Hướng dẫn Hàm Thay thế Từ đồng nghĩa trong Python

📝 Tổng quan

Hàm `synonym_replacement()` được thiết kế để **thay thế một số từ trong câu bằng từ đồng nghĩa** một cách ngẫu nhiên. Điều này rất hữu ích trong việc:

- Tạo dữ liệu đa dạng cho machine learning
- Tránh việc lặp lại từ ngữ trong văn bản
- Tạo biến thể của câu gốc

🔍 Phân tích từng dòng code

Dòng 1-2: Định nghĩa hàm và tách từ

```python
def synonym_replacement(comment):
    segmented_text = annotator.word_segment(comment)
```

Giải thích:

- `def synonym_replacement(comment):` - Tạo hàm nhận vào một tham số `comment` (câu cần xử lý)
- `annotator.word_segment(comment)` - Sử dụng VnCoreNLP để **tách từ tiếng Việt** chính xác
- **Tại sao cần tách từ?** Tiếng Việt không có dấu cách giữa các từ ghép, ví dụ: "học sinh" là một từ, không phải hai từ riêng biệt

Dòng 3-6: Xử lý kết quả tách từ

```python
if segmented_text:
    words = segmented_text[0].split()
else:
    words = comment.split()
```

Giải thích:

- `if segmented_text:` - Kiểm tra xem việc tách từ có thành công không
- `segmented_text[0].split()` - Lấy kết quả tách từ và chia thành danh sách các từ
- `else: words = comment.split()` - Nếu tách từ thất bại, sử dụng cách chia đơn giản bằng dấu cách

Ví dụ:

- Input: "Tôi là học sinh giỏi"
- Sau tách từ: ["Tôi", "là", "học_sinh", "giỏi"]
- Sau split(): ["Tôi", "là", "học_sinh", "giỏi"]

Dòng 7: Tạo bản sao

```python
new_words = words.copy()
```

Giải thích:

- Tạo một **bản sao** của danh sách từ gốc
- **Tại sao cần copy?** Để tránh thay đổi trực tiếp vào danh sách gốc

Dòng 8-11: Vòng lặp thay thế từ

```python
for i, word in enumerate(words):
    if word in synonyms_dict and random.random() < 0.3:
        new_words[i] = random.choice(synonyms_dict[word])
```

Giải thích chi tiết:

`enumerate(words)`

- Trả về cả **chỉ số** (i) và **giá trị** (word) của từng phần tử
- Ví dụ: ["Tôi", "là", "giỏi"] → (0,"Tôi"), (1,"là"), (2,"giỏi")

`word in synonyms_dict`

- Kiểm tra xem từ hiện tại có trong **từ điển từ đồng nghĩa** không
- `synonyms_dict` có dạng: `{"giỏi": ["xuất sắc", "tốt", "khá"], "đẹp": ["xinh", "lung linh"]}`

`random.random() < 0.3`

- `random.random()` tạo số thập phân ngẫu nhiên từ 0.0 đến 1.0
- `< 0.3` có nghĩa là có **30% xác suất** thay thế từ
- **Tại sao không thay thế 100%?** Để giữ tính tự nhiên của câu

`random.choice(synonyms_dict[word])`

- Chọn **ngẫu nhiên** một từ đồng nghĩa từ danh sách
- Ví dụ: từ "giỏi" → có thể chọn "xuất sắc" hoặc "tốt" hoặc "khá"

Dòng 12: Trả về kết quả

```python
return " ".join(new_words)
```

Giải thích:

- `" ".join(new_words)` - Nối các từ trong danh sách thành một chuỗi, cách nhau bởi dấu cách
- Ví dụ: ["Tôi", "là", "xuất sắc"] → "Tôi là xuất sắc"

🎯 Ví dụ hoạt động

Dữ liệu đầu vào:

```python
synonyms_dict = {
    "giỏi": ["xuất sắc", "tốt", "khá"],
    "đẹp": ["xinh", "lung linh", "dễ thương"],
    "thông minh": ["lanh lợi", "sáng dạ"]
}

comment = "Cô ấy rất đẹp và thông minh"
```

Quá trình xử lý:

1. **Tách từ**: ["Cô", "ấy", "rất", "đẹp", "và", "thông_minh"]
2. **Kiểm tra từng từ**:
   - "Cô" → không có trong synonyms_dict → giữ nguyên
   - "ấy" → không có trong synonyms_dict → giữ nguyên
   - "rất" → không có trong synonyms_dict → giữ nguyên
   - "đẹp" → có trong dict + 30% xác suất → có thể thay bằng "xinh"
   - "và" → không có trong synonyms_dict → giữ nguyên
   - "thông_minh" → có trong dict + 30% xác suất → có thể thay bằng "lanh lợi"
3. **Kết quả có thể**: "Cô ấy rất xinh và lanh lợi"

⚙️ Các tham số có thể tùy chỉnh

Thay đổi tỷ lệ thay thế:

python

```python
# Thay thế 50% từ
if word in synonyms_dict and random.random() < 0.5:

# Thay thế 10% từ (ít thay đổi hơn)
if word in synonyms_dict and random.random() < 0.1:
```

Bỏ qua một số từ:

python

```python
skip_words = ["tôi", "bạn", "chúng ta"]  # Không thay thế đại từ
if word in synonyms_dict and word not in skip_words and random.random() < 0.3:
```

🚀 Cách sử dụng

python

```python
# Thiết lập
import random
import py_vncorenlp

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

synonyms_dict = {
    "tốt": ["giỏi", "xuất sắc", "khá"],
    "xấu": ["tệ", "dở", "không tốt"],
    # ... thêm các từ khác
}

# Sử dụng
original = "Học sinh này rất tốt"
modified = synonym_replacement(original)
print(f"Gốc: {original}")
print(f"Đã thay: {modified}")
```

💡 Ứng dụng thực tế

1. **Data Augmentation**: Tạo thêm dữ liệu huấn luyện cho AI
2. **Content Marketing**: Tạo nhiều phiên bản của cùng một nội dung
3. **SEO**: Tránh việc lặp từ khóa quá nhiều
4. **Chatbot**: Tạo phản hồi đa dạng hơn

⚠️ Lưu ý quan trọng

- **Từ điển đồng nghĩa** cần được chuẩn bị kỹ lưỡng
- **Ngữ cảnh** rất quan trọng - không phải lúc nào thay thế cũng phù hợp
- **Kiểm tra kết quả** sau khi thay thế để đảm bảo câu vẫn có nghĩa
- **Tỷ lệ thay thế** nên được điều chỉnh tùy theo mục đích sử dụng