


# Prompt Engineering

AI Academy – Lancs Networks

# 1. Tổng hợp thông tin

- Prompt : Tôi đang đọc bài báo về một mô hình xử lý ngôn ngữ tự nhiên tên là ViT5. Bài báo nằm trong file đính kèm. Tổng hợp thông tin từ bài báo này. Cho tôi biết mô hình được dựa trên đâu, kiến trúc là gì, có thể làm được những bài toán nào, có ưu, nhược điểm gì ?
- Kết quả



ViT5: Tổng quan về mô hình xử lý ngôn ngữ tự nhiên cho tiếng Việt

**1. Nền tảng của ViT5**  
ViT5 là một mô hình Transformer được tiền huấn luyện cho tiếng Việt, dựa trên kiến trúc T5 (Text-to-Text Transfer Transformer) do Raffel et al. (2019) phát triển. Mô hình này được huấn luyện theo phương pháp tự giám sát, sử dụng một tập dữ liệu lớn và đa dạng của các văn bản tiếng Việt chất lượng cao [1].

**2. Kiến trúc của ViT5**  
ViT5 áp dụng kiến trúc Transformer, bao gồm các thành phần chính:

- Bộ mã hóa (encoder) và bộ giải mã (decoder) Transformer.
- Độ dài ngữ cảnh: ViT5 có hai phiên bản với độ dài ngữ cảnh khác nhau là 256 và 1024. Kết quả thực nghiệm cho thấy phiên bản 1024 có hiệu suất tốt hơn [1].

**3. Các bài toán mà ViT5 có thể xử lý**  
ViT5 được thiết kế để thực hiện các nhiệm vụ xử lý ngôn ngữ tự nhiên như:

- Tóm tắt văn bản (Abstractive Text Summarization): ViT5 đạt kết quả xuất sắc trên hai bộ dữ liệu Wikilingua và Vietnews, vượt qua nhiều mô hình đã được huấn luyện trước đó [1].
- Nhận dạng thực thể có tên (Named Entity Recognition - NER): ViT5 cạnh tranh với các mô hình hàng đầu hiện nay trên tập dữ liệu PhoNER COVID19 [1].

**4. Ưu điểm của ViT5**

- Hiệu suất cao: ViT5 đạt được kết quả tốt nhất trong các nhiệm vụ tóm tắt văn bản tiếng Việt, vượt trội so với các mô hình đa ngôn ngữ (↓) các mô hình tiền huấn luyện khác [1].

## 2. Coding

- Prompt : Tôi đang muốn xây dựng một mô hình phân loại văn bản. Đóng vai kỹ sư AI có 10 năm kinh nghiệm. Viết cho tôi mô hình đó, trong đó sử dụng mạng nơ ron, số lớp để phân lớp là 4. Viết trên pytorch
- Kết quả



Để xây dựng một mô hình phân loại văn bản sử dụng mạng nơ ron với số lớp phân lớp là 4, chúng ta có thể sử dụng PyTorch, một thư viện mạnh mẽ và linh hoạt cho học sâu. Dưới đây là một ví dụ về cách tạo mô hình phân loại văn bản đơn giản với 4 lớp ẩn bằng PyTorch.

Trước khi bắt đầu, bạn cần cài đặt PyTorch. Bạn có thể cài đặt PyTorch bằng pip nếu chưa cài đặt:

```
bash
```

[Copy code](#)

```
pip install torch torchvision torchaudio
```

Dưới đây là mã mẫu để xây dựng mô hình phân loại văn bản:

```
python
```

[Copy code](#)

```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, Dataset
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer

# Ví dụ dữ liệu giả định
texts = ["This is a positive text", "This is a negative text", "Neutral text example"] * 1
labels = ["positive", "negative", "neutral"] * 100
```

# 3. Refactor code

- Prompt : Tối ưu hóa đoạn code dưới đây, viết lại đoạn code sao cho nó gọn gàng hơn và sử dụng ít vòng for nhất có thể
- Kết quả

```
for index, row in df_process.iterrows():
    bank_list = row['bank_name'].split(',')
    topic_list = row['topic_name'].split(',')
    paragraphs = row['paragraphs']
    added = 0

    for bank in bank_list:
        for topic in topic_list:
            segment = ""
            for paragraph in paragraphs:
                if bank_checker.check_specific_bank(bank, paragraph) and topic_checker.check_specific_topic(topic, paragraph):
                    segment += paragraph + "\n"

            if segment:
                added += 1
                sql.insert_to_db('SAC_THAI', "text_paragraph,chu_de,ngan_hang,id_ban_tin,Ngay", (f"N'{segment}'", f"N'{topic}'"))
            else:
                print(f"Cannot find: {bank} and topic: {topic} in id_ban_tin: {row['id_ban_tin']}")

return 'Completed!'
```



Bạn có thể tối ưu hóa đoạn code trên bằng cách sử dụng ít vòng lặp `for` hơn và tận dụng các phương pháp xử lý dữ liệu hiệu quả của pandas. Dưới đây là cách tối ưu hóa code:

1. Sử dụng `itertuples()` để lặp qua các hàng của dataframe thay vì `iterrows()`, giúp cải thiện hiệu suất.
2. Kết hợp các điều kiện kiểm tra và thêm đoạn văn bản trong cùng một vòng lặp.
3. Sử dụng list comprehension và join để tạo đoạn văn bản thay vì nối chuỗi liên tục.

Dưới đây là phiên bản đã tối ưu hóa của đoạn code:

```
python Copy code

def optimize_code(df_process, bank_checker, topic_checker, sql):
    for row in df_process.itertuples(index=False):
        bank_list = row.bank_name.split(',')
        topic_list = row.topic_name.split(',')
        paragraphs = row.paragraphs

        for bank in bank_list:
            for topic in topic_list:
                segments = [paragraph for paragraph in paragraphs
                            if bank_checker.check_specific_bank(bank, paragraph) and topic_checker.check_specific_topic(topic, paragraph)]

                if segments:
                    segment_text = "\n".join(segments)
                    sql.insert_to_db(
                        'SAC_THAI',
                        "text_paragraph,chu_de,ngan_hang,id_ban_tin,Ngay",
                        (f"N'{segment_text}'", f"N'{topic}'", f"N'{bank}'", f"{row.id_ban_tin}")
                    )
```

## 4. Viết kịch bản Testing

- Prompt : Viết unittest cho đoạn code dưới đây
- Kết quả

```
def fizzbuzz(number):  
    if number % 3 == 0:  
        return "fizz"  
    elif number % 5 == 0:  
        return "buzz"  
    elif number % 15 == 0:  
        return "fizz buzz"  
    else:  
        return number
```



Để viết unittest cho đoạn code `fizzbuzz` bạn có thể làm theo các bước sau đây:

1. Sử dụng module `unittest` của Python.
2. Viết các test case để kiểm tra các điều kiện khác nhau (chia hết cho 3, chia hết cho 5, chia hết cho 15 và không chia hết cho bất kỳ số nào trong số đó).

Dưới đây là đoạn code unittest cho hàm `fizzbuzz`:

```
python Copy code  
  
import unittest  
  
def fizzbuzz(number):  
    if number % 3 == 0:  
        return "fizz"  
    elif number % 5 == 0:  
        return "buzz"  
    elif number % 15 == 0:  
        return "fizz buzz"  
    else:  
        return number  
  
class TestFizzBuzz(unittest.TestCase):  
  
    def test_fizz(self):  
        self.assertEqual(fizzbuzz(3), "fizz")  
        self.assertEqual(fizzbuzz(6), "fizz")  
        self.assertEqual(fizzbuzz(9), "↓")
```

# 5. Debugging

- Prompt : Tôi có đoạn code Java. Hãy xem đoạn code có chạy bị lỗi không. Nếu có lỗi thì sẽ lỗi ở đâu ? Làm thế nào để khắc phục
- Kết quả

```
public class Test {  
  
    private static void printLength(String str) {  
        System.out.println(str.length());  
    }  
  
    public static void main(String []args){  
  
        String myString = null;  
        printLength(myString);  
  
    }  
}
```



Đoạn code Java của bạn sẽ gây ra lỗi khi chạy. Lỗi cụ thể là `NullPointerException`. Nguyên nhân của lỗi này là do bạn đang cố gắng gọi phương thức `length()` trên một đối tượng `String` có giá trị `null`.

Khi `myString` được khởi tạo với giá trị `null` và được truyền vào phương thức `printLength`, việc gọi `str.length()` trong phương thức này sẽ dẫn đến `NullPointerException` vì `str` là `null`.

Để khắc phục lỗi này, bạn cần kiểm tra xem đối tượng `String` có phải là `null` trước khi gọi phương thức `length()`. Dưới đây là cách sửa lỗi:

```
java Copy code  
  
public class Test {  
  
    private static void printLength(String str) {  
        if (str != null) {  
            System.out.println(str.length());  
        } else {  
            System.out.println("The string is null.");  
        }  
    }  
  
    public static void main(String []args){  
        String myString = null;  
        printLength(myString);  
  
    }  
}
```

Với thay đổi này, chương trình sẽ kiểm tra nếu `str` là `null` và thay vì gây ra lỗi, nó sẽ in ra thông báo "The string is null.".



