

Lecture 14:

Pre-trained Language Models & LLM Era

Deep Learning for Natural Language Processing

Ho-min Park

homin.park@ghent.ac.kr

powersimmani@gmail.com

Lecture Contents

{ Part 1: Introduction and Paradigm Shift

{ Part 2: Pre-training Concepts

{ Part 3: BERT - Encoder-based Models

{ Part 4: GPT - Decoder-based Models

{ Part 5: Encoder-Decoder Models

{ Part 6: Fine-tuning Strategies

{ Part 7: Prompting and In-Context Learning

{ Part 8: Current Trends

{ Part 9: Ethics and Practice

Part 1/9:

Introduction & Paradigm Shift

- 1.** Review of Previous Lessons
- 2.** Paradigm Shift in AI

NLP Evolution: From Rules to Neural Networks

Rule-Based Systems

Hand-crafted rules & feature engineering

RNN & LSTM

Sequential processing with memory

Seq2Seq Models

Encoder-decoder for translation

Word Embeddings

Word2Vec, GloVe semantic relationships

Attention Mechanism

Dynamic focus on relevant parts

Next Generation →

Transfer learning & deep context

⚠ Limitations

Shallow representations • Task-specific training

💡 The Need

Better context understanding • Transfer learning capabilities

Paradigm Shift in AI

✗ Old Paradigm

Task-Specific Models



Task-specific models trained from scratch



Requires labeled data for each task



Heavy feature engineering needed



Limited scale and capabilities



No knowledge transfer between tasks

✨ New Paradigm

Foundation Models



General-purpose foundation models



Pre-train on massive text, then adapt



Self-supervised learning from unlabeled data



Scale matters: emergent capabilities



Democratization: accessible to all

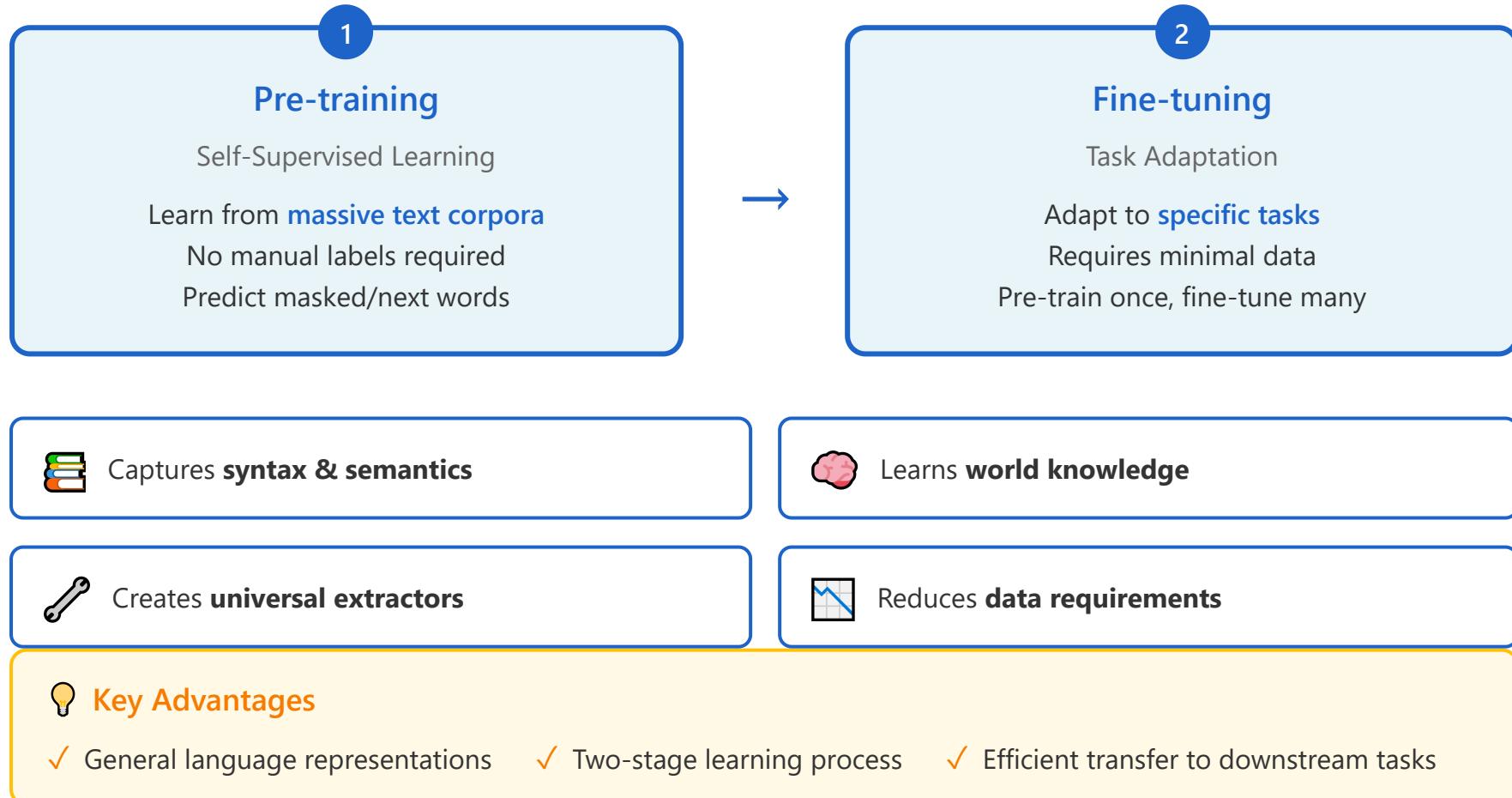


Part 2/9:

Pre-training Concepts

- 3. What is Pre-training?
- 4. Language Modeling Objective Function
- 5. The Importance of Scale

Pre-training → Fine-tuning Pipeline



Language Modeling Objective Functions

Autoregressive LM

Causal / Left-to-Right



$$P(w_t | w_1, \dots, w_{t-1})$$

→ Predict **next token**

⚡ **Causal masking** (uni-directional)

✍ Good for **generation** tasks

📝 GPT-style models

Masked LM

Bidirectional Context



$$P(w_{\text{masked}} | \text{context})$$

🎯 Predict **masked tokens**

↔ **Bidirectional** context (left+right)

🧠 Better **understanding**

BERT-style models

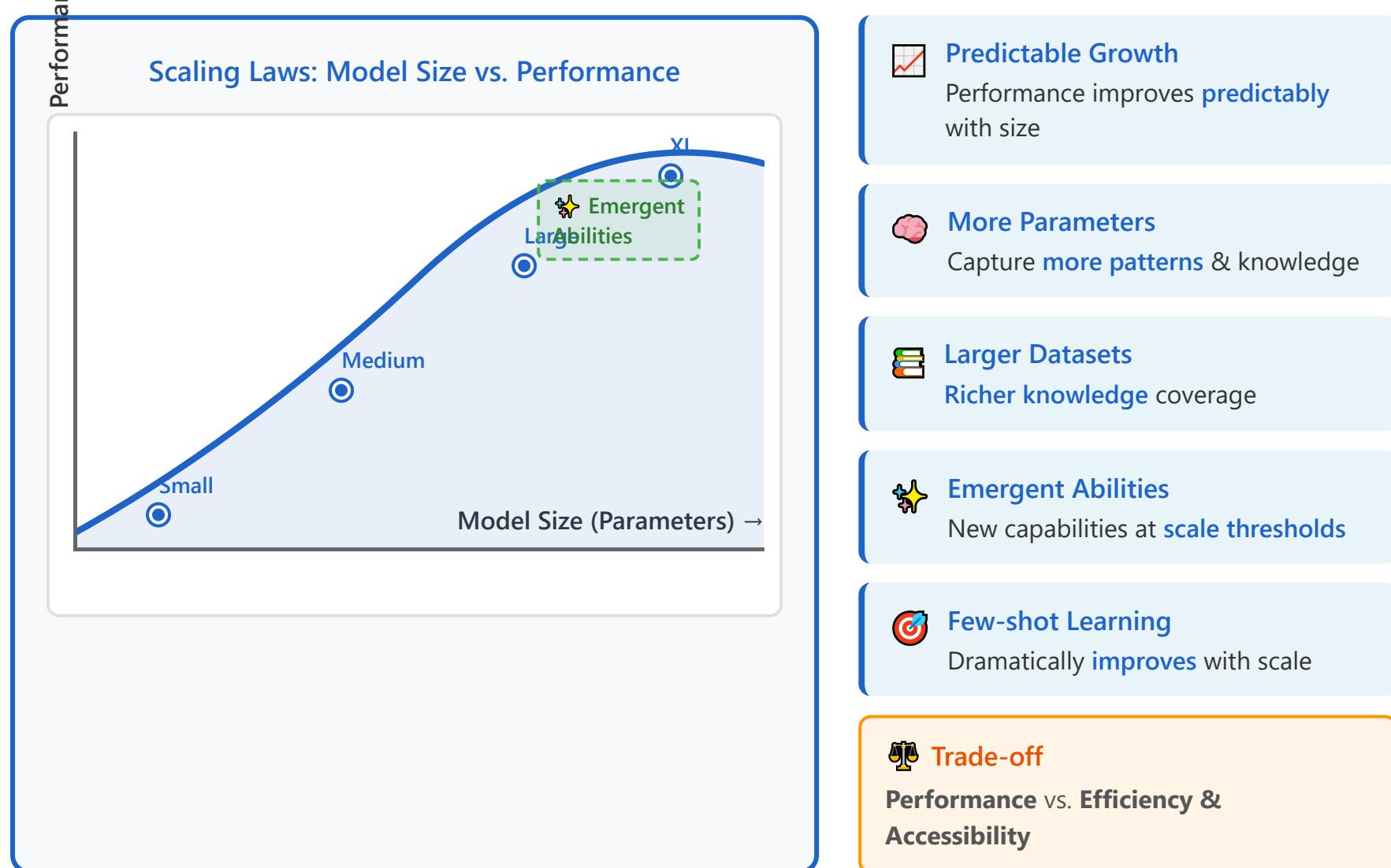
Training Approach

Self-supervision enables learning from unlimited text using **cross-entropy loss**

Architecture Fit

Different objectives suit different architectures and **downstream tasks**

The Importance of Scale



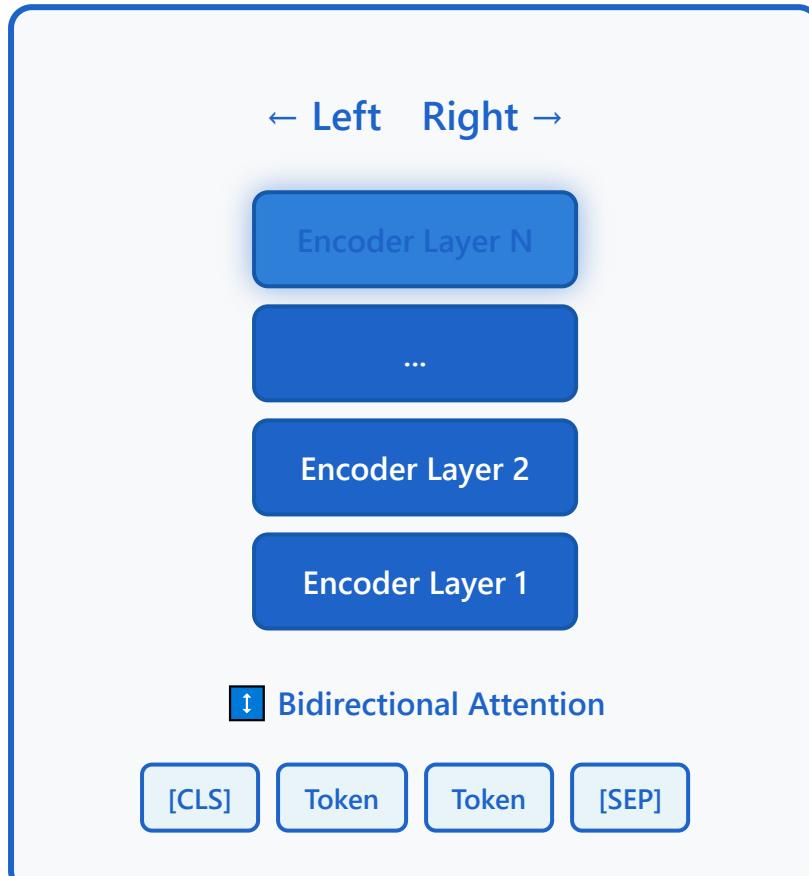
Part 3/9:

BERT - Encoder-based Models

- 6.** Introduction to BERT
- 7.** BERT Pre-training
- 8.** BERT Fine-tuning
- 9.** BERT Family Models

BERT: Bidirectional Encoder Representations from Transformers

Google, 2018 - Revolutionary NLP Model



Base: 110M params **Large:** 340M params
Dataset: BooksCorpus + Wikipedia (3.3B words)

- Bidirectional Context**
Deep **bidirectional** understanding from both directions
- Transformer Encoder**
Stack of **Transformer encoders** with self-attention
- Pre-training Corpus**
Trained on **BooksCorpus** and **Wikipedia**
- Two Model Sizes**
BERT-Base (110M) and BERT-Large (340M) parameters



State-of-the-Art Performance
Achieved SOTA on **11 NLP benchmarks**
Revolutionized NLP in 2018



1 Input Preparation

Tokenize the input text and add special tokens. [CLS] marks the start of the sentence, and [SEP] marks the end.

Example Input:

"The cat sat on the mat"
→ [CLS] The cat sat on the mat [SEP]

2 Masked Language Modeling (MLM)

Randomly mask 15% of input tokens. The model learns to predict masked words using bidirectional context.

Masking Example:

Input: [CLS] The cat sat on the mat [SEP]
Masked: [CLS] The [MASK] sat on the mat [SEP]
Target: Predict "cat"

Key Points

- 80% replaced with [MASK]
- 10% replaced with random word
- 10% keep original word

3 Next Sentence Prediction (NSP)

Learn to determine whether two sentences are consecutive. This improves the ability to understand relationships between sentences.

⌚ NSP Example:

Case 1 (IsNext) :

[CLS] The cat sat on the mat [SEP] It was sleeping [SEP]

Label: IsNext ✓

Case 2 (NotNext) :

[CLS] The cat sat on the mat [SEP] Paris is beautiful [SEP]

Label: NotNext X

4

Training Objective

Train the model by simultaneously optimizing both MLM and NSP losses.

MLM Loss

Cross-Entropy Loss aiming for accurate prediction of masked tokens

NSP Loss

Binary Classification Loss for correctly judging sentence pair continuity

⌚ Total Loss

Loss = MLM Loss + NSP Loss



BERT Fine-tuning Process

1 Load Pre-trained Model

Load the pre-trained BERT model. It has already learned general patterns of language.

💡 Pre-trained Knowledge

- Understanding grammatical structure
- Capturing semantic relationships
- Bidirectional context comprehension

2 Add Task-Specific Layer

Add an output layer specific to the task. Different structures are used depending on the task.

📊 Classification

[CLS] token output + Softmax Layer
(Sentiment analysis, topic classification, etc.)

🏷️ Token Classification

Each token output + Classification Layer
(Named entity recognition, POS tagging, etc.)

❓ Question Answering

Start/End Position Prediction Layers
(SQuAD, reading comprehension, etc.)

🔁 Sequence Pairing

[CLS] token + Binary Classifier
(Natural language inference, sentence similarity, etc.)

3 Fine-tune on Task Data

Fine-tune the entire model with task-specific training data. Typically, 2-4 epochs are sufficient.

Sentiment Analysis Example:

Input: [CLS] This movie is amazing [SEP]
→ BERT Encoding →
→ Classification Layer →
Output: Positive (95% confidence)

Training Settings

- Learning Rate: 2e-5 ~ 5e-5
- Epochs: 2-4
- Batch Size: 16 or 32

4 Inference & Prediction

Use the fine-tuned model to make predictions on new data.

Prediction Pipeline:

- Step 1:** Tokenize new input
- Step 2:** Pass through fine-tuned BERT
- Step 3:** Apply task-specific head
- Step 4:** Generate prediction with confidence score

Final Output

Generate results in the appropriate format for the task:
Classification labels, token tags, answer spans, etc.

BERT revolutionized NLP by introducing bidirectional pre-training and transfer learning to the field.

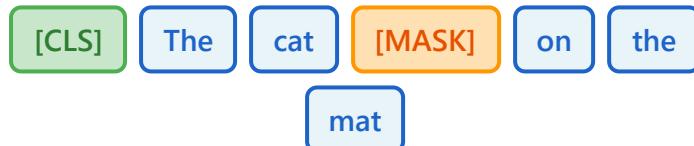
BERT Pre-training: Two Main Tasks

1

Masked Language Model

MLM - 15% masking

Input:



Predict: "sat"

Uses **bidirectional context**
to predict masked tokens

2

Next Sentence Prediction

NSP - Binary classification

Input:

A: The man went to the store.

B: He bought a gallon of milk.



IsNext ✓

[CLS] token for
sentence-level representation



Tokenization

WordPiece

30K vocabulary



Special Tokens

[CLS]

[SEP]

[MASK]

[PAD]



Training

4 days

on 4-16 Cloud TPUs

BERT Fine-tuning for Different Tasks

🧠 Pre-trained BERT (All Parameters)

+



Sequence Classification

Uses:



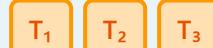
token representation

Sentiment Analysis,
Text Classification



Token Classification

Uses:



each token's output

NER, POS Tagging



Question Answering

Predicts:

Start & End spans

SQuAD, Reading
Comprehension

↓

⚡ End-to-End Fine-tuning

All parameters updated with task-specific data

Fast Training

Task-Specific

Transfer Learning



Only **few epochs** needed



Add layer on **top of BERT**



Few-shot & zero-shot

BERT Family: Variants and Improvements

 BERT (2018)

 Optimization

RoBERTa

No NSP • Dynamic masking • Larger batches

ALBERT

Parameter sharing • Factorized embeddings

DistilBERT

6 layers • 40% smaller • 97% performance

 Innovation

ELECTRA

Discriminative pre-training • More efficient

DeBERTa

Disentangled attention • Enhanced mask decoder



Domain-Specific

BioBERT

SciBERT

ClinicalBERT



Multilingual

mBERT

XLM-R

100+ languages

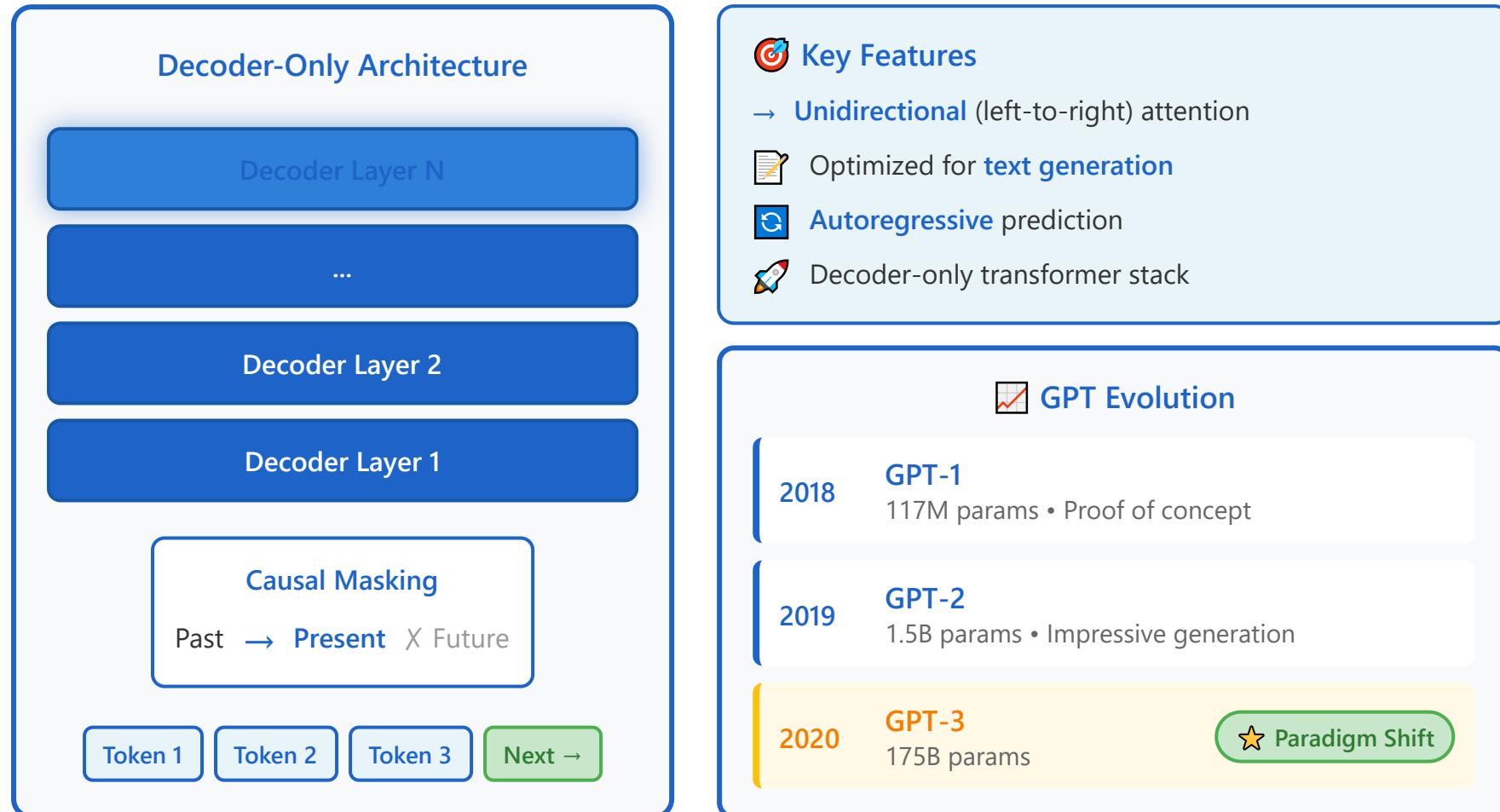
Part 4/9:

GPT - Decoder-based Models

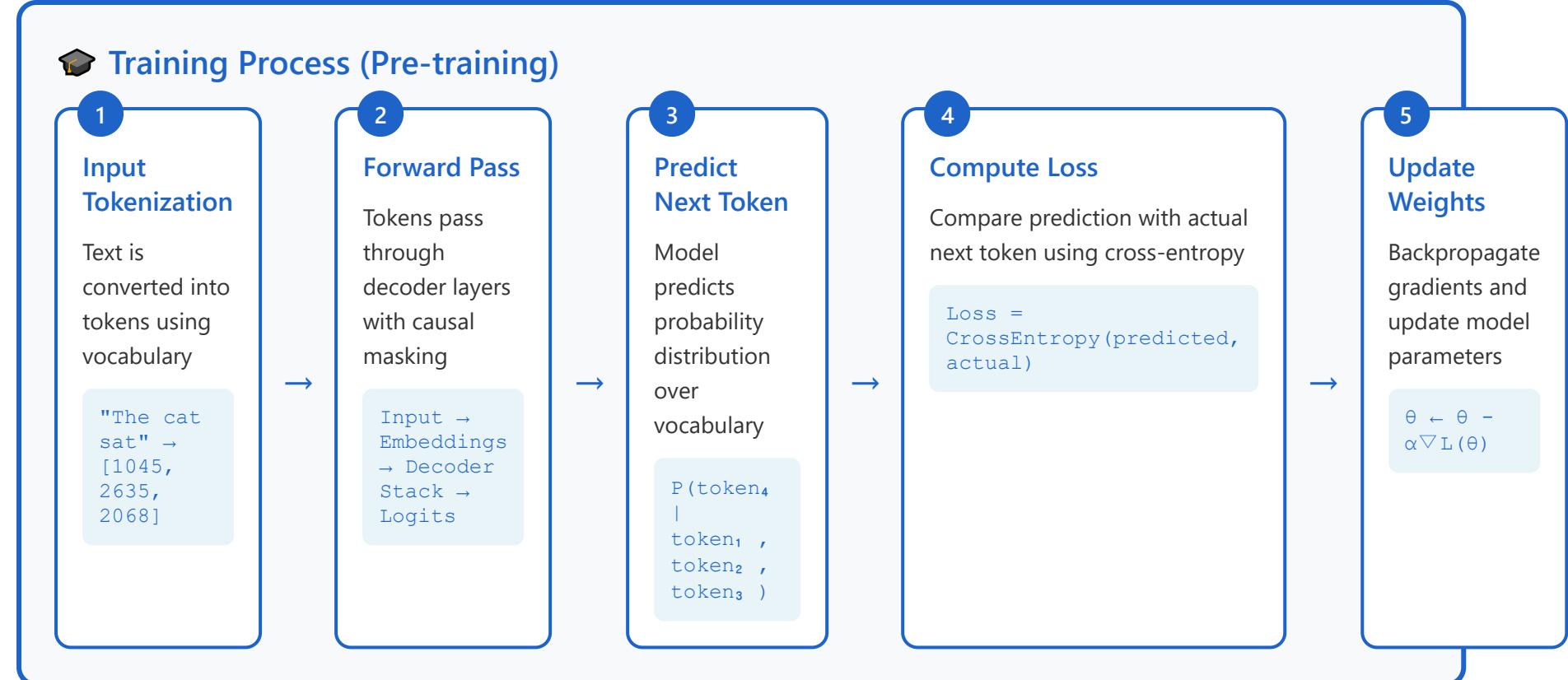
- 10.** Introduction to the GPT Series
- 11.** GPT Pre-training
- 12.** GPT-3 and Few-shot Learning
- 13.** GPT Family Development

GPT: Generative Pre-trained Transformer

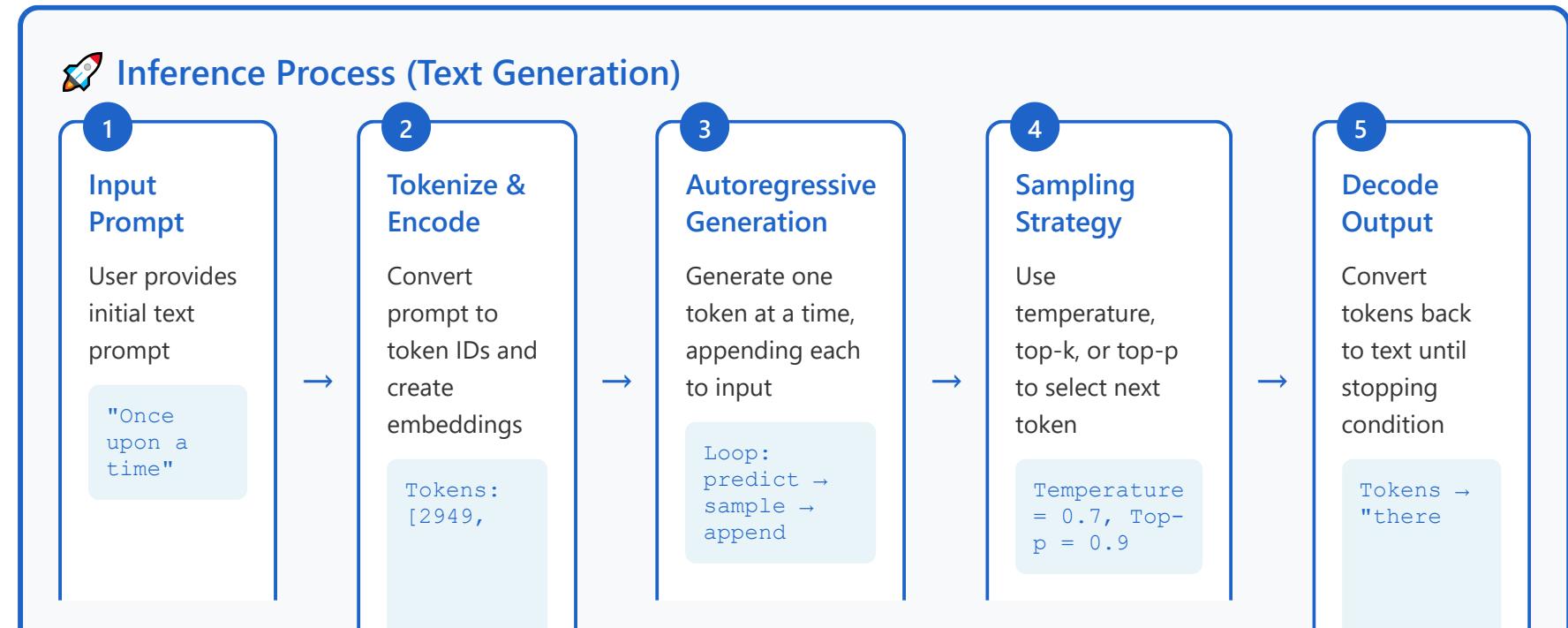
OpenAI - Autoregressive Decoder-only Architecture



Training Process (Pre-training)



Inference Process (Text Generation)



2402,
257, 640]

lived a
princess"

💡 Step-by-Step Example

- ▶ Input: "The weather is"
- ▶ Step 1: predict → "sunny"
- ▶ Step 2: "The weather is sunny" → predict → "today"
- ▶ Step 3: "The weather is sunny today" → predict → ".."
- ▶ Final Output: "The weather is sunny today."

GPT Pre-training: Causal Language Modeling

⌚ Autoregressive Prediction Process

1 The → cat

Predict next token

2 The cat → sat

Use previous outputs

3 The cat sat → on

Continue generation



Training Data

Diverse internet text: **Common Crawl, WebText**



Unidirectional Attention

Enables **efficient generation**



BPE Tokenization

Byte-pair encoding for vocabulary



No Explicit Fine-tuning

GPT-3: **Direct inference** capable



Pattern Learning



Massive Scale



Efficiency

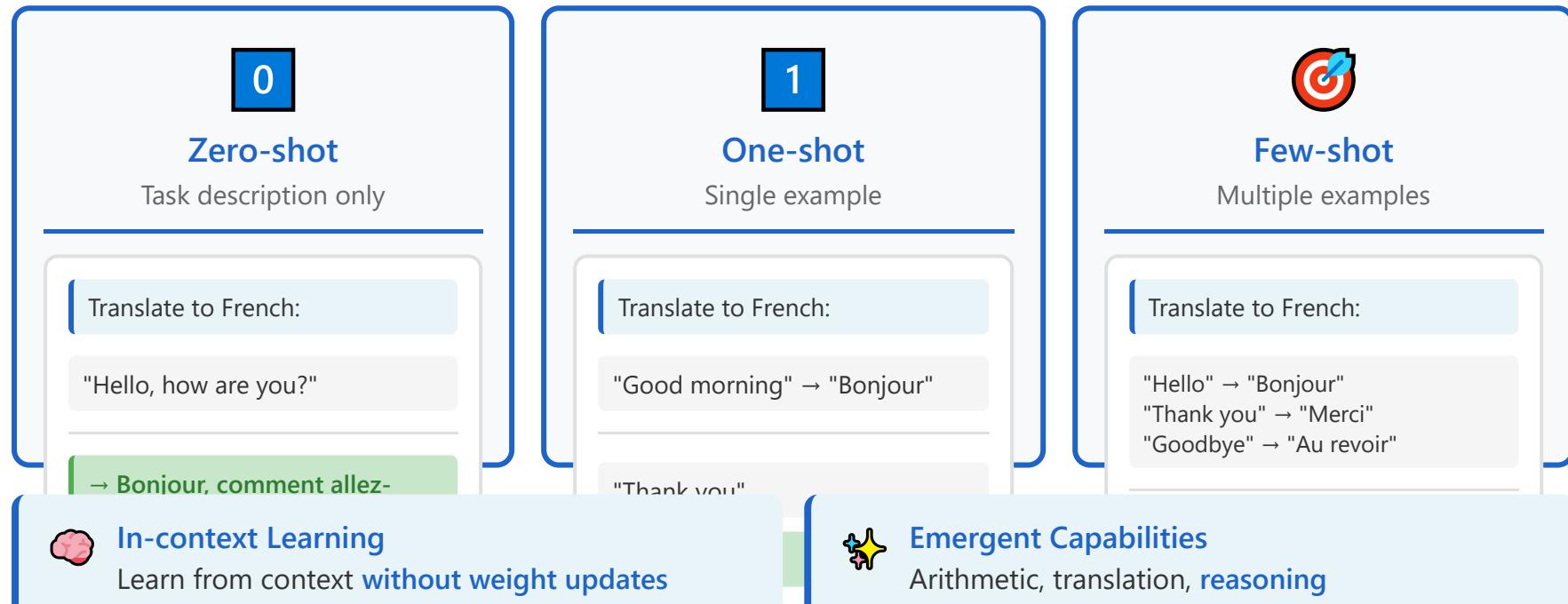
Learns patterns, facts, and reasoning
from data

Unlocks in-context learning abilities

Fast generation through causal
masking

GPT-3 and Few-shot Learning

175B parameters • 300B tokens • In-context Learning



Democratization of AI

Natural language interfaces enable users to interact with AI without technical expertise

GPT Family Development Timeline

Evolution & Innovations



InstructGPT

RLHF for instruction following

Evolution & Innovations



ChatGPT

Conversation-optimized with **safety alignment**



GPT-4

Multimodal • Improved reasoning & reliability



Codex

Code generation • Powers **GitHub Copilot**



Open Source Alternatives

GPT-Neo

GPT-J

Bloom

LLaMA



Safety

Continuous improvements



Accuracy

Enhanced reliability



API Access

Widespread development

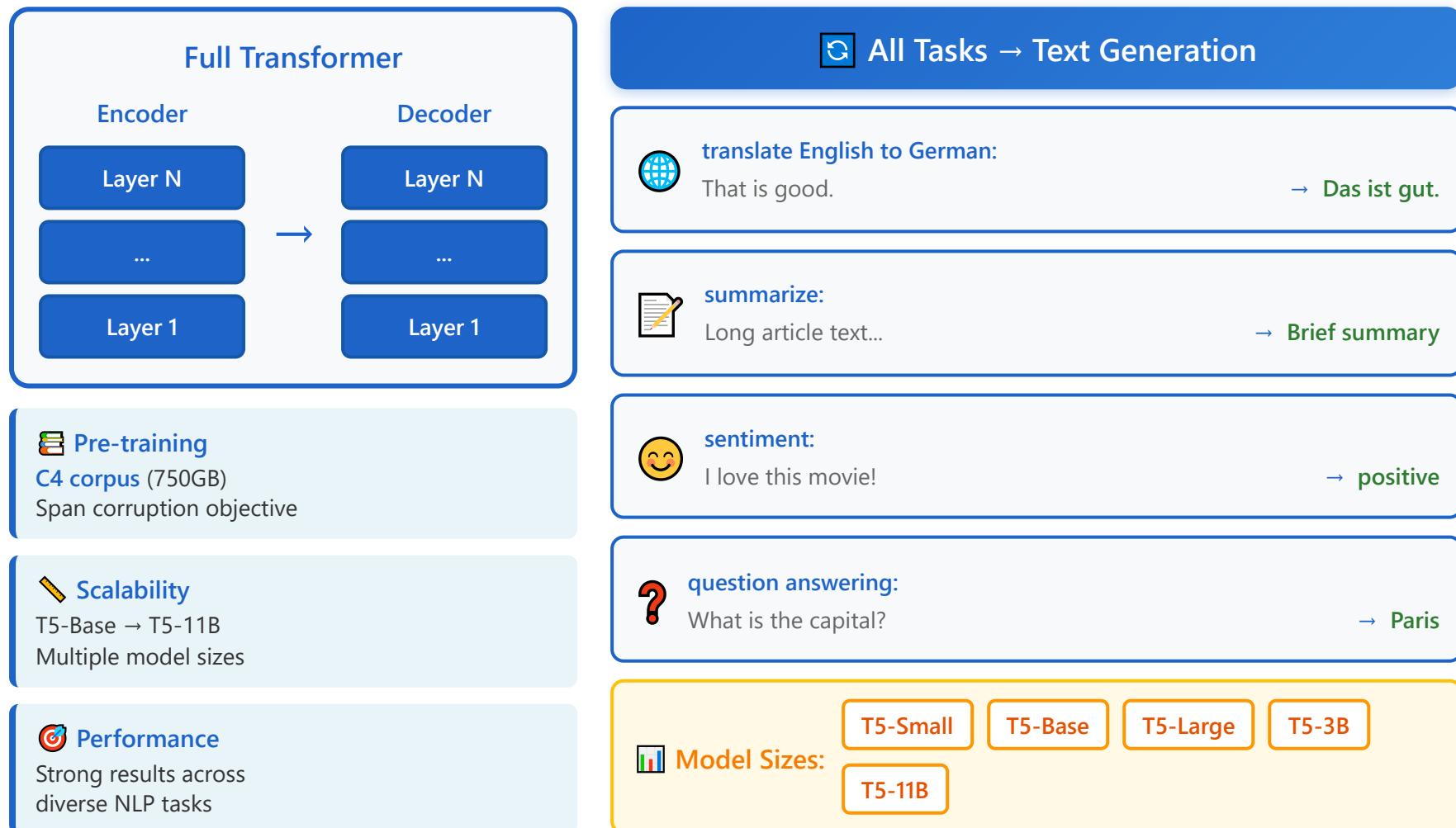
Part 5/9:

Encoder-Decoder Models

- 14.** T5 - Text-to-Text Framework
- 15.** BART and Other Models

T5: Text-to-Text Transfer Transformer

Google - Unified Framework for All NLP Tasks



BART and Other Encoder-Decoder Models

⟳ BART: Denoising Autoencoder

Combines BERT (Encoder) + GPT (Decoder)

1. Corrupt Input

Add noise

2. Encode

BERT-style

3. Decode

GPT-style

4. Reconstruct

Original text

Token Masking

The [MASK] sat

Token Deletion

The _ sat

Span Infilling

The [X] mat

Sentence Permutation

Sent2 Sent1 Sent3

Document Rotation

...end → start...

Text Infilling

A B [Y] E F



mBART

Multilingual BART for 50+ languages



PEGASUS

Gap sentence generation for summarization



Marian

Efficient neural machine translation



BART Strengths

Particularly strong for generation & summarization



Encoder-Decoder Models

Part 6/9:

Fine-tuning Strategies

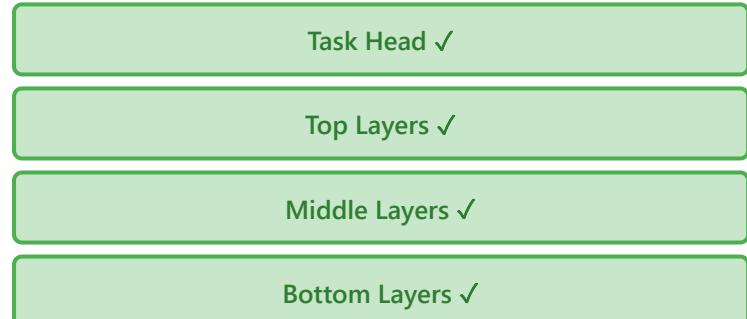
- 16.** Full Fine-tuning vs. Transfer Learning
- 17.** Parameter-Efficient Fine-tuning
- 18.** Fine-tuning Practical Tips

Fine-tuning Strategies Comparison



Full Fine-tuning

Update all parameters

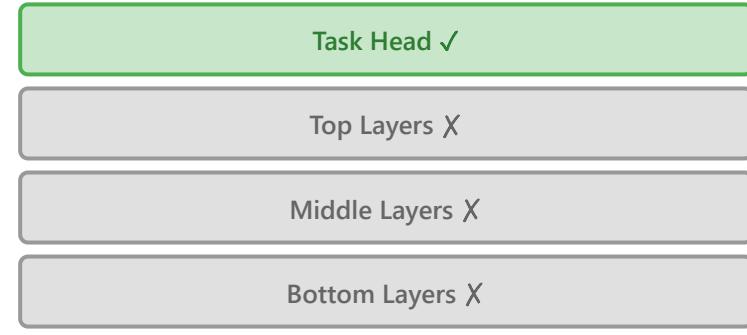


- ✓ Best performance
- ⚠ Slower training
- ⚠ Risk of forgetting



Feature Extraction

Freeze base model



- ✓ Faster training
- ✓ No catastrophic forgetting
- ⚠ Lower performance



Gradual Unfreezing

Progressive unlocking



Discriminative LR

Different rates per layer

Task Head ✓

Top Layers ⚡

Middle Layers →

Bottom Layers →

- ✓ Balanced approach
- ✓ Better stability
- Layer-by-layer control

Task Head (High LR)

Top Layers (Medium LR)

Middle Layers (Low LR)

Bottom Layers (Very Low LR)

- ✓ Fine-grained control
- ✓ Preserves lower features
- Requires tuning

🎯 Performance

Full FT: **Best**

Feature Ext: **Good**

⚡ Speed

Feature Ext: **Fastest**

Full FT: **Slowest**

💾 Memory

Feature Ext: **Low**

Full FT: **High**

💡 Decision Factors

Choice depends on **data size** and **computational budget** • Large data + resources → Full FT • Small data / limited resources → Feature Extraction

Parameter-Efficient Fine-tuning (PEFT)



LoRA

Low-Rank Adaptation

W
(Frozen)



A

x
B

Low-rank matrices

$$W' = W + AB$$

<1% parameters trainable



Adapter Layers

Bottleneck Modules

Transformer Layer (Frozen)



Adapter (Down → Up)



Transformer Layer (Frozen)



Adapter (Down → Up)



Small modules between layers



Prefix Tuning

Optimize continuous prompt vectors



Prompt Tuning

Learn soft prompts, freeze model



BitFit

Only tune bias terms, freeze weights



Single GPU Training

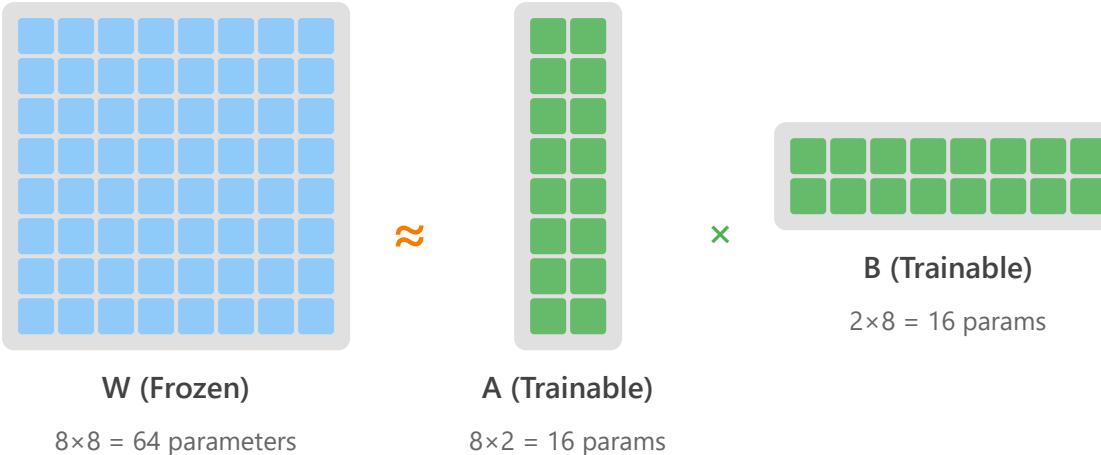
Fine-tune **large models** on limited hardware



High Efficiency

90-95% of full fine-tuning performance

Practical Example: LoRA Parameter Decomposition



64

Original Parameters

32

LoRA Parameters
(16 + 16)

50%

Reduction

With Rank $r = 2$: $(m \times r) + (r \times n) = (8 \times 2) + (2 \times 8) = 32$

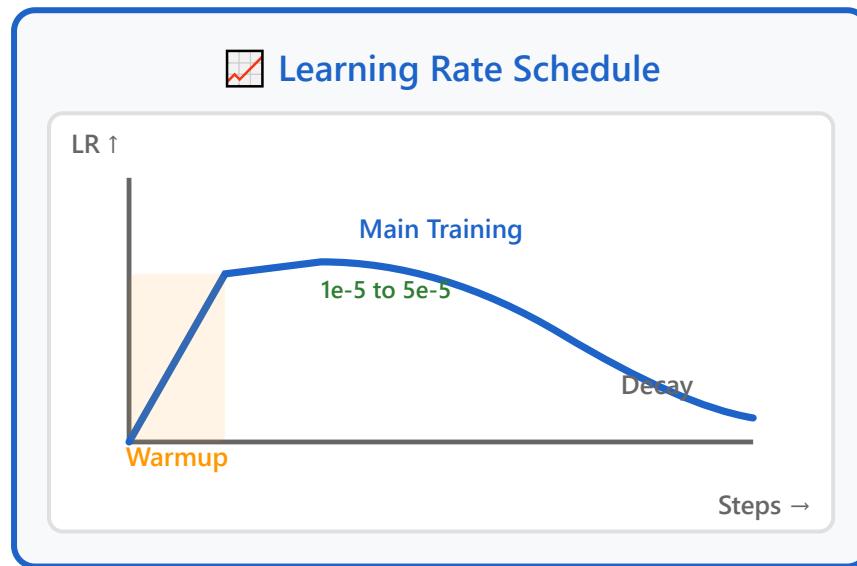
Lower rank = fewer parameters

($r=1$: 16 params, $r=2$: 32 params, $r=4$: 64 params)



In large-scale models, over 99% parameter reduction is possible!

Fine-tuning Practical Tips



- 1 **Learning Rate**
Start with **1e-5 to 5e-5** range
- 2 **Warmup Steps**
Use warmup to **stabilize training**
- 3 **Monitor Validation**
Track validation loss to **prevent overfitting**
- 4 **Data Quality**
Quality > Quantity for better results
- 5 **Batch Size**
Affects **convergence & generalization**
- 6 **Early Stopping**
Stop based on **validation metrics**
- 7 **Multi-task Learning**
Consider for **related tasks**

 **Best Practice**

— Training - - Validation

Save checkpoints and test **multiple configurations** to find optimal settings

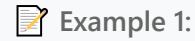
Part 7/9:

Prompting & In-Context Learning

- 19.** Prompt Engineering Basics
- 20.** Advanced Prompting Techniques
- 21.** Fine-tuning vs. Prompting

Prompt Engineering: Good vs. Bad Examples

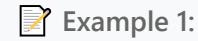
Bad Prompts



"Write something about AI."

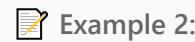
 Too vague

Good Prompts



"You are an AI expert. Write a 3-paragraph explanation of machine learning for beginners, using simple analogies."

 Clear role & constraints



 Example 2:

"Summarize this article."

⚠ No context or format

📝 Example 3:

"Translate to French."

⚠ Missing input text



Clear Instructions

Explicit & precise



Context

Background info

"Summarize the following article in 3 bullet points, focusing on key findings. Format: • Point 1\n• Point 2\n• Point 3"

✓ Specific format & context

📝 Example 3:

"Translate the following English text to French. Maintain a formal tone.\n\nText: 'Welcome to our conference.'"

✓ Complete with examples



Format

Define structure



Role

"You are an expert"



Iterative refinement: Test and improve prompts



Consistency: Maintain format across examples

Advanced Prompting Techniques

Chain-of-Thought (CoT)

Question:

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?

Prompt:

"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 $2 \times 3 = 6$ balls

4 Total: $5 + 6 = 11$ balls

✓ Final Answer:

Other Advanced Techniques

Few-shot Learning



Provide **multiple examples** in prompt for pattern recognition

Self-consistency



Sample **multiple outputs** and vote for most consistent answer



Tree of Thoughts

Explore **multiple reasoning paths** simultaneously



ReAct

Reasoning and acting in interleaved manner



Auto Prompt Optimization

Use **evolutionary methods** to improve prompts

11 tennis balls



Meta-prompts

Prompts that generate prompts for different tasks

Fine-tuning vs. Prompting: When to Use Each?

Prompting

 Quick - No training needed

 Flexible - Easy to iterate

 API-based - Simple access

 Pay per API call

✓ Ideal for:

Prototyping

Few examples

Varied tasks

Quick experiments

Fine-tuning

 Better performance - Task-specific

 Consistent - Reliable outputs

 Private data - Full control

 Training infrastructure needed

✓ Ideal for:

Production

Large datasets

Low latency

Custom domains



Requirements

Consider **task complexity** & goals



Data

Evaluate **dataset size** & quality



Resources

Balance **cost** & **infrastructure**



Hybrid Approach

Prompt-based fine-tuning (instruction tuning)
combines both benefits



Future Trend

Prompting becoming more powerful with larger
models

Part 8/9:

Current Trends

22. RLHF and Instruction Tuning

23. Present and Future

RLHF and Instruction Tuning

Reinforcement Learning from Human Feedback

Three-Stage RLHF Pipeline

1 Supervised Fine-tuning



Train on high-quality demonstration data

2 Reward Model



Learn human preferences from comparisons

3 PPO Training



Optimize policy using reward model



Helpfulness

Better at following instructions



Truthfulness

More accurate and reliable



Safety

Reduces harmful outputs



Instruction Tuning

Train on **diverse task instructions** to improve generalization



Constitutional AI

Self-improvement through principles and guidelines



Key to Success

RLHF and Instruction Tuning are fundamental to the success of **ChatGPT** and other assistant models

Present and Future: Current Trends in LLMs



Multimodal

Text, image, audio, video integration



Mixture of Experts

Sparse activation for efficiency



Long Context

100K+ tokens processing



Better Reasoning

Math, logic, commonsense



Retrieval Augmentation

Combining LMs with knowledge bases for accurate information



Smaller Efficient Models

On-device and edge deployment capabilities



Enhanced Capabilities

Improvements in mathematical reasoning, logical thinking, and commonsense understanding



Context Processing

Processing 100K+ tokens enables handling entire books and documents



Specialized Domain Models

Medical Legal Scientific Finance

Engineering



AGI Considerations

Focus on capabilities, safety, and alignment as we approach AGI

Part 9/9:

Ethics and Practice

24. Summary and Practical Considerations

Summary and Practical Considerations

🚀 Pre-trained LLMs Revolutionized the NLP Landscape

🏗 Choose Your Architecture

Encoder
BERT-style

Decoder
GPT-style

Both
T5-style



Bias & Fairness



Privacy



Misinformation



Environment



Diverse Evaluation

Test on diverse datasets and scenarios



Document Limitations

Clearly state model capabilities and constraints



Transparency

Open about model behavior and decisions



Human Oversight

Maintain human-in-the-loop for critical tasks



Environmental Cost

Training emissions and energy consumption



Accessibility

Democratization vs. resource concentration



Continuous Learning

Field evolves rapidly, stay updated



Responsible Deployment

Thank you

Ho-min Park

homin.park@ghent.ac.kr

powersimmani@gmail.com