Giải thích chi tiết code PhoBERT Sentiment Classification

Tổng quan

Code này xây dựng một mô hình AI để phân loại cảm xúc của văn bản tiếng Việt thành 3 loại:

- **POS (Positive)**: Tích cực
- **NEG (Negative)**: Tiêu cực
- **NEU (Neutral)**: Trung tính

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`: Framework deep learning chính
- `transformers`: Thư viện của Hugging Face để sử dụng các mô hình pre-trained như PhoBERT
- `pandas`: Xử lý dữ liệu dạng bảng (CSV)
- `sklearn.metrics`: Tính toán độ chính xác

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
```

Ví dụ cụ thể: Giả sử bạn có file CSV như sau:

```
comment,label
"Sản phẩm này rất tốt",POS
"Dịch vụ tệ quá",NEG
"Bình thường thôi",NEU
```

Class này sẽ:

1. Lấy cột "comment" làm văn bản đầu vào
2. Chuyển đổi nhãn: POS→0, NEG→1, NEU→2
3. Chuẩn bị tokenizer để chuyển text thành số

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
    }
```

Ví dụ tokenization:

- Input: "Sản phẩm này rất tốt"
- Output: `input_ids=[101, 7345, 2134, 1234, 5678, 102, 0, 0, ...]` (các số đại diện cho từ)
- `attention_mask=[1, 1, 1, 1, 1, 1, 0, 0, ...]` (1=từ thật, 0=padding)

3. Class PhoBERTClassifier - Mô hình chính

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:

- Sử dụng PhoBERT đã được pre-train trên dữ liệu tiếng Việt
- `dropout(0.1)`: Tắt ngẫu nhiên 10% neurons để tránh overfitting
- `logits`: Điểm số thô cho 3 class (chưa qua softmax)

Ví dụ output:

```
Input: "Sản phẩm tuyệt vời"
Logits: [2.1, -0.5, 0.3]  # [POS_score, NEG_score, NEU_score]
Prediction: POS (vì 2.1 là cao nhất)
```

4. Hàm train_model - Huấn luyện

python

```python
def train_model(model, dataloader, optimizer, device, epochs=3):
    model.train()
    for epoch in range(epochs):
        total_loss = 0
        for batch in dataloader:
            input_ids = batch["input_ids"].to(device)
            attention_mask = batch["attention_mask"].to(device)
            labels = batch["label"].to(device)

            optimizer.zero_grad()
            outputs = model(input_ids, attention_mask)
            loss = nn.CrossEntropyLoss()(outputs, labels)
            loss.backward()
            optimizer.step()
            total_loss += loss.item()
        print(f"Epoch {epoch+1}, Loss: {total_loss / len(dataloader)}")
```

Quy trình huấn luyện từng bước:

1. **Forward pass**: Đưa dữ liệu qua mô hình → nhận prediction
2. **Tính loss**: So sánh prediction với nhãn thật
3. **Backward pass**: Tính gradient (đạo hàm)
4. **Update weights**: Điều chỉnh trọng số để giảm loss

Ví dụ một batch:

```
Batch size = 2
Input: ["Tốt quá", "Tệ vãi"]
Labels: [0, 1]  # [POS, NEG]
Predictions: [[2.1, -0.5, 0.3], [-0.8, 1.9, 0.1]]
Loss: 0.25 (càng thấp càng tốt)
```

5. Hàm evaluate_model - Đánh giá

python

```python
def evaluate_model(model, dataloader, device):
    model.eval()
    predictions, true_labels = [], []
    with torch.no_grad():
        for batch in dataloader:
            # ... xử lý batch
            outputs = model(input_ids, attention_mask)
            preds = torch.argmax(outputs, dim=1)
            predictions.extend(preds.cpu().numpy())
            true_labels.extend(labels.cpu().numpy())
    accuracy = accuracy_score(true_labels, predictions)
    return accuracy
```

Ví dụ evaluation:

```
True labels:    [0, 1, 2, 0, 1]  # [POS, NEG, NEU, POS, NEG]
Predictions:    [0, 1, 2, 0, 0]  # [POS, NEG, NEU, POS, POS] - sai 1 cái
Accuracy: 4/5 = 0.8 = 80%
```

6. Main function - Chạy chương trình

python

```python
if __name__ == "__main__":
    # Load cấu hình và dữ liệu
    config = load_config()
    data = pd.read_csv(data_path, on_bad_lines='skip')

    # Khởi tạo tokenizer và model
    tokenizer = AutoTokenizer.from_pretrained("vinai/phobert-base")
    phobert = AutoModelForSequenceClassification.from_pretrained("vinai/phobert-base", nu

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

    # Huấn luyện và đánh giá
    model = PhoBERTClassifier(phobert, num_labels=3)
    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)
```

Ví dụ hoàn chỉnh với dữ liệu mẫu

Input CSV (augmented_test_2k.csv):

csv

```
comment,label
"Món ăn ngon tuyệt vời",POS
"Phục vụ chậm chạp",NEG
"Giá cả hợp lý",NEU
"Sẽ quay lại lần sau",POS
"Không đáng tiền",NEG
```

Quá trình xử lý:

1. Load dữ liệu → 5 samples
2. Tokenize text → chuyển thành số
3. Train 3 epochs:
   - Epoch 1: Loss = 1.2
   - Epoch 2: Loss = 0.8
   - Epoch 3: Loss = 0.5
4. Evaluate: Accuracy = 85%
5. Save model → file .pt

Các điểm quan trọng cần lưu ý

1. Hyperparameters

- `max_len=128`: Độ dài tối đa của câu (từ)
- `batch_size=16`: Số samples xử lý cùng lúc
- `lr=2e-5`: Learning rate (tốc độ học)
- `epochs=3`: Số lần duyệt qua toàn bộ dataset

2. GPU/CPU

python

```python
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

- Tự động chọn GPU nếu có, không thì dùng CPU
- GPU nhanh hơn CPU rất nhiều lần

3. Error Handling

python

```python
data = pd.read_csv(data_path, on_bad_lines='skip')
```

- Bỏ qua các dòng dữ liệu bị lỗi format

Cách mở rộng và ứng dụng

1. Thêm class mới

Muốn phân loại 5 cảm xúc: vui, buồn, tức giận, sợ hãi, bình thường

python

```python
# Thay đổi mapping
self.labels = data["label"].map({
    "HAPPY": 0, "SAD": 1, "ANGRY": 2, "FEAR": 3, "NEUTRAL": 4
}).fillna(4).values

# Thay đổi num_labels
phobert = AutoModelForSequenceClassification.from_pretrained(
    "vinai/phobert-base", num_labels=5
)
```

2. Sử dụng model đã train

python

```python
# Load model đã lưu
model = torch.load("models/phobert_best.pt")
model.eval()

# Predict cho text mới
def predict_sentiment(text):
    inputs = tokenizer.encode_plus(text, max_length=128,
                                   padding="max_length", truncation=True,
                                   return_tensors="pt")
    with torch.no_grad():
        outputs = model(inputs["input_ids"], inputs["attention_mask"])
        prediction = torch.argmax(outputs, dim=1).item()

    labels = {0: "Tích cực", 1: "Tiêu cực", 2: "Trung tính"}
    return labels[prediction]

# Test
result = predict_sentiment("Sản phẩm này thật tuyệt vời!")
print(result)  # Output: "Tích cực"
```

3. Fine-tuning cho domain khác

- **E-commerce**: Reviews sản phẩm
- **Social media**: Comments Facebook, Twitter
- **Customer service**: Phản hồi khách hàng
- **News**: Sentiment của tin tức

Chỉ cần thay đổi dữ liệu training, giữ nguyên kiến trúc mô hình!