Natural Language Processing (NLP)

Fine Tune Down Stream Task

GPT Architecture, Pre-training,

Fine-tuning & Text Generation

By:

Dr. Zohair Ahmed

www.youtube.com/@ZohairAI Subscribe

www.begindiscovery.com

Pre-training vs Fine-tuning: Why This Matters

- We distinguish two key phases:
  - **Pre-training:** Learn *language itself* using massive unlabeled text.
  - **Fine-tuning:** Adapt the pre-trained model to specific, labeled tasks.
- **Pre-training:**
  - No explicit labels; **next token is the label**.
  - **Objective:** maximize probability of each next token in long sequences.
- **Fine-tuning:**
  - Uses **small labeled datasets** for particular tasks (sentiment, QA, etc.).
  - Objective: map inputs to **task-specific labels** efficiently, leveraging pre-trained knowledge.

Pre-training Setup: Data and Objective

- **Data:** Large unlabeled corpus, tokenized into sequences.
- **Each sequence:**
  - Length up to maximum context (e.g., 512 tokens in the example).
- **Supervision:**
  - For each time step t, the **next token** x_{t+1} is the target label.
- **Objective (conceptually):**
  - For all sequences and all-time steps, **maximize**: $P(next\ token \mid all\ previous\ tokens)$
  - Equivalent to **minimizing negative log-likelihood** of correct next tokens.

Transition: From Language Modeling to Tasks

After pre-training:

- Model is **not** trained for any specific task yet.
- It only knows **how to predict the next token** well.

But we want:

- **Sentiment analysis**.
- **Text classification**.
- **Textual entailment**.
- **Question answering**.
- **Text generation / story completion**, etc.

Strategy:

- Use the **pre-trained model as a backbone**.
- Adapt it to downstream tasks via **fine-tuning** with labeled examples.
- **Pre-training** is like giving the model general linguistic knowledge.
- **Fine-tuning** then turns that into actionable skill for particular tasks.

Fine-tuning: Overall Setup

- **Inputs:**
  - A labeled dataset specific to some task:
  - **Example:** (review_text, sentiment_label).
- **Model:**
  - Take the **entire pre-trained Transformer** as a **backbone**.
  - Keep its ability to process sequences up to max length (e.g., 512).
- **Add a task-specific head:**
  - Typically, a **linear classification layer** on top of the final token representation.
  - Then apply **softmax** to get class probabilities.
- We reuse the heavy millions/billions of parameters from pre-training and only add a small number of extra parameters in the final layer for classification.

Classification Head: From 768 to Classes

- Suppose final token representation dimension is **768**.
- **Add a weight matrix W:** Shape: 768 × C, where C = number of classes.
- Compute logits:
  - $z = W^T h_m$
  - $z \in \mathbb{R}^C$
- Apply **softmax**:
  - p = softmax(z) gives class probabilities.
- Examples:
  - **Binary sentiment**: C = 2 (positive vs negative).
  - **Five-class sentiment**: very positive, positive, neutral, negative, very negative (C = 5).
- This W is newly introduced and is randomly initialized at the beginning of fine-tuning, then trained using the labeled data.

Fine-tuning Objective: Cross-Entropy over Classes

- For each input–label pair (x, y):
  - Model outputs probabilities p(y | x).
- Loss:
  - **Cross-entropy** between predicted distribution and true label.
  - Concretely: minimize $-\log p_{correct\_class}$.
- Training:
  - Sum (or average) this loss over all training examples.
  - Backpropagate to:
    a. Update W **(classification head)**.
    b. Optionally update **backbone parameters** (θ of Transformer).
- Mathematically similar to standard classification training; the difference is that most of the layers already have meaningful weights from pre-training.

Freezing vs Updating Layers

- Options during fine-tuning:
  - **Freeze entire backbone**:
    a. Do not update any pre-trained Transformer weights.
    b. Only train the new classification layer W.
  - **Partial freezing**:
    a. Freeze **first K layers**.
    b. Fine-tune only **last L − K layers** plus W.
  - **Full fine-tuning**:
    a. Allow gradients through **all Transformer layers** and W.

- Trade-offs:
  - More layers updated → more flexibility but **higher risk of overfitting** and more compute.
  - Freezing more layers → **less compute & simpler**, but possibly lower task performance.
- In practice, people experiment with freezing strategies to strike a balance between stability and task performance, especially when labeled data is small.

Variety of NLP Tasks via Fine-tuning

- Tasks:
  - **Sentiment analysis**.
  - **Multiple-choice QA**.
- Common pattern:
  - Decide how to **format inputs** (one or two sequences, separators).
  - Decide **what to read from outputs** (final token, etc.).
  - Add a **simple output head** (linear layer + softmax).
  - Fine-tune on **labeled data** for that task.
- Pre-training + fine-tuning:
  - Turn a generic next-token predictor into a **versatile NLU/NLG engine**.

Determinism and Lack of Variety

- **Deterministic generation**:
- All computations in the Transformer are **fixed** once weights are frozen.
- For a given prompt:
- The distribution over the next token is always the same.
- If we always pick the **argmax token**, we always get the **same next token**.
- **Consequence:**
  - For any prompt (e.g., “Once upon a time there was a king”):
    a. The entire 512-token continuation is **identical every time**.
- Problem:
  - No **variety**, no **creative diversity** in output.
- This is undesirable for many applications where you want multiple possible completions or more human-like variability.

Wish List for Good Text Generation

- We want generated text that is:
- **Non-degenerative** (no meaningless loops).
- **Non-repetitive** (avoid “I like to think that I like to think that…”).
- **Coherent** (semantically and syntactically consistent).
- **Creative** (different plausible outputs for the same prompt).
- **Bad examples:**
  - Pure repetition: “I like to think that I like to think that…”.
  - Incoherent: “I like to think that reference know how to think bestselling book.”
- **Good behavior:**
  - For prompt “I like …”, model might generate:
    a. “… to read a book.”
    b. “… to buy a hot beverage.”
    c. “… a person who cares about others.”
-

Need for Decoding Strategies

- **Training** (next-token prediction) is separate from **decoding** (how we choose outputs from predicted distributions).
- Without special decoding:
  - Always picking max probability token
    → **deterministic, low-variety outputs**.
- With proper decoding strategies:
  - We can:
    a. Introduce **stochasticity**.
    b. Balance **repetition avoidance** and **coherence**.
    c. Satisfy the wish list (creative but not nonsensical).
- Next step (beyond this lecture):
  - Study **decoding strategies** (sampling, top-k, nucleus, etc.) to achieve desired behavior.
-

Evaluating NLP Systems

Evaluating NLP Systems:

- Different tasks require **different evaluation metrics**:
- **Classification:** accuracy, precision, recall, F1.
- **Translation:** BLEU, METEOR, etc.
- **Clustering:** internal and external cluster quality scores.
- **Question answering:** exact match, span overlap.
- **Next-word prediction / language modeling:** perplexity.
- **Text generation:** automatic metrics + human evaluation.
- Key idea:
- Metrics should reflect the **true goal** of the task (correctness, fluency, coherence, usefulness).
- Before diving into each task, it's important to recognize that no single metric works for everything.

Classification Metrics

- Typical tasks: **sentiment analysis**, **text classification**, **textual entailment**.

Common metrics:

- **Accuracy:** fraction of all predictions that are correct.
- **Precision:** among items predicted as positive (or a class), how many are truly that class.
- **Recall:** among truly positive items, how many we correctly predicted as positive.
- **F1-score:** harmonic mean of precision and recall; balances false positives and false negatives.

Multi-class settings:

- Use **macro-averaged** and **micro-averaged** F 1 to summarize performance across all classes.
- For balanced datasets, accuracy can be fine.
- For imbalanced datasets (e.g., rare class), you look more carefully at precision, recall, and F1.

Metrics for Machine Translation

- **Task:** Convert text from **source language** to **target language**.
- Classical automatic metrics:
- **BLEU (Bilingual Evaluation Understudy):**
  - Based on **n-gram overlap** between system output and one or more reference translations.
  - Includes a **brevity penalty** to discourage overly short translations.
- **METEOR (Metric for Evaluation of Translation with Explicit ORdering), TER (Translation Edit Rate)**
- Also use alignment, synonyms, edit distance.
- **Limitations:**
  - High n-gram overlap does not always mean **good fluency or adequacy**.
  - Poor at capturing **paraphrases** or deeper semantic equivalence.
- When using GPT-like models for translation, BLEU is standard but should be combined with human judgments for serious evaluation.

Metric	Focus	Higher Better?
BLEU	n-gram precision	Yes
METEOR	Meaning + order	Yes
TER	Edit distance	No (lower is better)

How does BLEU work?

- BLEU is based on **n-gram precision** and a **brevity penalty**:
- **N-gram Matching**
  - It checks how many words or sequences of words (called n-grams) in the candidate translation appear in the reference translation.
  - **Example:**
    a. Reference: *"The cat is on the mat"*
    b. Candidate: *"The cat is on mat"*
    c. Unigrams (1-grams): The, cat, is, on, mat
    d. Matches: 4 out of 5 → Precision = 0.8
- **Multiple n-grams**
  - BLEU uses **1-gram, 2-gram, 3-gram, and 4-gram precision**.
  - Higher n-grams capture fluency and word order.
- **Brevity Penalty (BP)**
  - If the candidate is **too short**, BLEU penalizes it.
  - Formula: $BP = \begin{cases} 1 & if\ c > r \\ e^{(1-r/c)} & if\ c \leq r \end{cases}$ where:
    a. c= length of candidate
    b. r= length of reference
- **Final BLEU Score**
  - Combine n-gram precisions with geometric mean: $BLEU = BP \times \exp(\sum_{n=1}^{N} w_n \log p_n)$ where:
    a. p_n =precision for n-grams
    b. w_n =weight (usually equal for 1-4 grams)

Question Answering Metrics

- **Types of QA:**
  - **Span-based QA** (answer is a span in a passage).
  - **Multiple-choice QA**.
  - **Free-form generation QA**.
- **Span-based metrics:**
  - **Exact Match (EM):** 1 if predicted span text matches the gold answer string exactly, else 0.
  - **F1-score at token level:** measures overlap between predicted and gold answer tokens.
- **Multiple-choice QA:**
  - **Accuracy:** proportion of questions where the correct choice is selected.
- **Free-form generation:**
  - Compare to gold answers via **string similarity** or **semantic similarity**, but often rely on **human evaluation** for correctness.
- QA, EM and F1 are common metrics (e.g., SQuAD Stanford Question Answering Dataset).
- For multiple-choice, the formulation is just classification over answer options.

Question Answering Metrics

- **1. Exact Match (EM)**
- **Definition:** Percentage of predictions that match the reference answer **exactly**.
- **Formula:**

$$EM = \frac{Number\ of\ exact\ matches}{Total\ questions} \times 100$$

- **Question:**
  What is the capital of France?
- **Reference Answer:**
  Paris
- **Candidate Answer:**
- Case 1: Paris → **Exact Match = 1 (correct)**
- Case 2: paris → Usually normalized (lowercased), so **Exact Match = 1**
- Case 3: Paris city → **Exact Match = 0 (not identical)**
- Case 4: The capital of France is Paris → **Exact Match = 0**
- **Why so strict?**
- EM checks if the **predicted answer string exactly equals the reference string** after normalization (lowercase, remove punctuation, sometimes articles).

Metrics for Next-Word / Next-Token Prediction

- Core pre-training objective: **next-token prediction** in language modeling.
- Primary metric:
  - **Perplexity (PPL):**
    a. Exponential of the average negative log-likelihood per token.
    b. Intuition: lower perplexity → model is **less "perplexed"** by the data.
- Relationship to training loss:
  - If L is the average cross-entropy loss per token,
    a. Perplexity = exp(L).
- Interpretation:
  - Good for comparing **language models** on the same dataset.
  - Lower perplexity generally means **better predictive modeling**, but does not guarantee "better" generation quality.
- Perplexity is closely tied to the loss function used during pre-training, so it's a natural way to track training progress and compare models.

Perplexity (PPL)

- **Example Sentence:** I love NLP
- **True tokens:** ["I", "love", "NLP"]
- **Model predicted probabilities for each next token:**
- For "I" → 0.5
- For "love" → 0.4
- For "NLP" → 0.2
- **Step 1: Compute Negative Log-Likelihood (NLL)**
- (using natural log) $N_{LLi} = -log\left(P(token_i)\right)$
- I: −log(0.5)=0.693
- love: −log(0.4)=0.916
- NLP: −log(0.2)=1.609
- **Step 2: Average Loss**
- $L = \frac{0.693+0.916+1.609}{3} = \frac{3.218}{3} \approx 1.073$
- **Step 3: Compute Perplexity**
- PPL $= e^{L} = e^{1.073} \approx 2.92$
- Perplexity = 2.92 → The model is about **3 times "confused"** on average for each token. Lower PPL = better prediction.

Text Generation Metrics (Beyond Perplexity)

- For **long-form generation** (stories, summaries, open-ended text):
  - No single perfect automatic metric.
- Common automatic metrics:
  - **Perplexity**: reflects next-token prediction quality, but may not correlate perfectly with **human judgments**.
  - **n-gram-based metrics** (BLEU/ROUGE-like) when we have reference texts.
- Qualitative and human-centric evaluation:
  - **Fluency:** grammatical correctness and naturalness.
  - **Coherence:** logical consistency across the entire passage.
  - **Relevance:** staying on topic with respect to the prompt.
  - **Diversity / non-repetition:** lack of loops or boring repetition.
- Often:
  - Use a **combination** of automatic metrics and **human annotation** to assess generation quality.

Connecting Metrics to Model Development

- **During pre-training:**
  - Focus on **loss** and **perplexity** on held-out data.
- During **fine-tuning for tasks**:
  - Choose metrics aligned with task:
    a. Classification / entailment → accuracy, F1, confusion matrix.
    b. QA → EM, token-level F1, or accuracy over choices.
    c. Translation → BLEU (plus human checks).
    d. Clustering over embeddings → clustering validity scores.
- **For text generation behavior**:
  - Track perplexity but also inspect:
    a. **Examples of generated text**.
    b. Human ratings for **fluency, coherence, and diversity**.
- Iterative cycle:
  - Metrics → identify weaknesses → adjust **architecture, fine-tuning strategy, or decoding**.