Hướng dẫn chi tiết: Data Augmentation cho AI tiếng Việt

📚 Giới thiệu

Code này dùng để **tăng cường dữ liệu** (data augmentation) cho các bình luận tiếng Việt. Tức là từ 1 bình luận gốc, ta tạo thêm các phiên bản tương tự để có nhiều dữ liệu hơn để train AI.

🔧 Phần 1: Import thư viện

python

```python
import pandas as pd
from transformers import MarianMTModel, MarianTokenizer
import random
import py_vncorenlp
import torch
import os
```

Giải thích từng thư viện:

- `pandas`: Xử lý dữ liệu dạng bảng (CSV, Excel)
- `transformers`: Thư viện AI của Hugging Face (dù code không dùng đến)
- `random`: Tạo số ngẫu nhiên
- `py_vncorenlp`: Xử lý ngôn ngữ tự nhiên tiếng Việt
- `torch`: PyTorch (dù không dùng trực tiếp)
- `os`: Thao tác với hệ thống file

Ví dụ thực tế:

python

```python
# Pandas giúp đọc file CSV như này:
data = pd.read_csv("comments.csv")
print(data.head())  # Hiển thị 5 dòng đầu
```

🛠️ Phần 2: Khởi tạo VnCoreNLP

python

```python
py_vncorenlp.download_model(save_dir="C:/VnCoreNLP")
annotator = py_vncorenlp.VnCoreNLP(annotators=["wseg"], save_dir="C:/VnCoreNLP")
```

Giải thích:

- Tải model xử lý tiếng Việt về máy
- Tạo đối tượng `annotator` để tách từ (word segmentation)
- `wseg` = word segmentation (tách từ ghép tiếng Việt)

Ví dụ tách từ:

```python
# Input: "Sản phẩm rất tốt"
# Output sau tách từ: "Sản_phẩm rất tốt"
```

📖 Phần 3: Từ điển đồng nghĩa

```python
synonyms_dict = {
    "đẹp": ["xinh", "lộng lẫy", "mỹ miều"],
    "tuyệt vời": ["xuất sắc", "hoàn hảo", "tuyệt diệu"],
    "tốt": ["tuyệt", "ok", "hài lòng"],
    "nhanh": ["mau", "lẹ", "tốc độ"],
    "ổn": ["tốt", "được", "hài lòng"],
}
```

Giải thích:

- Dictionary (từ điển) chứa các từ đồng nghĩa
- Key (khóa): từ gốc
- Value (giá trị): danh sách các từ thay thế

Ví dụ sử dụng:

```python
# Câu gốc: "Sản phẩm rất đẹp"
# Có thể thay thành: "Sản phẩm rất xinh" hoặc "Sản phẩm rất lộng lẫy"
```

🔄 Phần 4: Hàm thay thế từ đồng nghĩa

python

```python
def synonym_replacement(comment):
    # Tách từ bằng VnCoreNLP
    segmented_text = annotator.word_segment(comment)
    if segmented_text:
        words = segmented_text[0].split()
    else:
        words = comment.split()

    # Tạo bản sao để thay đổi
    new_words = words.copy()

    # Duyệt qua từng từ
    for i, word in enumerate(words):
        # Nếu từ có trong từ điển VÀ random < 0.3 (30% khả năng)
        if word in synonyms_dict and random.random() < 0.3:
            # Chọn ngẫu nhiên 1 từ đồng nghĩa
            new_words[i] = random.choice(synonyms_dict[word])

    # Ghép lại thành câu
    return " ".join(new_words)
```

Giải thích từng bước:

1. **Tách từ:** Chia câu thành các từ riêng biệt
2. **Tạo bản sao:** Để không làm hỏng câu gốc
3. **Duyệt từng từ:** Kiểm tra từng từ một
4. **Kiểm tra điều kiện:**
   - Từ có trong từ điển không?
   - Random có < 0.3 không? (tức 30% cơ hội thay đổi)
5. **Thay thế:** Chọn ngẫu nhiên từ đồng nghĩa
6. **Ghép lại:** Tạo thành câu mới

Ví dụ chi tiết:

python

```python
# Input: "Món ăn rất tốt và nhanh"
# Bước 1: ["Món", "ăn", "rất", "tốt", "và", "nhanh"]
# Bước 2: Kiểm tra "tốt" -> có trong từ điển, random = 0.2 < 0.3 -> thay bằng "tuyệt"
# Bước 3: Kiểm tra "nhanh" -> có trong từ điển, random = 0.8 > 0.3 -> không thay
# Output: "Món ăn rất tuyệt và nhanh"
```

📈 Phần 5: Hàm tăng cường dữ liệu

python

```python
def augment_data(data):
    augmented_data = []

    # Duyệt qua từng dòng dữ liệu
    for _, row in data.iterrows():
        comment = row["comment"]  # Lấy bình luận
        label = row["label"]      # Lấy nhãn (positive/negative)
        rate = row["rate"]        # Lấy điểm đánh giá

        # Thêm cả bình luận gốc và bản thay đổi
        augmented_data.extend([
            {"comment": comment, "label": label, "rate": rate},  # Bản gốc
            {"comment": synonym_replacement(comment), "label": label, "rate": rate},  # Bản tha
        ])

    return pd.DataFrame(augmented_data)
```

Giải thích:

- Duyệt qua từng dòng trong dataset
- Với mỗi bình luận, tạo 2 phiên bản:
  1. Bản gốc (không thay đổi)
  2. Bản thay đổi (dùng từ đồng nghĩa)
- Label và rate giữ nguyên vì ý nghĩa không đổi

Ví dụ:

python

```python
# Dữ liệu gốc: 1000 bình luận
# Sau augmentation: 2000 bình luận (gấp đôi)

# Trước:
# "Sản phẩm tốt", positive, 5
# Sau:
# "Sản phẩm tốt", positive, 5      (bản gốc)
# "Sản phẩm tuyệt", positive, 5    (bản thay đổi)
```

🚀 Phần 6: Chạy chương trình chính

python

```python
if __name__ == "__main__":
    try:
        # Tạo đường dẫn file
        project_root = os.path.abspath(os.path.join(os.path.dirname(__file__), ".."))
        input_path = os.path.join(project_root, "data", "raw", "test_5k.csv")
        output_dir = os.path.join(project_root, "data", "processed")
        output_path = os.path.join(output_dir, "augmented_test_2k.csv")
```

Giải thích đường dẫn:

- `__file__`: File hiện tại (data_augmentation.py)
- `os.path.dirname(__file__)`: Thư mục chứa file hiện tại
- `".."`: Lùi lên 1 cấp thư mục
- `os.path.join()`: Ghép đường dẫn an toàn

Cấu trúc thư mục:

```
project/
├── scripts/
│   └── data_augmentation.py  (file này)
├── data/
│   ├── raw/
│   │   └── test_5k.csv      (file input)
│   └── processed/
│       └── augmented_test_2k.csv  (file output)
```

📁 Phần 7: Đọc và kiểm tra file

python

```python
        print(f"📄 Đang đọc file từ: {input_path}")
        if not os.path.exists(input_path):
            raise FileNotFoundError(f"Không tìm thấy file: {input_path}")

        data = pd.read_csv(input_path, usecols=["comment", "label", "rate"], on_bad_lines='skip
```

Giải thích:

- Kiểm tra file có tồn tại không
- Chỉ đọc 3 cột cần thiết: comment, label, rate
- `on_bad_lines='skip'`: Bỏ qua các dòng lỗi format

Ví dụ file CSV:

```csv
comment,label,rate
"Sản phẩm tốt",positive,5
"Giao hàng chậm",negative,2
"Chất lượng ổn",positive,4
```

✅ Phần 8: Kiểm tra cấu trúc và xử lý

```python
print("✅ Đọc file thành công, bắt đầu kiểm tra cấu trúc...")
required_columns = {"comment", "label", "rate"}
if not required_columns.issubset(data.columns):
    raise ValueError("❌ File CSV phải có đầy đủ các cột: comment, label, rate")

print("🚀 Cấu trúc hợp lệ, bắt đầu tăng cường dữ liệu...")
augmented_data = augment_data(data)

os.makedirs(output_dir, exist_ok=True)
augmented_data.to_csv(output_path, index=False)
```

Giải thích:

1. **Kiểm tra cột:** Đảm bảo có đủ 3 cột cần thiết
2. **Tăng cường dữ liệu:** Gọi hàm `augment_data()`
3. **Tạo thư mục:** Tạo thư mục output nếu chưa có
4. **Lưu file:** Xuất ra file CSV mới

🎯 Phần 9: Xử lý lỗi

```python
except Exception as e:
    print(f"❌ Lỗi: {e}")
```

Các lỗi có thể gặp:

- File không tồn tại
- Thiếu cột dữ liệu
- Lỗi format CSV
- Lỗi quyền ghi file

💡 Tóm tắt quy trình

1. **Chuẩn bị:** Tải VnCoreNLP, tạo từ điển đồng nghĩa
2. **Đọc dữ liệu:** Load file CSV gốc
3. **Tăng cường:** Tạo phiên bản mới bằng cách thay từ đồng nghĩa
4. **Lưu kết quả:** Export ra file mới với gấp đôi dữ liệu

🔥 Lợi ích của Data Augmentation

- **Tăng kích thước dataset:** Từ 1000 → 2000 mẫu
- **Cải thiện độ chính xác:** Model học được nhiều cách diễn đạt khác nhau
- **Giảm overfitting:** Model không nhớ cứng các cụm từ cụ thể
- **Tiết kiệm chi phí:** Không cần thu thập thêm dữ liệu thật

🛠️ Cách chạy code

1. Cài đặt thư viện:

bash

```
pip install pandas transformers py_vncorenlp torch
```

2. Chuẩn bị file CSV với 3 cột: comment, label, rate
3. Chạy script:

bash

```
python data_augmentation.py
```

4. Kiểm tra file output trong thư mục `data/processed/`