Giải thích chi tiết code huấn luyện PhoBERT Classifier

1. Import các thư viện cần thiết

python

```python
import torch
import torch.nn as nn
from transformers import AutoModelForSequenceClassification, AutoTokenizer
from torch.utils.data import Dataset, DataLoader
from sklearn.metrics import accuracy_score
import pandas as pd
import os
from utils import load_config, save_model
```

Giải thích:

- `torch`: Thư viện PyTorch chính để xây dựng và huấn luyện mô hình deep learning
- `torch.nn`: Module chứa các layer neural network (Linear, Dropout, Loss functions...)
- `transformers`: Thư viện Hugging Face để sử dụng các mô hình transformer như BERT, PhoBERT
- `Dataset, DataLoader`: Để quản lý và load dữ liệu theo batch
- `sklearn.metrics`: Để tính toán độ chính xác
- `pandas`: Để đọc và xử lý file CSV
- `utils`: File tự định nghĩa chứa các hàm tiện ích

Ví dụ: Giống như bạn chuẩn bị đồ nghề trước khi làm việc - torch là "búa tạ", transformers là "máy khoan chuyên dụng"

2. Class TextDataset - Chuẩn bị dữ liệu

python

```python
class TextDataset(Dataset):
    def __init__(self, data, tokenizer, max_len):
        self.sentences = data["comment"].values
        self.labels = data["label"].map({"POS": 0, "NEG": 1, "NEU": 2}).fillna(0).values
        self.tokenizer = tokenizer
        self.max_len = max_len
```

Giải thích từng dòng:

- `self.sentences = data["comment"].values`: Lấy cột "comment" từ DataFrame làm văn bản đầu vào
- `self.labels = data["label"].map(...)`: Chuyển đổi nhãn text thành số:
  - "POS" (tích cực) → 0
  - "NEG" (tiêu cực) → 1
  - "NEU" (trung tính) → 2
- `fillna(0)`: Nếu có nhãn thiếu, gán mặc định là 0 (POS)

Ví dụ dữ liệu:

```
comment: "Sản phẩm này rất tuyệt vời!"
label: "POS" → chuyển thành 0
```

python

```python
def __getitem__(self, idx):
    sentence = self.sentences[idx]
    inputs = self.tokenizer.encode_plus(
        sentence, max_length=self.max_len, padding="max_length",
        truncation=True, return_tensors="pt"
    )
    label = torch.tensor(int(self.labels[idx]), dtype=torch.long)
    return {
        "input_ids": inputs["input_ids"].squeeze(),
        "attention_mask": inputs["attention_mask"].squeeze(),
        "label": label
    }
```

Giải thích tokenization:

- `tokenizer.encode_plus()`: Chuyển văn bản thành số (token IDs) mà máy hiểu được
- `max_length=self.max_len`: Giới hạn độ dài tối đa (cắt nếu quá dài)
- `padding="max_length"`: Thêm padding nếu câu ngắn hơn max_length
- `truncation=True`: Cắt bớt nếu câu dài hơn max_length
- `return_tensors="pt"`: Trả về tensor PyTorch

Ví dụ tokenization:

```
Input: "Tôi thích sản phẩm này"
→ token_ids: [101, 1234, 5678, 9012, 3456, 102]
→ attention_mask: [1, 1, 1, 1, 1, 1] (1 = token thật, 0 = padding)
```

3. Class PhoBERTClassifier - Mô hình AI

python

```python
class PhoBERTClassifier(nn.Module):
    def __init__(self, phobert_model, num_labels=3):
        super().__init__()
        self.phobert = phobert_model
        self.dropout = nn.Dropout(0.1)

    def forward(self, input_ids, attention_mask):
        outputs = self.phobert(input_ids, attention_mask=attention_mask)
        return outputs.logits
```

Giải thích:

- `nn.Module`: Lớp cha của tất cả mô hình neural network trong PyTorch
- `self.phobert = phobert_model`: Sử dụng mô hình PhoBERT đã được pre-train
- `nn.Dropout(0.1)`: Kỹ thuật regularization, tắt ngẫu nhiên 10% neurons để tránh overfitting
- `forward()`: Định nghĩa cách dữ liệu đi qua mô hình
- `outputs.logits`: Kết quả dự đoán thô (chưa qua softmax)

Ví dụ: Giống như một "bác sĩ AI" đã học từ hàng triệu văn bản tiếng Việt, giờ chuyên khoa phân loại cảm xúc

4. Hàm train_model - Huấn luyện mô hình

python

```python
def train_model(model, dataloader, optimizer, device, epochs=3):
    model.train()  # Chế độ huấn luyện
    for epoch in range(epochs):
        total_loss = 0
        for batch in dataloader:
            # Chuyển dữ liệu lên GPU/CPU
            input_ids = batch["input_ids"].to(device)
            attention_mask = batch["attention_mask"].to(device)
            labels = batch["label"].to(device)

            # Bước huấn luyện
            optimizer.zero_grad()  # Xóa gradient cũ
            outputs = model(input_ids, attention_mask)  # Dự đoán
            loss = nn.CrossEntropyLoss()(outputs, labels)  # Tính loss
            loss.backward()  # Tính gradient
            optimizer.step()  # Cập nhật trọng số
            total_loss += loss.item()
        print(f"Epoch {epoch+1}, Loss: {total_loss / len(dataloader)}")
```

Giải thích từng bước:

1. `model.train()`: Bật chế độ huấn luyện (dropout hoạt động)
2. `optimizer.zero_grad()`: Xóa gradient từ batch trước (quan trọng!)
3. `outputs = model(...)`: Cho dữ liệu qua mô hình để dự đoán
4. `loss = CrossEntropyLoss()`: Tính độ sai lệch giữa dự đoán và thực tế
5. `loss.backward()`: Tính gradient (đạo hàm) để biết cần điều chỉnh như thế nào
6. `optimizer.step()`: Cập nhật trọng số của mô hình

Ví dụ quá trình học:

```
Epoch 1: Loss = 1.2 (mô hình còn "ngớ ngẩn")
Epoch 2: Loss = 0.8 (đã thông minh hơn)
Epoch 3: Loss = 0.4 (khá giỏi rồi)
```

5. Hàm evaluate_model - Đánh giá mô hình

python

```python
def evaluate_model(model, dataloader, device):
    model.eval()  # Chế độ đánh giá (tắt dropout)
    predictions, true_labels = [], []
    with torch.no_grad():  # Không tính gradient (tiết kiệm bộ nhớ)
        for batch in dataloader:
            input_ids = batch["input_ids"].to(device)
            attention_mask = batch["attention_mask"].to(device)
            labels = batch["label"].to(device)
            outputs = model(input_ids, attention_mask)
            preds = torch.argmax(outputs, dim=1)  # Lấy class có xác suất cao nhất
            predictions.extend(preds.cpu().numpy())
            true_labels.extend(labels.cpu().numpy())
    accuracy = accuracy_score(true_labels, predictions)
    return accuracy
```

Giải thích:

- `model.eval()`: Chuyển sang chế độ đánh giá (dropout tắt)
- `torch.no_grad()`: Không tính gradient để tiết kiệm bộ nhớ
- `torch.argmax(outputs, dim=1)`: Chọn class có điểm số cao nhất

Ví dụ dự đoán:

```
outputs = [0.1, 0.8, 0.1]  # [POS, NEG, NEU]
→ argmax = 1 (NEG - tiêu cực)
```

6. Main function - Chương trình chính

python

```python
if __name__ == "__main__":
    # Load cấu hình
    config = load_config()
    project_root = config["project_root"]
    data_path = os.path.join(project_root, "data", "processed", "augmented_test_2k.csv")
    model_path = os.path.join(project_root, "models", "phobert_best.pt")
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

Giải thích:

- `load_config()`: Đọc file cấu hình dự án
- `os.path.join()`: Nối đường dẫn file một cách an toàn
- `torch.device()`: Chọn GPU nếu có, không thì dùng CPU

python

```
# Đọc và chuẩn bị dữ liệu
data = pd.read_csv(data_path, on_bad_lines='skip')
tokenizer = AutoTokenizer.from_pretrained("vinai/phobert-base")
phobert = AutoModelForSequenceClassification.from_pretrained("vinai/phobert-base", num_labe
```

Giải thích:

- `on_bad_lines='skip'`: Bỏ qua các dòng CSV bị lỗi format
- `AutoTokenizer.from_pretrained()`: Tải tokenizer PhoBERT từ Hugging Face
- `AutoModelForSequenceClassification.from_pretrained()`: Tải model PhoBERT với 3 nhãn output

python

```
# Tạo dataset và dataloader
dataset = TextDataset(data, tokenizer, max_len=128)
dataloader = DataLoader(dataset, batch_size=16, shuffle=True)
```

Giải thích:

- `max_len=128`: Giới hạn độ dài câu tối đa 128 token
- `batch_size=16`: Xử lý 16 mẫu cùng lúc (tối ưu bộ nhớ và tốc độ)
- `shuffle=True`: Xáo trộn dữ liệu mỗi epoch

python

```
# Khởi tạo và huấn luyện
model = PhoBERTClassifier(phobert, num_labels=3)
model = model.to(device)
optimizer = torch.optim.Adam(model.parameters(), lr=2e-5)

train_model(model, dataloader, optimizer, device)
accuracy = evaluate_model(model, dataloader, device)
save_model(model, model_path)
```

Giải thích:

- `model.to(device)`: Chuyển mô hình lên GPU/CPU
- `Adam(lr=2e-5)`: Optimizer Adam với learning rate 0.00002 (nhỏ để không "phá" pre-trained weights)
- `save_model()`: Lưu mô hình đã huấn luyện

Tóm tắt quy trình

1. **Chuẩn bị dữ liệu**: CSV → Dataset → DataLoader
2. **Tải mô hình**: PhoBERT pre-trained → Thêm classifier layer
3. **Huấn luyện**: Forward pass → Tính loss → Backward pass → Cập nhật weights
4. **Đánh giá**: Test trên dữ liệu → Tính accuracy
5. **Lưu mô hình**: Để sử dụng sau này

Ví dụ thực tế: Giống như dạy một người đã biết tiếng Việt (PhoBERT) cách phân loại cảm xúc. Bạn đưa hàng nghìn ví dụ, người đó học cách nhận biết, rồi bạn kiểm tra xem học được bao nhiêu %.