**Lecture Notes: Pretraining Strategies in Transformer Models**

**Overview**

This lecture discusses three main pretraining strategies used in transformer-based architectures:

1. **Encoder-only models** (e.g., BERT)
2. **Encoder-decoder models** (e.g., T5, BART)
3. **Decoder-only models** (e.g., GPT, ChatGPT, LLaMA)

The session highlights their architectural differences, use-cases, and pretraining methodologies.

**1. Encoder-Only Models**

**Example:** BERT (Bidirectional Encoder Representations from Transformers)

* **Pretraining Strategy:** Masked Language Modeling (MLM)
  + Randomly mask some tokens from the input.
  + The model learns to predict the masked tokens using surrounding context.
  + It is a **self-supervised** task: no labeled data is needed.
* **Architecture:**
  + Fully bidirectional attention using self-attention.
  + Not autoregressive (can see both left and right contexts).
* **Use-case:** Best suited for **understanding tasks** (e.g., classification, NER, QA).

**2. Encoder-Decoder Models**

**Examples:** T5 (Text-to-Text Transfer Transformer), BART (Bidirectional and AutoRegressive Transformer)

**Architecture:**

* Based on **vanilla Transformer**:
  + **Encoder**: Processes full input (e.g., document, instruction + input).
  + **Decoder**: Generates output step-by-step using masked multi-head attention and cross-attention with encoder outputs.
* **Components:**
  + Input embeddings + positional encodings.
  + **Encoder blocks:** Each has multi-head attention, feed-forward layers, residual connections, and normalization.
  + **Decoder blocks:** Contain masked multi-head self-attention, cross-attention (with encoder outputs), and feed-forward layers.

**Pretraining Approach:**

* The objective is still **language modeling** (e.g., next word prediction).
* Input to encoder = entire prefix (e.g., instruction + document).
* Input to decoder = output sequence (e.g., summary, translation), shifted right.
* Output = probabilities over vocabulary using a softmax layer.

**Motivation:**

* Can be used for **generative tasks**.
* However, computationally **expensive** (due to dual encoder-decoder structure).

**BART Pretraining Strategy:**

* Combines ideas from BERT and GPT.
* Uses **noisy input generation** to make pretraining **self-supervised**:
  + **Token Masking**: Similar to BERT; mask random tokens.
  + **Token Deletion**: Remove random tokens.
  + **Sentence Permutation**: Randomly shuffle sentence order.
  + **Text Infilling**: Mask spans of text.
  + **Document Rotation**: Circularly shift the text.
* Input (corrupted text) → Encoder
* Output (original text) → Decoder

**3. Decoder-Only Models**

**Examples:** GPT series (GPT-1, 2, 3, 4), ChatGPT, LLaMA

* **Architecture:**
  + Only the **decoder** part of the transformer.
  + Uses **masked self-attention** (autoregressive): at time step *t*, only sees tokens from 1 to *t-1*.
* **Pretraining Strategy:** Causal Language Modeling (CLM)
  + Given tokens 1 to *t-1*, predict token *t*.
  + Self-supervised and highly efficient for **next-token prediction**.
* **Advantages:**
  + Suitable for **generative tasks** (e.g., code generation, writing).
  + Parameter efficient compared to encoder-decoder models.
* **Disadvantages:**
  + Cannot use bidirectional context.
  + Less suited for tasks requiring full input understanding.

**Comparison Summary**

| **Model Type** | **Example** | **Context Type** | **Main Use-Case** | **Training Style** | **Pros** | **Cons** |
| --- | --- | --- | --- | --- | --- | --- |
| Encoder-only | BERT | Bidirectional | Understanding (e.g. QA, NER) | Masked Language Modeling | Full input context | Not suitable for generation |
| Encoder-decoder | T5, BART | Both | Generation & Understanding | Masked/Noisy Seq-to-Seq | Flexible; handles both tasks | Computationally heavy |
| Decoder-only | GPT, ChatGPT | Left-to-right | Generation (e.g. Chat, Code) | Causal Language Modeling | Lightweight; generative power | Lacks right context comprehension |

**Conclusion**

* Different transformer pretraining strategies serve different goals.
* **Encoder-only models** are best for understanding tasks.
* **Decoder-only models** are efficient and powerful for generation.
* **Encoder-decoder models** are versatile but require more resources.
* Modern trends favor **decoder-only architectures** due to scalability and generation efficiency (e.g., LLaMA, GPT).

**Key Terms:**

* **Masked Language Modeling (MLM)**: Predict masked tokens.
* **Causal Language Modeling (CLM)**: Predict next token using previous tokens.
* **Self-supervised learning**: Learning from data without human-annotated labels.
* **Cross-attention**: Decoder attends to encoder outputs.

**Leveraging Unlabeled Data in Encoder-Decoder Models**

**Challenge:**

* Tasks like summarization or translation need labeled data
* Need a way to use **self-supervised learning** (like BERT)

**Solution: BART (Bidirectional and Auto-Regressive Transformer)**

* Combines bidirectional encoding (like BERT) with autoregressive decoding (like GPT)
* Enables self-supervised training on unlabeled data

**Pre-training via Noisy Inputs:**

* **Objective:** Recover original text from corrupted versions
* **Corruption Strategies:**
  1. **Token Masking:** Randomly mask tokens
  2. **Permutation:** Shuffle token order (e.g., d e a b c)
  3. **Rotation:** Rotate sequence (e.g., c d e a b)
  4. **Token Deletion:** Remove random tokens

**Workflow:**

* Corrupted text → Encoder
* Original text → Decoder (used to train next token prediction)

**Output:**

* Decoder produces hidden representations
* Passed through Softmax for token probabilities

**Summary**

* **Encoder-only (BERT):** Good for classification, uses MLM, bidirectional
* **Encoder-decoder (T5, BART):** Best for seq2seq, needs both components, allows multitask learning
* **Decoder-only (GPT):** Best for generation, efficient, autoregressive
* **BART pre-training:** Corrupt input + predict original = Self-supervised seq2seq training

**T5: Text-to-Text Transfer Transformer**

**Introduction to T5 (Text-to-Text Transfer Transformer)**

* Developed by Google, introduced around the same time as Facebook’s BART model.
* T5 aims to **unify all NLP tasks into a text-to-text format**, regardless of the task type (classification, translation, summarization, etc).
* The central philosophy: **any problem can be cast as a text-to-text problem**.
* The architecture is based on the **encoder-decoder Transformer model**.

**Core Idea**

* Every NLP task is framed as:
  + **Input**: Instructional prefix + text (e.g., "Translate English to German: How are you?")
  + **Output**: Target response (e.g., "Wie geht es dir?")
* This allows the model to treat tasks like classification, translation, regression, and summarization in a **uniform way**.
* Unlike BERT, which does not take task instructions explicitly, T5 uses a **task prefix**.

**Example Tasks in T5 Format**

1. **Translation**:
   * Input: "Translate English to German: How are you?"
   * Output: "Wie geht es dir?"
2. **Acceptability Classification (CoLA)**:
   * Input: "CoLA: The course is jumping well."
   * Output: "unacceptable"
3. **Semantic Textual Similarity (STS-B)**:
   * Input: "STS-B: A rhino grazed on the grass || A rhino is grazing in the field."
   * Output: "3.8" (string representation of score)
4. **Summarization**:
   * Input: "Summarize: State authorities dispatched emergency crews..."
   * Output: "Authorities sent crews after severe weather in Mississippi."

**Pre-training Strategy**

* Model is pre-trained on **C4 Corpus** (Colossal Clean Crawled Corpus):
  + Derived from Common Crawl
  + **Aggressive preprocessing**: Remove non-English, HTML/JS, duplicates, noise
* **Corruption Strategy**:
  + Similar to denoising in BART
  + Mask **spans of text** instead of individual tokens
  + **Sentinel Tokens**: Each masked span is replaced with a unique sentinel token (e.g., <X>, <Y>)
  + Output is only the masked spans (not the entire sequence)

Example:

* Input: "Thank you me to your week."
* Output: " for inviting last"

**Architecture**

* **Standard Transformer encoder-decoder architecture**
* No major change in the base model structure
* Various model sizes trained, including BERT-base size for fair comparison

**Fine-tuning and Evaluation**

* Fine-tuned on a wide range of benchmark tasks:

1. **GLUE Benchmark**:
   * CoLA (acceptability)
   * SST-2 (sentiment analysis)
   * MRPC (paraphrase detection)
   * STS-B (semantic similarity)
   * QQP (duplicate question detection)
   * MNLI (entailment classification)
   * QNLI (question answering)
   * RTE (entailment detection)
   * WNLI (pronoun resolution)
2. **CNN/DailyMail** (summarization)
3. **SQuAD** (question answering)
4. **SuperGLUE Benchmark** (harder variant of GLUE):
   * Includes tasks such as RTE, WSC, MultiRC, ReCoRD, and BoolQ
5. **Machine Translation Benchmarks**:
   * WMT-14 (English-German, English-French)
   * WMT-15 (English-Romanian)

**Training Notes**

* Used C4 dataset for pre-training
* Fine-tuning was performed using multiple checkpoints
* Accuracy evaluated at each checkpoint for model selection
* Uses standard Transformer blocks with layer configurations similar to BERT-base and larger variants

**Summary**

* T5 offers a **unified approach** to solving NLP tasks via a text-to-text paradigm
* Handles **diverse task types** under a single model
* Leverages **sentinel tokens and span masking** during pre-training
* Shows strong performance across multiple benchmarks (GLUE, SuperGLUE, SQuAD, etc.)

**Key Contributions**

* Proposes a **unified text-to-text framework**
* Introduces **C4 corpus** for clean web-scale pretraining
* Extends denoising objectives using **sentinel tokens** for span prediction
* Achieves **state-of-the-art performance** on several NLP tasks

**Reference**

* "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer" by Colin Raffel et al. (2020)

**Prefix Language Modeling and Decoder-Only Architectures:**

**Decoder-Only Language Models Overview**

**1. Modeling Strategy**

* **Objective**: Predict the next token given a sequence of previous tokens.
* **Formulation**: Auto-regressive modeling (P(x) = P(x1) \* P(x2|x1) \* ... \* P(xn|x1,...xn-1))
* **Architecture**: Decoder-only transformer block with masked self-attention.
* **Input**: A sequence of tokens x1, x2, ..., xi-1
* **Output**: Prediction of token xi

**2. Classification Perspective**

* Vocabulary size V -> model outputs a probability distribution over V classes.
* Linear layer + softmax converts hidden state to output distribution.
* Sampling from this distribution = classification over vocabulary.

**3. Comparison with Encoder-Only Models (e.g., BERT)**

| **Feature** | **Encoder-Only (BERT)** | **Decoder-Only (GPT)** |
| --- | --- | --- |
| Input | All tokens with one masked | All previous tokens |
| Prediction | Predict the masked token | Predict next token |
| Attention Mask | Bidirectional | Causal (left-to-right) |

**Evolution of GPT Models**

**GPT-1 (2018)**

* Parameters: 117M
* Layers: 12
* Hidden Dim: 768
* FFN Dim: 3072
* Tokenizer: Byte Pair Encoding (BPE)
* Data: BookCorpus (~7K books)
* Focus: Introduced decoder-only transformer for generative modeling

**GPT-2 (2019)**

* Parameters: 1.5B
* Context Window: Increased to 1024 tokens
* Tokenizer: Vocabulary expanded to 50K
* Data: Common Crawl (8M documents, excluding Wikipedia)
* Improvements:
  + Layer Normalization changes
  + Focus on text generation quality
  + Introduced zero-shot learning behavior

**GPT-3 (2020)**

* Parameters: 175B
* Layers: 96
* Attention Heads: 96
* Vector Dim: 12288
* Training FLOPs per day: Significantly high
* Observations:
  + In-context learning emerged (few-shot via prompting)
  + No fine-tuning needed, examples in prompt were sufficient
* Cost: Very high compute/training cost

**GPT-4 (2023)**

* Architecture and parameters: Undisclosed (Black-box model)
* Estimated Cost: 15–30x more than GPT-3 per token
* Features:
  + Instruction fine-tuning
  + Reinforcement Learning from Human Feedback (RLHF)
  + Multimodal capabilities assumed

**Instruction Fine-Tuning and RLHF**

* **Instruction Fine-Tuning**: Train model to follow user instructions (e.g., prompt-response format)
* **RLHF**: Refines model outputs based on human preferences using reinforcement learning

**Open Source Alternatives: LLaMA Series**

* **LLaMA 1/2/3**: Meta's open-source decoder-only models
* Parameters: Scales across versions
* Features:
  + Uses Rotary Positional Embeddings (RoPE)
  + BPE for tokenization
  + Diverse datasets: Common Crawl, GitHub, Wikipedia, Books

**Conclusion**

Decoder-only models (GPT family) represent powerful generative systems capable of language modeling through simple next-token prediction. Through scale and architectural innovations (e.g., layer normalization, larger context, RLHF), models like GPT-3 and GPT-4 demonstrate emergent behaviors such as in-context learning and instruction following.

Next step in the learning path: Instruction Fine-Tuning and Reinforcement Learning from Human Feedback (RLHF) for aligning models with human preferences.