BERT Fine-Tuning

Module 2 of LLM Theory to Practice Series

CATAPANG JASPER KYLE

Tokyo University of Foreign Studies

2025/08/20

Overview

Today's journey: From concepts to hands-on fine-tuning

1. **Why it matters** — Relevance of BERT fine-tuning for linguistics, education, and digital humanities.
2. **Crash course** — Key components of BERT and related NLP concepts in plain language.
3. **Fine-tuning methods** — Prompting vs full fine-tuning vs PEFT/LoRA.
4. **Training recipe** — What goes into preparing and training a model well.
5. **Evaluation** — Looking beyond a single score; robust testing and fairness.
6. **Case studies** — GodziLLa-2 and EMoTES-3K.
7. **Hands-on activity** — Building and evaluating a BERT-based classifier.

Why This Matters for Language and Education

BERT and similar models are transforming how we study, teach, and work with language.

- **Linguistics research:** Discover new word usages, track language change, and analyze multilingual corpora.
- **Language teaching:** Create adaptive exercises, detect learner errors, and give tailored feedback.
- **Assessment:** Automatically score essays, oral exams, and comprehension tasks with high consistency.
- **Digital humanities:** Mine historical texts, annotate corpora, and link linguistic patterns to cultural context.

In short: Knowing how these models work helps you trust (or question) their outputs and apply them responsibly in your field.

Crash Course: BERT modules

- Tokenizer
- Special tokens
- Embeddings
- Encoder
- Pooling
- Head
- Length limit

Tokenizer

- Breaks input into manageable sub-units. Instead of treating “words” as indivisible, it splits them into frequent fragments, much like morphological decomposition.
- Handles unknown or rare forms by reverting to smaller, productive units. This avoids out-of-vocabulary failures when encountering new derivations, borrowings, or creative spellings.
- Example: “internationalization” → `intern` + `##ational` + `##ization`.
- This approach mirrors how humans interpret unfamiliar words: we parse pieces, recombine meaning, and generalize.

Special tokens

- BERT prepends and appends markers to signal discourse boundaries and task focus.
- `[CLS]` token serves as a container for the overall sentence-level meaning; it is the designated "summary position."
- `[SEP]` token divides distinct text segments (e.g., two sentences in entailment tasks or a question–answer pair).
- These markers are minimal interventions but crucial — they give structure and function to what would otherwise be an undifferentiated sequence of tokens.

Embeddings

- Each token is projected into a continuous vector space, a kind of “coordinate system” where proximity corresponds to similarity.
- Three layers of information are fused:
  - Token embeddings (identity of the word or subword)
  - Position embeddings (order in sequence)
  - Segment embeddings (which sentence/segment it belongs to).
- This allows the model to capture not only lexical meaning but also word order and discourse segmentation — a richer representation than text-as-words alone.

Encoder

- BERT's encoder is a stack of Transformer blocks using self-attention. Each word looks at all others simultaneously, not just neighbors.
- This setup makes it possible to capture both local dependencies (e.g., adjective–noun) and long-distance ones (e.g., subject–verb agreement across clauses).
- Attention patterns learned here function like data-driven syntax–semantics links: words adjust their interpretation depending on every other word in the sentence.
- The depth of multiple layers allows increasingly abstract representations, moving from lexical relations toward propositional meaning.

Pooling

- After encoding, we need a single vector to stand for the entire sentence or sequence.
- The [CLS] token embedding is the conventional choice: it has been trained to accumulate information from the whole sequence.
- Alternatives like mean pooling average across tokens, producing a more uniform “gist” representation.
- This step is comparable to how readers or listeners summarize a sentence into an overall judgment or meaning.

Head

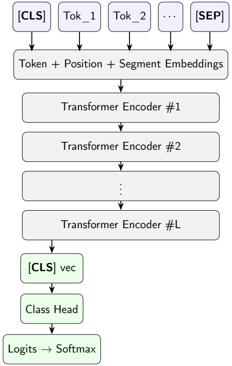
- A lightweight classifier (usually a linear layer) sits on top of the pooled representation.
- Its job: map high-dimensional representations into a concrete output — class label, score, or decision.
- Think of it as the interface between abstract meaning and categorical judgments. For example, deciding whether a sentence expresses "positive" or "negative" sentiment, or assigning a proficiency level.

Length limit

- BERT processes up to 512 tokens per sequence. Beyond that, text must be truncated or split.
- This creates a trade-off: longer discourse contexts risk being cut, but splitting can disrupt coherence.
- The limit reflects computational cost — attention grows quadratically with sequence length — but also echoes human working memory constraints.
- Later architectures (Longformer, BigBird, etc.) extend this window, enabling broader discourse modeling.

BERT 101

- **Tokenizer:** wordpiece; maps text → subwords + special tokens [CLS], [SEP].
- **Embeddings:** token + position + segment.
- **Encoder:** stacked Transformer blocks with self-attention.
- **Pooling:** use [CLS] (or mean-pool) for classification.
- **Head:** a small feed-forward layer for logits.
- **Length:** max sequence (usually 512); truncation strategy matters.



Assumption: "[CLS] pooling is always best."

Skeptic: Mean-pooling or attentive pooling can outperform on certain tasks/lengths.

Prompting vs Fine-Tuning (vis à vis BERT)

Prompting

- No training, quick iteration.
- Lower infra and governance load.
- Limited control; brittle formatting.
- May require retrieval for facts.

Fine-Tuning (BERT)

- Strong control over behavior.
- Predictable latency; smaller model.
- Requires labeled data + MLOps.
- Risk: overfit to style, label noise.

Assumption: Fine-tuning always outperforms prompting.

Skeptic: On small/dirty data, a frozen encoder + calibrated head or a prompt baseline can be competitive.

Takeaway: Treat prompting as capability *selection*, fine-tuning as capability *shaping*.

Task Shapes You Can Fine-Tune I

Common objectives

- **Single-label classification:** one class from K. Input: text. Output: $\arg\max$ softmax.
- **Multi-label classification:** multiple true classes. Output: sigmoid per class + thresholds.
- **Pair classification (e.g., NLI):** inputs (premise, hypothesis) with segment IDs.
- **Regression:** real-valued target; MSE/MAE loss; consider calibration.

Task Shapes You Can Fine-Tune II

Data and splits

- **Label schema:** unambiguous, consistent granularity; document guidelines.
- **Imbalance:** use stratified splits, class weights, or re-sampling; report macro-F1.
- **Leakage traps:** near-duplicates; same source across train/val; time-based splits when relevant.
- **Length profile:** histogram of tokens; informs max_len and chunk policy.

Assumption: IID splits reflect production.

Skeptic: Create slice/temporal splits to test robustness.

Crash Course: Class Imbalance

- **What it is:** When some categories in your data appear much more often than others. *Example:* 95% "cats" and only 5% "dogs" in an image dataset.
- **Why it's a problem:** The model can get "lazy" and mostly guess the majority class, still getting high accuracy but performing poorly on rare classes.
- **How to handle it:**
  - **Stratified splits:** Make sure each train/test split keeps the same proportion of each class.
  - **Class weights:** Tell the model to pay more attention to rare classes by giving them more "importance" during training.
  - **Re-sampling:** Balance the dataset by duplicating rare examples or reducing common ones.
- **How to report performance:** Use **macro-F1 score** — it treats all classes equally, so rare classes matter just as much as common ones.

Crash Course: Leakage Traps

- **Near-duplicates:** Multiple versions of essentially the same text (e.g., copied statutes, cloned clauses, paraphrases) can cause the model to "memorize" rather than generalize.
- **Source overlap:** Documents from the same author, case, or source appearing in both training and validation sets inflate apparent performance.
- **Temporal leakage:** Training on later data and validating on earlier data (e.g., laws or rulings issued after the test period) gives the model unfair hindsight.
- **Mitigations:** Deduplicate aggressively, enforce strict source separation, and design splits along time or domain lines when chronology or provenance matters.

Thresholds & Confidence

- **Setting the cut-off:** We choose a confidence level where the model will say "yes" or "no." Example: Only mark as "positive" if the model is at least 70% sure.
- **Finding the best cut-off:** Test different values on validation data to get the best balance between correct "yes" and correct "no" answers.
- **Checking confidence:** Compare what the model thinks (its confidence) to how often it's actually right. If it's overconfident or underconfident, adjust it so the numbers match reality.
- **Knowing when to skip:** If the model's confidence is too low, it can choose not to answer instead of guessing.

Assumption: Even a small accuracy increase matters.

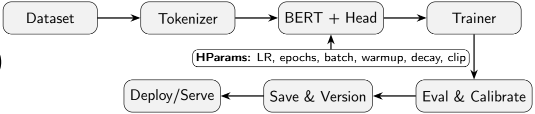
Skeptic: Check if the improvement is real — small changes may just be luck.

Crash Course: The Training Recipe

Think of training a model like cooking a dish:

- **Ingredients:** The dataset is your main ingredient. The better it is, the better the final dish.
- **Preparation:** The tokenizer chops text into pieces the model understands.
- **Cooking method:** The model (BERT + a task-specific head) learns patterns from the prepared ingredients.
- **Seasoning and heat:** Training settings like learning rate, batch size, and regularization keep the model from "overcooking" or "undercooking."
- **Taste test:** Evaluation checks if the model performs well on unseen data.
- **Final plating:** Save the trained model, record the settings, and get it ready to serve (deploy).

The Actual ML Training Recipe



Ingredients

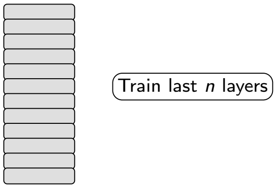
- Optimizer: **AdamW** (β_1=0.9, β_2=0.999, $\epsilon \sim 10^{-8}$)
- Schedule: warmup (3–10%) + linear decay
- LR band: $\{2 \cdot 10^{-5}, 3 \cdot 10^{-5}, 5 \cdot 10^{-5}\}$
- Batch: 16–32 (or grad accumulation)
- Reg: weight decay ~ 0.01, dropout, grad clip
- Early stopping on macro-F1
- Seeds + deterministic flags

Assumption: Small tweaks to settings are enough.

Skeptic: If the data is bad, no amount of fine-tuning will fix it.

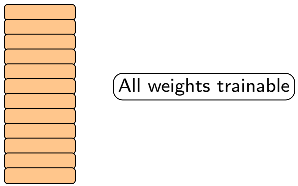
Freezing vs Full FT vs PEFT

Frozen Encoder



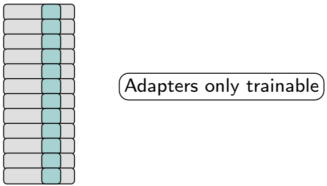
- Fastest, lowest VRAM
- Strong baseline
- Ceiling limited

Full Fine-Tune



- Best capacity
- Most costly
- Overfit risk

PEFT (e.g., LoRA)



- 1–10% trainable params
- Great for iteration
- Slight cap vs full FT

Assumption: Full FT is the default.

Skeptic: If PEFT reaches ∼98% of full FT at ∼10% of the cost, prefer PEFT unless you *need* the extra 2%.

GodziLLa-2

My own LLM project — ranked #2 worldwide (Open LLM Leaderboard, Nov 2023)

- **Origin:** Built at Maya Philippines, based on Meta AI's **Llama 2**.
- **Goal:** Push the limits of *instruction-tuned* LLMs for accuracy, reasoning, and reliability.
- **Training method: PEFT LoRA** (Parameter-Efficient Fine-Tuning using Low-Rank Adapters).
  - Trained domain-specific adapters (Finance, Instructions, Commonsense Reasoning, etc.).
  - Merged them into a composite model.
- **Data:** Combination of **Maya-curated datasets** and public sources.
- **Capabilities:** Enhanced truthfulness, instruction following, question answering, and basic reasoning.
- **Achievements:**
  - #2 worldwide on Open LLM Leaderboard (70B version).
  - Beat ChatGPT at the time (GPT-3.5), on TruthfulQA and HellaSwag benchmarks. Beat GPT-4 also on some other benchmarks.

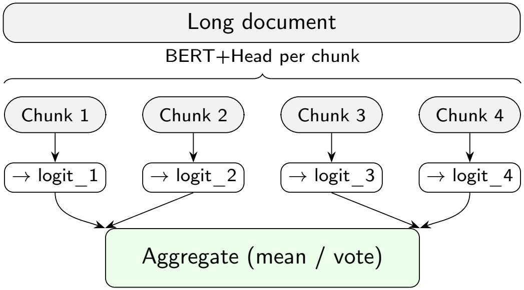
Assumption: Bigger models always win.

Skeptic: Smart fine-tuning can outperform raw scale.

Length Handling & Document Policy

Policies

- **Truncate**: simplest; risk of losing signal.
- **Sliding windows**: stride over text; aggregate.
- **Chunk-&-vote**: majority/mean of chunk logits.
- **Representation pooling**: mean of final hidden states across chunks.
- **Curriculum**: start short → longer sequences later.



Assumption: Sentence-level metrics reflect doc usage.

Skeptic: Report doc-level metrics when users read documents.

For long transcripts, this means you may need to split them into segments before analysis — otherwise, key details might be dropped.

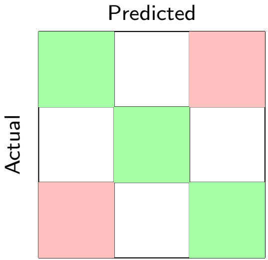
Evaluation Beyond a Single Number I

Look at the *full picture* of performance, not just one score

- Show performance for each category, not just the average.
- Break results down into "slices" — e.g., short vs long text, different sources, time periods, or language styles.
- Test how it handles "messy" data — typos, reworded sentences, or slightly changed inputs.
- Check how confident the model is in its answers, and adjust the decision threshold accordingly.
- Include a measure of uncertainty — e.g., by running the test multiple times.

Evaluation Beyond a Single Number II

Minimal confusion matrix



Selective prediction

- Sometimes, it's better for the model to say "I'm not sure" than give a bad answer.
- Keep track of how often the model chooses to answer vs skip, and how accurate it is in each case.
- Use the model's confidence scores to decide when to trust it.

Assumption: A single score tells the whole story.

Skeptic: Decision-makers care more about *where* it fails than the overall number.

Memorization, Privacy, and Reproducibility I

Memorization & privacy

- Data provenance and licenses; avoid sensitive content leakage.
- Reduce steps/epochs on small data; stronger regularization.
- Prefer smaller trainable sets (PEFT) when feasible.
- Consider DP techniques if required (with utility trade-offs).

Memorization, Privacy, and Reproducibility II

Reproducibility & governance

- Fix seeds; enable deterministic ops where practical.
- Version *everything*: data snapshot, code, hparams, tokenizer, model weights.
- Log artifacts and metrics; store training/eval configs with the checkpoint.
- Guard against contamination: deduplicate; source-aware or time-based splits.
- Recertify on a schedule; monitor drift and recalibrate thresholds.

Assumption: "We'll fix it in post."

Skeptic: Governance demands you fix it *in data and process*, not just in the UI.

Related Work: EMoTES-3K

Emotion-based Morality in Tagalog and English Scenarios (EMoTES-3K)
Catapang & Visperas, 2023

- **Goal:** Capture commonsense morality judgments in both Filipino and English.
- **Dataset:** 3,000 parallel scenarios labeled for moral/immoral actions *and* their emotional justifications.
- **Models used:**
  - **Classification:** Fine-tuned RoBERTa — 94.95% (EN), 88.53% (FIL).
  - **Generation:** FLAN-T5 — clear moral explanations, struggled with mixed-morality cases.
- **Relevance here:**
  - Showcases practical fine-tuning of Transformer models (similar to BERT architecture).
  - Highlights cross-lingual, low-resource challenges relevant to our discussion.

Key Takeaway

EMoTES-3K bridges the gap in moral reasoning resources for Filipino while demonstrating the adaptability of Transformer-based models for culturally specific NLP tasks.

Hands-on Activity

We will implement a **BERT-based classification** pipeline using:

- **Dataset:** CEFR classification sample data
- **Framework:** `Python` and `Huggingface`
- **Model:** Pre-trained BERT variant (base, uncased)
- **Platform:** Google Colab

Goal

Fine-tune BERT to classify CEFR label efficiently, then evaluate and interpret results.