Pretrained Language Models

Joseph Le Roux

20/08/26

Outline

- The Pre-training Paradigm
- BERT: Encoder-only Pre-training
- GPT: Decoder-only Pre-training
- Modern Large Language Models

1
The Pre-training Paradigm

1 The Fundamental Problem: Labeled Data is Scarce

Supervised learning requires labeled examples

- POS tagging: 50k sentences with grammatical annotations.
- Sentiment analysis: 10k movie reviews with polarity labels.
- Machine translation: millions of aligned sentence pairs.
- Each new task $\Rightarrow$ new annotation effort, new data collection.

But unlabeled text is essentially free

- The web contains trillions of words of raw text (Wikipedia, books, Common Crawl…).
- Can we learn *general linguistic knowledge* from raw text, without labels?

Key insight: language modeling is self-supervised

- Predicting the next word requires understanding syntax, semantics, world knowledge.
- The label (next word) is *free* provided by the text itself.

1 Transfer Learning in NLP

The two-stage paradigm

1. Pre-training: train a large Transformer on a massive corpus with a self-supervised objective.
   - No task-specific labels needed.
   - Learn general representations of language.
2. Fine-tuning: adapt the pre-trained model to a downstream task with a small labeled dataset.
   - Add a task-specific head (e.g., a linear classifier on top).
   - Train for a few epochs — much less data and compute required.

Why it works

- The pre-trained model has already learned syntax, semantics, and world facts.
- Fine-tuning just needs to *specialize* these representations for the task.
- Analogy: ImageNet pre-training for computer vision.

2

BERT: Encoder-only Pre-training

2 BERT: Bidirectional Encoder Representations from Transformers

[Dev+19]

Architecture

- Transformer encoder only (no decoder) 12 or 24 layers.
- $\text{BERT}_{\text{BASE}}$: 12 layers, 768 hidden dims, 12 heads, 110M parameters.
- $\text{BERT}_{\text{LARGE}}$: 24 layers, 1024 hidden dims, 16 heads, 340M parameters.

Key property: Bidirectionality

- Unlike GPT (causal/left-to-right), BERT sees the *full* sentence context at each position.
- Position i can attend to positions both before *and* after i.
- Result: richer, context-sensitive representations.

2 BERT Pre-training Objectives

1. Masked Language Modeling (MLM)

- Randomly mask 15% of input tokens.
- Of those: 80% replaced by `[MASK]`, 10% by a random word, 10% unchanged.
- Objective: predict the original token at masked positions.

$$\mathscr{L}_{\text{MLM}} = -\sum_{i \in \mathscr{M}} \log p_\theta(x_i \mid \tilde{x})$$

- Forces the model to use *both* left and right context.

2. Next Sentence Prediction (NSP)

- Input: two segments A and B separated by `[SEP]`.
- 50% of the time B follows A in the corpus (*IsNext*); 50% random (*NotNext*).
- Predict which case: binary classification on the `[CLS]` token.
- Helps with tasks requiring understanding of sentence relationships (QA, NLI).

2 BERT Input Format

Special tokens

- `[CLS]`: prepended to every input — its final hidden state is used for classification.
- `[SEP]`: separates sentence pairs (or ends single sentences).
- `[MASK]`: replaces masked tokens during pre-training.

```python
from transformers import BertTokenizer, BertModel
import torch

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')

text = "The quick brown fox jumps over the lazy dog."
inputs = tokenizer(text, return_tensors='pt')
# inputs: input_ids, attention_mask, token_type_ids

with torch.no_grad():
    outputs = model(**inputs)

# outputs.last_hidden_state: (1, seq_len, 768) - one vector per token
# outputs.pooler_output:     (1, 768)           - [CLS] vector (for classification)
cls_repr = outputs.pooler_output
```

2 Fine-tuning BERT

General approach

- Initialize with BERT weights (pre-trained).
- Add a small task-specific head on top.
- Train the whole model (BERT + head) on labeled data — a few epochs suffice.

Common fine-tuning setups

Task	Input	BERT output used	Head
Text classification	Single sentence	`[CLS]` vector	Linear(d, $\|C\|$)
NER / POS tagging	Sentence	All token vectors	Linear(d, $\|L\|$) per token
Question Answering	Question + Passage	All token vectors	2 linear layers (start/end)
NLI (entailment)	Premise + Hypothesis	`[CLS]` vector	Linear(d, 3)

2 Fine-tuning BERT for Classification

```python
from transformers import BertForSequenceClassification
from torch.optim import AdamW

model = BertForSequenceClassification.from_pretrained(
    'bert-base-uncased', num_labels=2  # e.g., positive/negative sentiment
)
optimizer = AdamW(model.parameters(), lr=2e-5)

for batch in train_loader:
    input_ids, attention_mask, labels = batch
    outputs = model(input_ids=input_ids,
                    attention_mask=attention_mask,
                    labels=labels)
    loss = outputs.loss   # cross-entropy computed internally
    loss.backward()
    optimizer.step()
    optimizer.zero_grad()
```

- Typical fine-tuning: 3–5 epochs, learning rate $\approx 2 - 5 \times 10^{-5}$.
- State-of-the-art on GLUE benchmark (2019) with a single model.

3 GPT: Decoder-only Pre-training

3 GPT: Generative Pre-trained Transformer

[Rad+18]

Architecture

- Transformer decoder only — causal (left-to-right) self-attention.
- GPT-1: 12 layers, 768 hidden dims, 12 heads, 117M parameters.
- GPT-2 (2019): 1.5B parameters; GPT-3 (2020): 175B parameters.

Pre-training objective: causal language modeling

Standard autoregressive LM objective — predict the next token:

$$\mathscr{L}_{\text{CLM}} = -\sum_{i} \log p_\theta(x_i \mid x_1, \ldots, x_{i-1})$$

- Simple, scalable, and self-supervised.
- The causal mask ensures x_i cannot see future tokens.

3 GPT vs. BERT

	BERT	GPT
Architecture	Encoder	Decoder
Attention	Bidirectional	Causal (left-to-right)
Pre-training	MLM + NSP	Causal LM
Strengths	Understanding tasks	Generation tasks
Fine-tuning	Most NLU tasks	Text generation, few-shot
Typical use	Classification, NER, QA	Chatbots, summarization, MT

When to use which?

- **BERT-style**: classification, tagging, extraction — tasks with a fixed output space.
- **GPT-style**: generation — tasks where the output is an open-ended sequence.
- **Encoder-decoder** (T5, BART): tasks that transform one sequence into another (translation, summarization).

3 Scaling Laws

More data + bigger models = better performance

Kaplan et al. (OpenAI, 2020) [Kap+20] showed that LM performance (perplexity) scales as a power law with:

- Number of model parameters N
- Dataset size D (tokens)
- Compute budget C (FLOPs)

$$L(N) \propto N^{-\alpha}, \quad L(D) \propto D^{-\beta}, \quad L(C) \propto C^{-\gamma}$$

Implications

- GPT-3 (175B params, 300B tokens) achieves remarkable few-shot performance.
- *Emergent abilities*: capabilities that appear abruptly above a threshold scale (chain-of-thought reasoning, arithmetic, etc.).
- Chinchilla (DeepMind, 2022): optimal compute allocation $\Rightarrow$ smaller models trained on more data outperform larger models on less data.

3 In-Context Learning and Prompting

GPT-3: few-shot without gradient updates

- Zero-shot: task description in the prompt, no examples.
- One-shot: one demonstration example in the prompt.
- Few-shot: k demonstration examples in the prompt ($k \leq 20$ or so).
- No fine-tuning: the model generalizes from the context window alone.

Example (few-shot sentiment analysis)

Review: "I loved this film." Sentiment: Positive
Review: "Boring and too long." Sentiment: Negative
Review: "An absolute masterpiece!" Sentiment:

Model completes: *Positive*

Why it works

- The model has learned so much about language and the world during pre-training that it can adapt to new tasks from demonstrations.

4 Modern Large Language Models

4 From GPT-3 to ChatGPT: Instruction Tuning

Problem with vanilla LMs

- A language model trained to predict the next token will *complete any prompt, not /follow instructions*.
- GPT-3 may generate harmful, biased, or irrelevant continuations.

4 Instruction tuning (FLAN [Wei+22], InstructGPT)

Fine-tune on a diverse set of tasks expressed as natural language instructions:

{"Translate the following English text to French: \text\"}
{"Summarize the article in 3 sentences: \article\"}
{"Answer the following question based on the passage: \question\ \passage\"}

- Makes models better at following new instructions *zero-shot*.
- FLAN (Google, 2021): fine-tuned T5 on 60+ NLP tasks as instructions — strong zero-shot generalization.

4 RLHF: Reinforcement Learning from Human Feedback [Ouy+22]

InstructGPT (Ouyang et al., OpenAI, 2022) — the foundation of ChatGPT

Three steps

1. Supervised Fine-Tuning (SFT): fine-tune GPT-3 on human-written demonstrations of desired behavior.
2. Reward Model (RM): train a model to predict human preference from pairwise comparisons of model outputs.
3. RL fine-tuning: optimize the SFT model with PPO (Proximal Policy Optimization) to maximize the reward model's score.

Effect

- Models that are *helpful*, *harmless*, and *honest* not just fluent.
- InstructGPT (1.3B) preferred over GPT-3 (175B) by human raters!
- $\Longrightarrow$ Alignment: making model behavior match human intentions.

Democratizing large language models

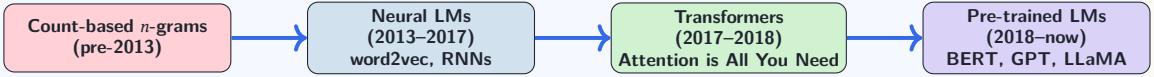
Model	Organization	Params	Notes
LLaMA 2	Meta	7B–70B	Instruction-tuned variants
Mistral 7B	Mistral AI	7B	Efficient, sliding window attention
Mixtral 8×7B	Mistral AI	47B active	Mixture of Experts
Gemma	Google	2B–7B	Lightweight, academic use
Falcon	TII (UAE)	7B–180B	Permissive license

Practical implications

- Run locally (quantized 4-bit models fit on consumer GPUs).
- Fine-tune on domain-specific data: medical, legal, code.
- LoRA (Low-Rank Adaptation): fine-tune only a small fraction of parameters, efficient and cheap.

4 Summary: The NLP Landscape in 2026

The paradigm shift



Key takeaways

- *Scale* + *self-supervision* = general-purpose language understanding and generation.
- Pre-train once, fine-tune (or prompt) everywhere.
- Transformers are the backbone of essentially all state-of-the-art NLP.
- Open-weight models make LLM research and application accessible.

5 References

[Dev+19] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. URL: https://www.aclweb.org/anthology/N19-1423.

[Kap+20] Jared Kaplan et al. *Scaling Laws for Neural Language Models*. 2020. arXiv: 2001.08361 [cs.LG]. URL: https://arxiv.org/abs/2001.08361.

[Ouy+22] Long Ouyang et al. *Training language models to follow instructions with human feedback*. 2022. arXiv: 2203.02155 [cs.CL]. URL: https://arxiv.org/abs/2203.02155.

[Rad+18] Alec Radford et al. *Improving language understanding by generative pre-training*. 2018.

[Wei+22] Jason Wei et al. "Finetuned Language Models are Zero-Shot Learners." In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=gEZrGCozdqR.