# TÓM TẮT NGẮN (Executive summary)

Bạn đang dùng **PhoBERT-large** (kiểu RoBERTa-large) cho **phân loại cảm xúc nhị phân** (pos/neg) trên tiêu đề tài chính tiếng Việt. Pipeline hoạt động như sau:

1. **Word segmentation**: tách từ đa âm tiếng Việt (ví dụ *Ngân_hàng, nợ_xấu*).
2. **Tokenization (BPE)**: biến câu đã tách từ thành các **subword**, chèn token đặc biệt `<s>` và `</s>`, rồi **pad/truncate** về độ dài cố định (128).
3. **Mã hoá đầu vào**: tạo `input_ids` (ID token) và `attention_mask` (đánh dấu pad).
4. **Embedding**: mỗi token → **vector 1024 chiều** (token embedding + positional embedding).
5. **Encoder 24 lớp Transformer**: qua 24 khối *Self-Attention → FFN* để tạo **vector ngữ cảnh hoá** cho từng token.
6. **Đại diện câu**: lấy **vector của `<s>`** ở lớp cuối như *tóm tắt toàn câu*.
7. **Head phân loại**: *Dropout → Dense(1024→1024)+tanh → Dropout → Dense(1024→2)* → **logits**.
8. **Softmax & quyết định**: chuyển logits thành xác suất pos/neg; **argmax** để ra nhãn.
9. **Huấn luyện**: tối ưu Cross-Entropy có **class weights**, AdamW, warmup, FP16, early-stopping; chọn mô hình theo **F1-macro** trên validation.

---

# GIẢI THÍCH CHI TIẾT THEO BƯỚC

## Bước 1 — Word segmentation (tách từ)

- **Vì sao cần?** Tiếng Việt dùng khoảng trắng giữa **âm tiết**, không phải "từ". Từ đa âm (như *Ngân hàng*) sẽ bị chia nhỏ nếu không ghép lại → mô hình hiểu sai ngữ nghĩa.
- **Cách làm**: dùng bộ tách từ (thường là **VnCoreNLP / RDRSegmenter**).
- **Kết quả**: "Ngân hàng báo lỗ do nợ xấu tăng cao" → "**Ngân_hàng báo_lỗ** do **nợ_xấu tăng_cao**".

- **Ảnh hưởng**: giúp quá trình tokenization BPE sau đó tạo **subword hợp lý**, sát hơn với dữ liệu mà PhoBERT đã được pretrain.

### Thuật ngữ liên quan

- **Word segmentation**: ghép các âm tiết thành *từ đa âm* có nghĩa.
- **Token**: đơn vị xử lý (từ hoặc *subword*).
- **Subword**: mảnh con của từ dùng trong BPE để xử lý từ hiếm/chuỗi chưa gặp.

---

## Bước 2 — Tokenization BPE + token đặc biệt + pad/truncate

- **Tokenizer (BPE)**: thuật toán **Byte-Pair Encoding** học từ kho ngữ liệu lớn một **bảng gộp** ký tự/chuỗi ký tự thường đi cùng (`bpe.codes`). Nhờ đó, mọi từ (kể cả lạ) đều có thể biểu diễn thành vài **subword** quen.
- **Token đặc biệt** (chuẩn RoBERTa):
  - `<s>`: *bắt đầu câu* (tương tự `[CLS]` của BERT),
  - `</s>`: *kết thúc câu*,
  - `<pad>`: token *đệm*, để câu ngắn đủ dài.
- **Chuẩn hoá độ dài**:
  - **Truncation**: nếu >128 token thì cắt bớt (thường giữ đầu câu).
  - **Padding**: nếu <128, chèn `<pad>` cho đủ 128.
- **Đầu ra**:
  - `input_ids` (mảng số ID của token),
  - `attention_mask` (1: token thật, 0: pad).

### Thuật ngữ liên quan

- **BPE (`bpe.codes`)**: bảng luật gộp để tạo **subword**.
- `vocab.txt`: từ điển ánh xạ subword → ID.
- **Padding/Truncation**: đệm/cắt để mọi câu có cùng **độ dài L=128**.
- **Attention mask**: cho Transformer biết **bỏ qua** phần pad khi tính attention.

---

# Bước 3 — Embedding: từ ID → vector 1024 chiều

Mỗi token tại vị trí *i* được biểu diễn bằng
**token_embedding[id] + positional_embedding[i] → vector 1024**.

- **Token embedding**: bảng tra cứu học được trong pretrain (mang ý nghĩa từ/subword).
- **Positional embedding**: thêm thông tin **thứ tự** (Transformer không tự biết trật tự như RNN).
- **Dạng tensor**: với batch size **B**, câu dài **128**, hidden **1024** → **[B, 128, 1024]**.

### Thuật ngữ liên quan

- **Embedding**: biểu diễn dense nhiều chiều của token/vị trí.
- **Hidden size**: số chiều của không gian biểu diễn (PhoBERT-large: **1024**).
- **Batch size**: số câu xử lý song song trong một bước (ví dụ B=16/32).

---

# Bước 4 — 24 lớp Transformer Encoder: “ngữ cảnh hoá” token

Mỗi **lớp** gồm hai khối chính:

1. **Multi-Head Self-Attention**
    - “Self-Attention” cho phép **mỗi token** “nhìn” **toàn bộ** các token khác để học **quan hệ phụ thuộc** (ai liên quan ai, mức độ bao nhiêu).

    - **Multi-Head**: tách không gian 1024 thành **16 “đầu”** (head), mỗi đầu học một kiểu quan hệ khác (cú pháp, ngữ nghĩa, phủ định, thời gian…).
    - **Mask** bảo đảm pad không ảnh hưởng: những vị trí pad bị chặn ra khỏi tính attention.
2. **Feed-Forward Network (FFN)**
    - Hai lớp tuyến tính (thường *1024 → 4096 → 1024*) với kích hoạt **GELU**, đóng vai trò “trộn” và “phi tuyến hoá” đặc trưng sau attention.

Cả hai khối đều kèm:

- **Residual connection** (nối tắt) giúp gradient ổn định,
- **LayerNorm** (chuẩn hoá) giúp hội tụ tốt,
- **Dropout** giảm overfitting.

## Vì sao “ngữ cảnh hoá”?

Sau nhiều lớp, **vector của mỗi token** không chỉ chứa nghĩa “bản thân” nó, mà còn “nghĩa trong **bối cảnh**” cả câu (ai bổ nghĩa cho ai, phủ định, nguyên nhân-hậu quả, …). Do đó, vector của các token **khác nhau** và **giàu ngữ cảnh**.

## Tại sao PhoBERT-large dùng 24 lớp & 16 heads?

- **Sâu hơn / nhiều head hơn** → khả năng mô hình hoá **quan hệ phức tạp** tốt hơn, đặc biệt hữu ích cho câu dài/ý phức.
- Đánh đổi: **tài nguyên** (VRAM) và **thời gian huấn luyện** tăng.

## Thuật ngữ liên quan

- **Self-Attention**: cơ chế tính “ai chú ý đến ai” trong câu.
- **Multi-Head**: nhiều “góc nhìn” song song.
- **Residual/Skip connection**: cộng đầu vào với đầu ra lớp con để giữ thông tin & ổn định gradient.
- **LayerNorm**: chuẩn hoá kích hoạt theo feature để học dễ hơn.
- **GELU**: hàm kích hoạt mượt, hiệu quả cho Transformer.

---

## Bước 5 — Đại diện câu: vector `<s>` ở lớp cuối

- Theo chuẩn RoBERTa/BERT, lấy **vector của token `<s>`** (vị trí đầu tiên) ở **lớp cuối cùng** làm **đại diện toàn câu**.
- Lý do: trong huấn luyện phân loại, **head** đặt trực tiếp lên vector `<s>`, **loss** back-prop "ép" vector này học cách **tóm tắt toàn bộ nội dung câu** để phục vụ phân loại.
- **Kích thước**: **[B, 1024]** (mỗi câu → 1 vector 1024 chiều).

> Ghi chú: có các chiến lược pooling khác (mean/max pooling toàn chuỗi), nhưng chuẩn RoBERTa dùng `<s>` và hoạt động rất tốt trên nhiều tác vụ.

---

## Bước 6 — Head phân loại nhị phân

Chuỗi thao tác (trên vector `<s>`):

**Dropout → Dense(1024→1024) + tanh → Dropout → Dense(1024→2)**
→ thu được **logits** `[logit_neg, logit_pos]`.

- **Dropout**: chống overfitting.
- **Dense 1024→1024 + tanh**: học biến đổi *phi tuyến* để phân tách tốt hơn.
- **Dense 1024→2** (*Out_proj*): gom đặc trưng thành 2 điểm số (neg/pos).
- **Softmax** trên logits → xác suất **P(neg), P(pos)**; dự đoán = **argmax**.

### Thuật ngữ liên quan

- **Logits**: điểm số thô trước softmax.
- **Softmax**: chuẩn hoá logits thành phân phối xác suất.
- **Argmax**: chọn nhãn có xác suất lớn nhất.

---

## Bước 7 — Huấn luyện: tối ưu & chống lệch lớp

- **Mục tiêu (loss)**: Cross-Entropy có **class weights** để bù lệch nhãn (*pos* ≫ *neg*).

$$\mathcal{L} = -w_y \log P(y \mid x)$$

  *w_y* lớn hơn cho **neg** → mô hình "quan tâm" hơn đến lỗi với tin xấu.

- **Tối ưu**: **AdamW**, *weight decay* 0.01, **warmup** một phần nhỏ bước đầu để LR tăng dần rồi giảm.
- **FP16**: rút ngắn thời gian và tiết kiệm VRAM.
- **Early-Stopping**: dừng sớm khi không cải thiện (giảm overfitting).
- **Chọn mô hình**: `load_best_model_at_end=True`, theo **F1-macro** trên validation.

### Thuật ngữ liên quan

- **Class imbalance**: lệch lớp; **class weights** để cân bằng ảnh hưởng.
- **AdamW**: biến thể Adam có tách weight decay đúng cách.
- **Warmup**: khởi động LR mượt để ổn định huấn luyện.
- **Overfitting**: học quá kỹ train set → kém tổng quát; dùng dropout, early-stopping, regularization để tránh.

---

## Bước 8 — Suy luận (Inference)

- **Chuẩn bị**: bật `eval`, tắt dropout, không tính gradient.
- **Pipeline**: word segmentation → BPE + `<s>`,`</s>` + pad/truncate → embedding → 24 lớp Transformer → lấy `<s>` → head phân loại → softmax → **pos/neg**.
- **Giải thích xác suất**: bạn hay in `prob_pos` = P(pos). Cũng có thể điều chỉnh **ngưỡng** khác 0.5 nếu muốn ưu tiên phát hiện **neg** (ví dụ rủi ro tài chính).

---

# VÍ DỤ MINH HOẠ

**1) “FPT công bố lợi nhuận tăng mạnh quý 3/2024”**

- Sau tách từ: “FPT công_bố lợi_nhuận tăng_mạnh quý 3/2024”
- Mô hình:
    - Tập trung (attention) vào *lợi_nhuận, tăng_mạnh* → vector `<s>` nghiêng về **tích cực**.
    - Kết quả: **pos** với `prob_pos` rất cao (~0.98).

**2) “Ngân hàng báo lỗ do nợ xấu tăng cao”**

- Sau tách từ: “Ngân_hàng báo_lỗ do nợ_xấu tăng_cao”
- Mô hình:
    - Attention liên kết *báo_lỗ* với *nợ_xấu, tăng_cao* → ngữ cảnh **tiêu cực**.
    - Kết quả: **neg** với `prob_pos` rất thấp (~0.03).

---

# GHI CHÚ THỰC HÀNH & GIỚI HẠN

- **Chất lượng tách từ** ảnh hưởng trực tiếp đến đầu vào tokenizer → ảnh hưởng hiệu năng.
- **Truncation** ở 128 token: chú ý câu quá dài có thể mất thông tin cuối câu.
- **Lệch lớp**: nếu tin “neg” quá ít, dù đã dùng weights, vẫn nên **bổ sung dữ liệu neg** thực tế.
- **Kiểm thử thực địa** (tin mới ngoài test set) quan trọng để đánh giá **tổng quát hoá**.
- **Khả năng giải thích**: có thể dùng *attention visualization* / *saliency* để minh hoạ token nào ảnh hưởng nhiều.

---

# TỔNG KẾT

PhoBERT-large xử lý headline theo chuỗi **chuẩn Transformer**:
*segmentation → BPE → embeddings → 24×(self-attention + FFN) → sentence vector* `<s>` *→ head phân loại → softmax*.
Nhờ **tách từ** và **self-attention đa đầu sâu 24 lớp**, vector `<s>` mang đủ *nghĩa + ngữ cảnh* để phân biệt **pos/neg** hiệu quả; kết hợp **class weights** và tối ưu thích hợp đã giúp mô hình của bạn đạt **F1-macro ~0.86 trên test**, phù hợp với mục tiêu phân loại cảm xúc tiêu đề tài chính tiếng Việt.

-