CSC 4315: INTRODUCTION TO ARTIFICIAL INTELLIGENCE

Week 14, Lecture 27

BERT, GPT, and Pre-training Strategies

Lecture Overview

This lecture covers the major pre-trained language models including BERT and GPT, their architectures, and fine-tuning strategies.

Topics Covered:

- Pre-training revolution
- BERT architecture and MLM
- GPT autoregressive modeling
- BERT vs GPT comparison
- Fine-tuning strategies

1. Pre-trained Language Models

1.1 The Pre-training Revolution

```
import torch
import torch.nn as nn

print("Pre-trained Language Models")
print("="*70)

print("Key insight: Learn general language understanding first")
print("  1. Pre-train on massive unlabeled text (Wikipedia, books)")
print("  2. Fine-tune on specific task with small labeled data")

print("\nWhy pre-training works:")
print("  - Language has structure that transfers")
print("  - Grammar, semantics, world knowledge")
print("  - Labeled data is expensive; unlabeled is abundant")

print("\nPre-training objectives:")
print("  - Masked Language Modeling (BERT)")
print("  - Next Token Prediction (GPT)")
print("  - Span Corruption (T5)")

print("\nScale of pre-training:")
print("  BERT-base: 110M params, trained on 3B words")
print("  GPT-2: 1.5B params")
print("  GPT-3: 175B params")
print("  GPT-4: ~1.8T params (estimated)")
```

```
Pre-trained Language Models
======================================================================
Key insight: Learn general language understanding first
  1. Pre-train on massive unlabeled text (Wikipedia, books)
  2. Fine-tune on specific task with small labeled data

Why pre-training works:
  - Language has structure that transfers
  - Grammar, semantics, world knowledge
  - Labeled data is expensive; unlabeled is abundant

Pre-training objectives:
  - Masked Language Modeling (BERT)
  - Next Token Prediction (GPT)
  - Span Corruption (T5)

Scale of pre-training:
  BERT-base: 110M params, trained on 3B words
  GPT-2: 1.5B params
  GPT-3: 175B params
  GPT-4: ~1.8T params (estimated)
```

2. BERT

2.1 BERT Architecture

```
print("BERT: Bidirectional Encoder Representations from Transformers")
print("="*70)

print("Key innovation: Bidirectional context")
print("  'The bank was on the river ___'")
print("  GPT: Only sees left context")
print("  BERT: Sees both left AND right context")

print("\nPre-training objectives:")
print("\n1. Masked Language Modeling (MLM):")
print("   'The [MASK] sat on the mat' -&gt; 'cat'")
print("   - Randomly mask 15% of tokens")
print("   - Predict masked tokens from context")

print("\n2. Next Sentence Prediction (NSP):")
print("   [CLS] Sentence A [SEP] Sentence B [SEP]")
print("   - Predict if B follows A (50% real, 50% random)")

print("\nBERT Input:")
print("  Token:    [CLS] The cat sat [SEP] On mat [SEP]")
print("  Segment:    A    A   A   A    A    B   B   B")
print("  Position:   0    1   2   3    4    5   6   7")

print("\nBERT sizes:")
print(f"{'Model':&lt;12} {'Layers':&gt;8} {'Hidden':&gt;8} {'Heads':&gt;8} {'Params':&gt;10}")
print("-" * 50)
print(f"{'BERT-base':&lt;12} {'12':&gt;8} {'768':&gt;8} {'12':&gt;8} {'110M':&gt;10}")
print(f"{'BERT-large':&lt;12} {'24':&gt;8} {'1024':&gt;8} {'16':&gt;8} {'340M':&gt;10}")
```

```
BERT: Bidirectional Encoder Representations from Transformers
======================================================================
Key innovation: Bidirectional context
  'The bank was on the river ___'
  GPT: Only sees left context
  BERT: Sees both left AND right context

Pre-training objectives:

1. Masked Language Modeling (MLM):
   'The [MASK] sat on the mat' -&gt; 'cat'
   - Randomly mask 15% of tokens
   - Predict masked tokens from context

2. Next Sentence Prediction (NSP):
   [CLS] Sentence A [SEP] Sentence B [SEP]
   - Predict if B follows A (50% real, 50% random)

BERT Input:
  Token:    [CLS] The cat sat [SEP] On mat [SEP]
  Segment:    A    A   A   A    A    B   B   B
  Position:   0    1   2   3    4    5   6   7

BERT sizes:
Model          Layers   Hidden    Heads     Params
--------------------------------------------------
BERT-base          12      768       12       110M
BERT-large         24     1024       16       340M
```

3. GPT

3.1 GPT Architecture

```
print("GPT: Generative Pre-trained Transformer")
print("="*70)

print("Key difference from BERT: Autoregressive (left-to-right)")

print("\nPre-training objective: Next Token Prediction")
print("  'The cat sat on the' -&gt; 'mat'")
print("  Uses causal masking (can only see past)")

print("\nGPT vs BERT:")
print(f"{'':15} {'BERT':&gt;15} {'GPT':&gt;15}")
print("-" * 45)
print(f"{'Direction':&lt;15} {'Bidirectional':&gt;15} {'Left-to-right':&gt;15}")
print(f"{'Pre-training':&lt;15} {'MLM + NSP':&gt;15} {'Next token':&gt;15}")
print(f"{'Best for':&lt;15} {'Understanding':&gt;15} {'Generation':&gt;15}")
print(f"{'Architecture':&lt;15} {'Encoder-only':&gt;15} {'Decoder-only':&gt;15}")

print("\nGPT Evolution:")
print("  GPT-1 (2018): 117M params, 12 layers")
print("  GPT-2 (2019): 1.5B params, 48 layers")
print("  GPT-3 (2020): 175B params, 96 layers")
print("  GPT-4 (2023): ~1.8T params (MoE)")

print("\nEmergent abilities:")
print("  - In-context learning")
print("  - Few-shot prompting")
print("  - Chain-of-thought reasoning")
```

```
GPT: Generative Pre-trained Transformer
======================================================================
Key difference from BERT: Autoregressive (left-to-right)

Pre-training objective: Next Token Prediction
  'The cat sat on the' -&gt; 'mat'
  Uses causal masking (can only see past)

GPT vs BERT:
                            BERT             GPT
---------------------------------------------
Direction         Bidirectional   Left-to-right
Pre-training          MLM + NSP      Next token
Best for          Understanding      Generation
Architecture       Encoder-only    Decoder-only

GPT Evolution:
  GPT-1 (2018): 117M params, 12 layers
  GPT-2 (2019): 1.5B params, 48 layers
  GPT-3 (2020): 175B params, 96 layers
  GPT-4 (2023): ~1.8T params (MoE)

Emergent abilities:
  - In-context learning
  - Few-shot prompting
  - Chain-of-thought reasoning
```

4. Fine-tuning Pre-trained Models

4.1 Fine-tuning Strategies

```
print("Fine-tuning Pre-trained Models")
print("="*70)

print("Fine-tuning BERT for classification:")
print("""
from transformers import BertModel, BertTokenizer

class BertClassifier(nn.Module):
    def __init__(self, num_classes):
        super().__init__()
        self.bert = BertModel.from_pretrained('bert-base-uncased')
        self.classifier = nn.Linear(768, num_classes)

    def forward(self, input_ids, attention_mask):
        outputs = self.bert(input_ids, attention_mask=attention_mask)
        pooled = outputs.last_hidden_state[:, 0]  # [CLS] token
        return self.classifier(pooled)
""")

print("Fine-tuning tips:")
print("  - Use small learning rate (2e-5 to 5e-5)")
print("  - Train for 2-4 epochs")
print("  - Warmup learning rate")
print("  - May freeze some layers initially")

print("\nParameter-efficient fine-tuning:")
print("  - LoRA: Low-rank adaptation")
print("  - Adapters: Small trainable modules")
print("  - Prompt tuning: Learn soft prompts")
print("  - Prefix tuning: Prepend trainable tokens")
```

```
Fine-tuning Pre-trained Models
======================================================================
Fine-tuning BERT for classification:

from transformers import BertModel, BertTokenizer

class BertClassifier(nn.Module):
    def __init__(self, num_classes):
        super().__init__()
        self.bert = BertModel.from_pretrained('bert-base-uncased')
        self.classifier = nn.Linear(768, num_classes)

    def forward(self, input_ids, attention_mask):
        outputs = self.bert(input_ids, attention_mask=attention_mask)
        pooled = outputs.last_hidden_state[:, 0]  # [CLS] token
        return self.classifier(pooled)

Fine-tuning tips:
  - Use small learning rate (2e-5 to 5e-5)
  - Train for 2-4 epochs
  - Warmup learning rate
  - May freeze some layers initially

Parameter-efficient fine-tuning:
  - LoRA: Low-rank adaptation
  - Adapters: Small trainable modules
  - Prompt tuning: Learn soft prompts
  - Prefix tuning: Prepend trainable tokens
```

Summary

Key Takeaways:

- Pre-training learns general language understanding

- BERT uses bidirectional context with MLM
- GPT uses autoregressive next-token prediction
- BERT for understanding, GPT for generation
- Fine-tune with small LR (2e-5 to 5e-5)
- Parameter-efficient methods for large models

Practice Exercises:

1. Compare BERT and GPT on text classification
2. Fine-tune BERT for sentiment analysis
3. Implement masked language modeling
4. Experiment with different fine-tuning strategies
5. Compare full fine-tuning vs LoRA