**1. Phân tích đoạn văn (Context Processing)**

Mô hình **BERT** hoặc các biến thể như **PhoBERT** có khả năng hiểu ngữ nghĩa của đoạn văn bằng cách trích xuất các đặc điểm quan trọng từ đoạn văn. Dưới đây là các bước:

**a. Tokenization**

* Đoạn văn được chia nhỏ thành các **token** (các từ hoặc cụm từ).
* Mỗi token được ánh xạ thành một vector số sử dụng bộ tokenizer tương ứng (ví dụ: bert-base-uncased cho tiếng Anh hoặc vinai/phobert-base cho tiếng Việt).

**Ví dụ**:

* Đoạn văn: *"Môi trường là không gian sống của con người."*
* Tokenized: ['Môi', 'trường', 'là', 'không', 'gian', 'sống', 'của', 'con', 'người', '.']

**b. Hiểu ngữ cảnh (Contextual Embeddings)**

* Các token sau đó được đưa qua các lớp của **BERT** để tạo ra **embedding ngữ cảnh**: Các vector số đại diện cho ý nghĩa của từ trong ngữ cảnh.

**Ví dụ**:

* "Môi trường" có embedding khác khi ở đoạn văn về sinh thái so với đoạn văn về hóa học.

**2. Trích xuất thông tin quan trọng (Key Information Extraction)**

Mục tiêu ở đây là tìm ra các đoạn hoặc từ khóa quan trọng trong đoạn văn, có thể được sử dụng làm câu hỏi hoặc đáp án.

**a. Xác định ý chính**

* Sử dụng mô hình QA (Question Answering) của BERT để trả lời các câu hỏi đơn giản từ đoạn văn.
* Câu trả lời thường là một cụm từ hoặc một câu trong đoạn văn.

**Ví dụ**:

* Đoạn văn: *"Môi trường là không gian sống của con người, bao gồm không khí, đất, nước và các sinh vật."*
* Câu hỏi: *"Môi trường bao gồm những yếu tố nào?"*
* Đáp án: *"Không khí, đất, nước và các sinh vật."*

**b. Sinh câu hỏi tự động**

* Từ các đoạn thông tin chính, sinh câu hỏi phù hợp bằng cách đảo cấu trúc câu hoặc sử dụng các công cụ tạo câu hỏi (Question Generation).

**Ví dụ**:

* Đoạn văn: *"Trái đất quay quanh mặt trời với tốc độ trung bình là 29,8 km/s."*
* Câu hỏi sinh ra: *"Trái đất quay quanh mặt trời với tốc độ bao nhiêu?"*

**3. Sinh phương án trả lời (Answer Generation)**

Sau khi đã có câu hỏi và đáp án đúng:

* **Đáp án đúng**: Được trích xuất từ đoạn văn bằng mô hình QA.
* **Đáp án nhiễu (Distractors)**:
  + Sinh ra từ các từ khóa khác trong đoạn văn.
  + Hoặc sử dụng các thuật toán như từ đồng nghĩa, từ trái nghĩa.

**Ví dụ**:

* Đoạn văn: *"Nước chiếm 71% bề mặt Trái đất."*
* Câu hỏi: *"Nước chiếm bao nhiêu phần trăm bề mặt Trái đất?"*
* Đáp án đúng: *71%*
* Đáp án nhiễu: *51%, 61%, 81%*

**4. Hiệu chỉnh (Post-Processing)**

**a. Đảm bảo tính hợp lý của câu hỏi**

* Kiểm tra ngữ pháp và logic của câu hỏi.
* Loại bỏ các câu hỏi mơ hồ hoặc không rõ ràng.

**b. Đánh giá chất lượng câu hỏi**

* Đánh giá chất lượng câu hỏi dựa trên các tiêu chí:
  + Rõ ràng.
  + Có một đáp án đúng duy nhất.
  + Các đáp án nhiễu không dễ đoán.

**5. Hoạt động tổng thể**

**Luồng xử lý**

1. **Đầu vào**: Một đoạn văn.
2. **Xử lý**:
   * Trích xuất ý chính từ đoạn văn.
   * Sinh câu hỏi phù hợp.
   * Sinh đáp án đúng và các đáp án sai.
3. **Đầu ra**: Một câu hỏi trắc nghiệm hoàn chỉnh với các đáp án.

**Sơ đồ hoạt động**

css

Copy code

[Đoạn văn]

↓

[Tách ý chính] - Mô hình QA

↓

[Sinh câu hỏi] - Mô hình Tạo Câu hỏi

↓

[Sinh đáp án] - Đáp án đúng + Đáp án nhiễu

↓

[Hiệu chỉnh]

↓

[Câu hỏi trắc nghiệm]

**Ứng dụng thực tế**

* **Giáo dục**: Tạo nhanh các đề thi từ tài liệu giảng dạy.
* **Luyện thi**: Sinh nhiều dạng câu hỏi cho học sinh luyện tập.
* **E-learning**: Kết hợp vào các hệ thống học liệu trực tuyến.

**Nguyên lý hoạt động của file**

File Python này tự động sinh câu hỏi trắc nghiệm từ các đoạn văn (context) được cung cấp trong một file CSV, kết hợp với mô hình ngôn ngữ (QA pipeline) để tìm ra đáp án chính xác và tạo ra các đáp án nhiễu.

**1. Tải mô hình và thư viện**

* **Mô hình sử dụng:** bert-large-uncased-whole-word-masking-finetuned-squad
  + Đây là mô hình BERT đã được fine-tune trên bộ dữ liệu SQuAD để thực hiện nhiệm vụ trả lời câu hỏi (Question Answering - QA).
  + Pipeline question-answering giúp sử dụng mô hình này dễ dàng bằng cách cung cấp một đoạn văn (context) và câu hỏi (question) để tìm ra câu trả lời.

python

Copy code

qa\_pipeline = pipeline("question-answering", model="bert-large-uncased-whole-word-masking-finetuned-squad")

* **Thư viện sử dụng:**
  + random: Để chọn đáp án nhiễu ngẫu nhiên.
  + pandas: Để đọc file CSV chứa dữ liệu.

**2. Đọc dữ liệu từ file CSV**

* File CSV (data.csv) chứa hai cột:
  + **context**: Đoạn văn cung cấp thông tin.
  + **questions**: Danh sách các câu hỏi liên quan đến đoạn văn, phân tách bằng dấu chấm phẩy (;).
* Mỗi dòng của file CSV đại diện cho một ngữ cảnh và các câu hỏi tương ứng.

python

Copy code

data = pd.read\_csv("data.csv")

**3. Sinh câu hỏi trắc nghiệm**

**Bước 1: Duyệt qua từng dòng trong file**

* Lấy đoạn văn (context) và danh sách câu hỏi (questions).

python

Copy code

for idx, row in data.iterrows():

context = row['context']

questions = row['questions'].split(";") # Các câu hỏi phân cách bằng dấu chấm phẩy (;)

**Bước 2: Trả lời câu hỏi**

* Với từng câu hỏi trong danh sách:
  + Mô hình QA trả về đáp án chính xác dựa trên ngữ cảnh.

python

Copy code

result = qa\_pipeline(question=question.strip(), context=context)

answer = result['answer']

**Bước 3: Tạo đáp án nhiễu**

* Hàm generate\_distractors được gọi để tạo các đáp án nhiễu:
  + Lấy ngẫu nhiên từ các từ trong ngữ cảnh nhưng không trùng với đáp án chính xác.

python

Copy code

distractors = generate\_distractors(answer, context)

**Bước 4: Xáo trộn đáp án**

* Kết hợp đáp án chính xác và các đáp án nhiễu vào danh sách options.
* Xáo trộn thứ tự đáp án để đảm bảo không cố định.

python

Copy code

options = [answer] + distractors

random.shuffle(options)

**Bước 5: In câu hỏi và đáp án**

* Câu hỏi và các đáp án được in ra với các lựa chọn từ A đến D.

python

Copy code

for i, option in enumerate(options):

print(f"{chr(65 + i)}. {option}")

**4. Hàm sinh đáp án nhiễu**

Hàm generate\_distractors:

* Lấy tất cả các từ trong ngữ cảnh.
* Loại bỏ các từ đã xuất hiện trong đáp án chính xác.
* Chọn ngẫu nhiên một số từ từ danh sách còn lại (đáp án nhiễu).

python

Copy code

def generate\_distractors(answer, context, num\_distractors=3):

words = list(set(context.split()) - set(answer.split()))

distractors = random.sample(words, min(num\_distractors, len(words)))

return distractors

**Nguyên lý tổng quát**

1. **Input:**
   * File CSV chứa đoạn văn và danh sách câu hỏi.
2. **Quy trình:**
   * Với mỗi câu hỏi:
     + Sử dụng mô hình QA để tìm đáp án chính xác.
     + Sinh các đáp án nhiễu từ ngữ cảnh.
     + Xáo trộn các đáp án và định dạng câu hỏi trắc nghiệm.
3. **Output:**
   * Các câu hỏi trắc nghiệm được in ra console, bao gồm đáp án chính xác và các lựa chọn khác.