase
- Hardware limitations
- Environmental impact

Chinchilla Scaling Laws

Hoffmann et al. (2022)

Previous models were **over-parameterized and under-trained!**

Key insight:

- For a given compute budget, should balance model size and data
- **Optimal ratio**: ~20 tokens per parameter

Comparison:

Model	Parameters	Tokens	Tokens/Param
GPT-3	175B	300B	1.7 (under-trained!)
Chinchilla	70B	1.4T	20 (optimal)

Result: Chinchilla (70B) outperforms GPT-3 (175B)!

The Path to ChatGPT

Evolution from GPT-3 to ChatGPT:

Stage	Model	Key Innovation
1	GPT-3	Predict next token
2	InstructGPT	Follow instructions
3	ChatGPT	Helpful & harmless chat

Three key innovations:

1. **Instruction tuning**: Train to follow instructions
2. **RLHF**: Reinforcement Learning from Human Feedback
3. **Safety guardrails**: Reduce harmful outputs

Instruction Tuning

What is Instruction Tuning?

Fine-tune the model on (instruction, response) pairs to make it better at following user commands.

Training examples:

```
Instruction: "Explain quantum computing to a 5-year-old"
Response: "Imagine you have a magic coin that can be heads
          AND tails at the same time until you look at it..."

Instruction: "Write a Python function to sort a list"
Response: "def sort_list(items):
              return sorted(items)"

Instruction: "Summarize this article in 3 sentences"
Response: [concise 3-sentence summary]
```

Before vs After Instruction Tuning

Before (GPT-3):

```
User: "Write a haiku about AI"

GPT-3: "Write a haiku about AI
is a common creative writing
exercise that many people
enjoy. Here are some tips
for writing haikus..."
```

Continues describing rather than doing

After (InstructGPT):

```
User: "Write a haiku about AI"

```

RLHF: Reinforcement Learning from Human Feedback

The 3-step RLHF process:

```
Step 1: Supervised Fine-tuning (SFT)
- Train on human-written examples of good responses
- Model learns basic instruction-following

Step 2: Reward Model Training
- Generate multiple responses to same prompt
- Humans rank responses from best to worst
- Train a model to predict human preferences

Step 3: RL Optimization (PPO)
- Generate responses with policy model
- Score with reward model
- Update policy to maximize reward
```

RLHF: Worked Example

Training the reward model:

```
Prompt: "How do I make a cake?"

Response A (Rating: 4/5):
"Here's a simple recipe: Preheat oven to 350F.
Mix 2 cups flour, 1.5 cups sugar..."

Response B (Rating: 2/5):
"Cake is a type of dessert that originated in
ancient civilizations..."

Response C (Rating: 1/5):
"I cannot help with that request."

Reward model learns:
- Helpful, direct answers get high scores
- Off-topic or unhelpful responses get low scores
```

ChatGPT's Impact

Launched: November 30, 2022

Growth:

- 1 million users in 5 days
- 100 million users in 2 months
- Fastest-growing consumer application ever

Why so successful?

- Easy to use (conversational interface)
- Broadly capable (many tasks)
- Accessible (free tier)
- Impressive demos went viral

The Modern LLM Landscape

Post-GPT-3 developments (2020-2024):

Year	Milestone
2021	Anthropic founded (Claude)
2022	ChatGPT launched
2023	GPT-4 (multimodal, improved reasoning)
2023	Llama 2 (open weights, 70B params)
2023	Gemini (Google's multimodal LLM)
2024	Claude 3, GPT-4o, Llama 3
2024	Smaller efficient models (Phi, Mistral)

Key trends:

Open Source LLMs

Discussion

Should powerful AI models be open or closed?

Closed (GPT-4, Claude):

- Better safety control
- Monetization easier
- Protect IP
- No transparency
- Vendor lock-in
- Limited customization

Open (Llama, Mistral):

- Transparency

Practical Prompting Techniques

Effective prompting strategies for modern LLMs:

1. Zero-shot with clear instructions:

```
Classify the following text as spam or not spam.
Only respond with "spam" or "not spam".

Text: "Congratulations! You've won $1,000,000!"
```

2. Few-shot with examples:

```
Text: "Meeting at 3pm tomorrow" -> not spam
Text: "URGENT: Send money now!" -> spam
Text: "Can you review this document?" -> not spam
Text: "You've been selected for a prize!" ->
```

Chain-of-Thought Prompting

For complex reasoning tasks:

```
Q: Roger has 5 tennis balls. He buys 2 more cans of
   tennis balls. Each can has 3 balls. How many
   tennis balls does he have now?

Let's think step by step:
1. Roger starts with 5 tennis balls
2. He buys 2 cans of tennis balls
3. Each can has 3 balls, so 2 cans = 2 * 3 = 6 balls
4. Total = 5 + 6 = 11 tennis balls

A: 11 tennis balls
```

Key insight

Adding "Let's think step by step" dramatically improves reasoning accuracy!

Current Limitations

What LLMs still struggle with:

1. **Factual accuracy**
    - Hallucinations and confabulation
    - No citations or sources
2. **Reasoning**
    - Multi-step logic
    - Mathematical proofs
3. **Knowledge grounding**
    - Knowledge cutoff date
    - Can't access real-time info

Key Takeaways

1. **Scaling works**
    - GPT -> GPT-2 -> GPT-3 showed clear improvements
    - Power law scaling continues to hold
2. **Few-shot learning emerged at scale**
    - No fine-tuning needed for many tasks
    - In-context learning is powerful
3. **Scaling laws provide predictability**
    - But diminishing returns and compute costs are real
    - Chinchilla scaling: balance model size and data
4. **RLHF changed everything**

Readings

Required Readings

1. **Radford et al. (2019)** - "Language Models are Unsupervised Multitask Learners" (GPT-2) [PDF]
2. **Brown et al. (2020)** - "Language Models are Few-Shot Learners" (GPT-3) [ArXiv]
3. **Kaplan et al. (2020)** - "Scaling Laws for Neural Language Models" [ArXiv]

Recommended Readings

- **Wei et al. (2022)** - "Emergent Abilities of Large Language Models" [ArXiv]
- **Ouyang et al. (2022)** - "Training language models to follow instructions" (InstructGPT) [ArXiv]
- **Hoffmann et al. (2022)** - "Training Compute-Optimal Large Language Models" (Chinchilla) [ArXiv]

Next Lecture Preview

Lecture 23: Implementing GPT from Scratch

- Building a mini-GPT in PyTorch
- Tokenization with BPE
- Training loop and optimization
- Sampling strategies (greedy, top-k, nucleus)
- Hands-on coding session

Questions?