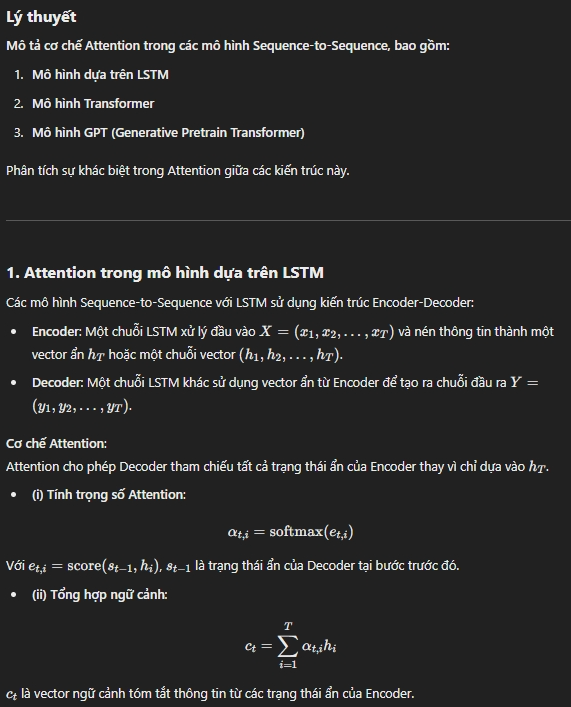
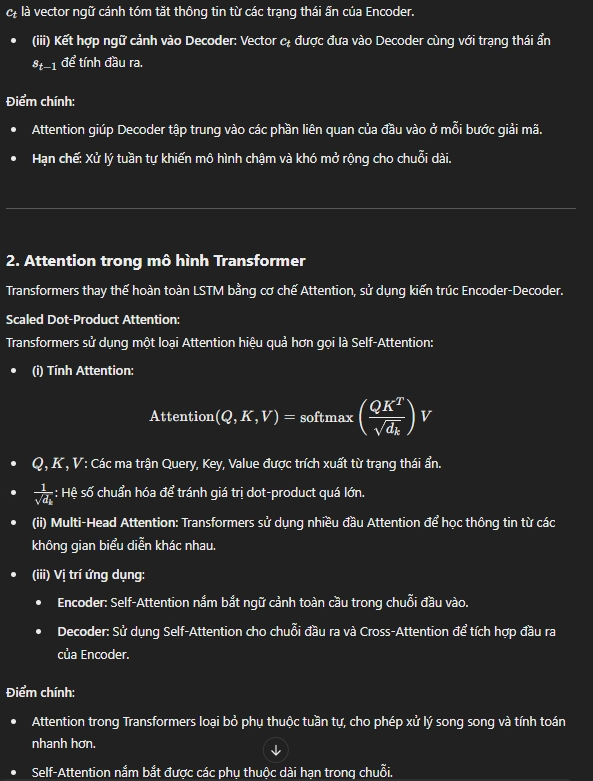
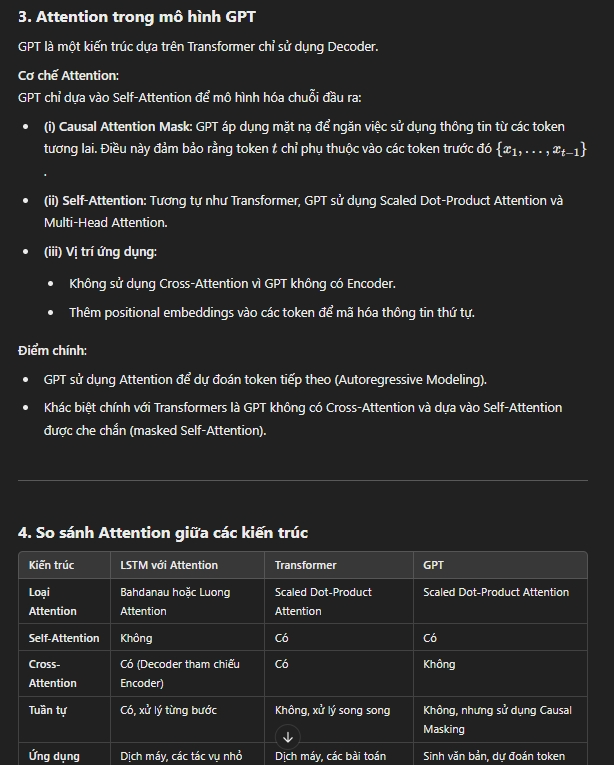
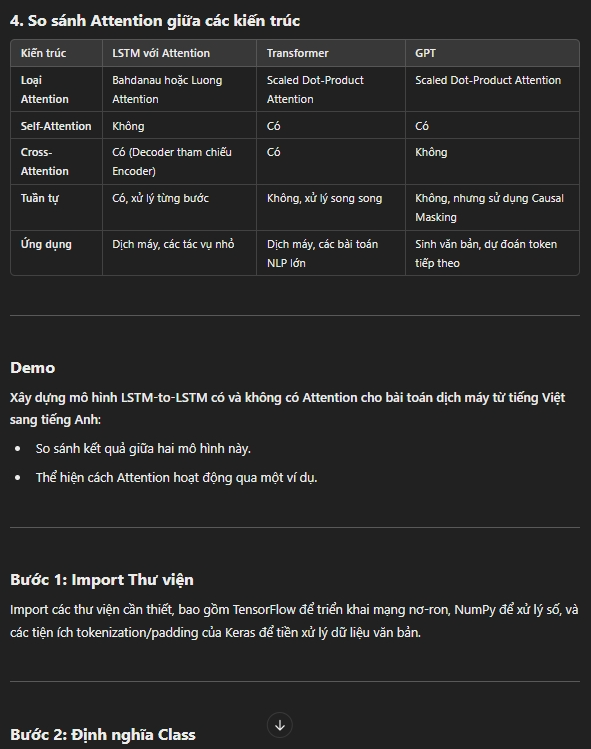
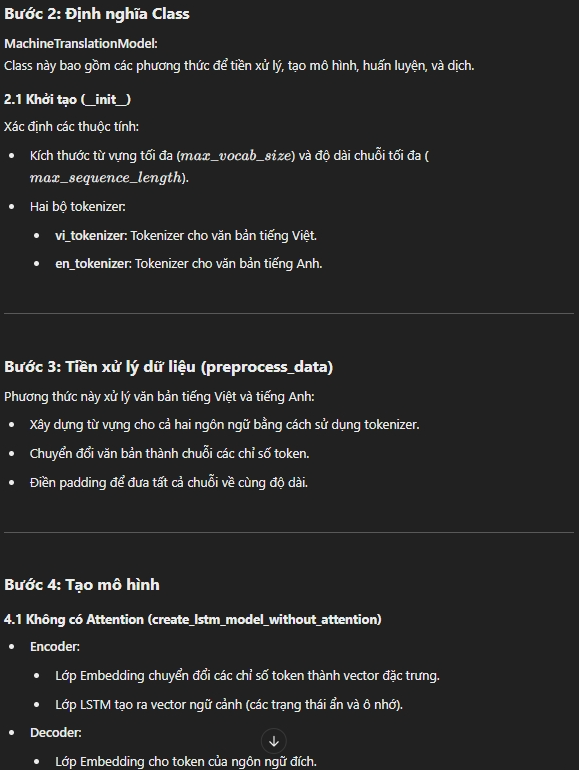
**EXERCISE 1**

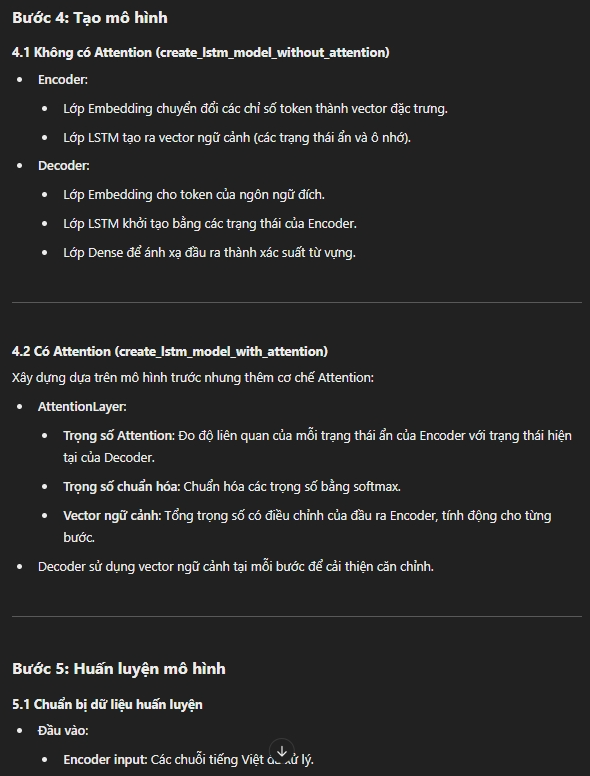
****

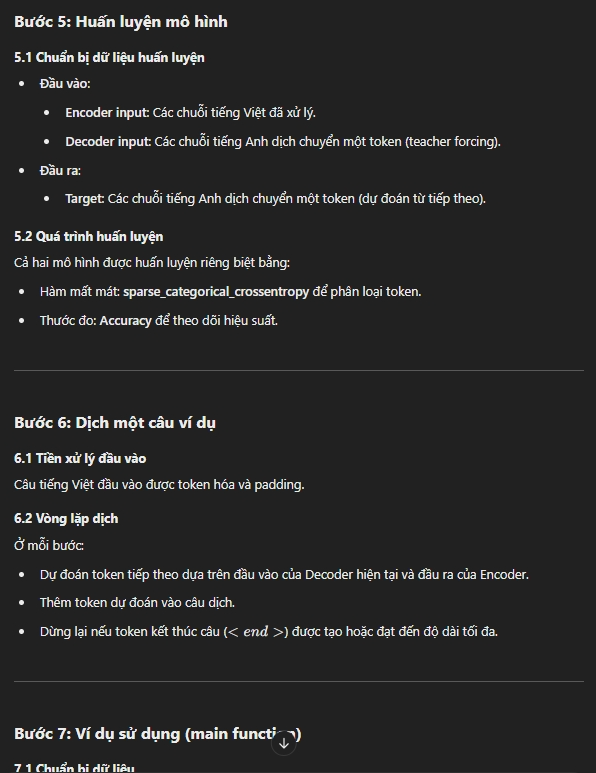
****

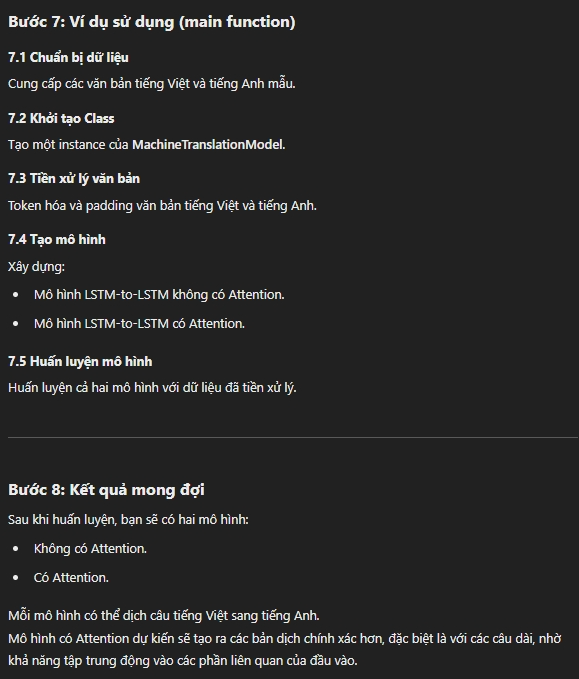
****

****

****

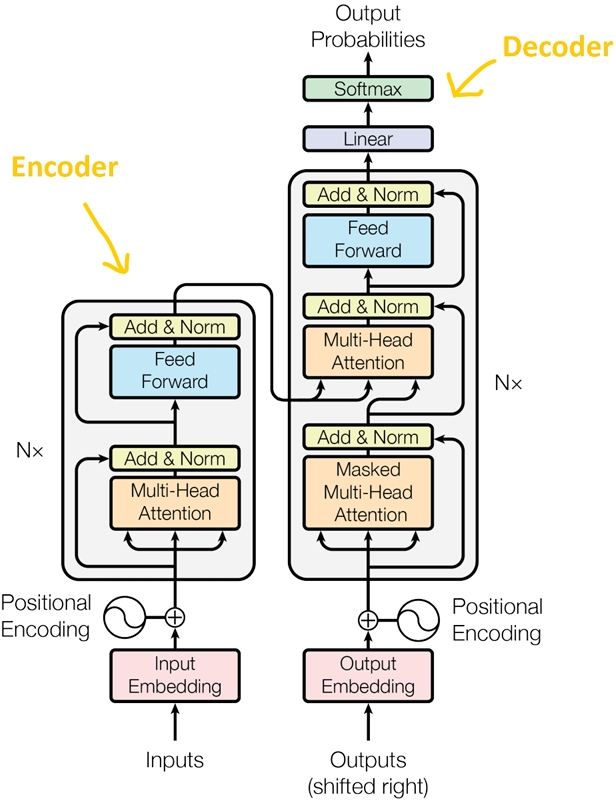
****

****

****

**EXERCISE 2**

1. **Theory**



Hình 1: Kiến trúc Transformer

Giải thích cho hình 1:

- Mã hóa vị trí: Vì Transformer không có các lớp hồi quy hoặc lớp tích chập, nó không tự nhiên biết thứ tự của các token đầu vào. Do đó, cần có một cách để mô hình biết thông tin này, đó là nhiệm vụ của mã hóa vị trí. Sau các lớp embedding, tạo ra các embedding token, chúng ta thêm các vector mã hóa vị trí đại diện cho vị trí của mỗi từ trong câu.

- Lớp chuẩn hóa: Trong kiến trúc của sơ đồ, lớp "Add & Norm" đề cập đến lớp chuẩn hóa. Lớp này đơn giản là chuẩn hóa đầu ra của multi-head attention, cải thiện hiệu quả hội tụ.

- Kết nối dư: Kết nối dư là một khái niệm đơn giản của việc thêm đầu vào của một khối vào đầu ra của nó. Kết nối này cho phép xếp chồng nhiều lớp trong mạng. Trong sơ đồ, kết nối dư được sử dụng sau các khối FFN (Feed-Forward Network) và attention. Trong phần "Add" của "Add & Norm," nó đại diện cho kết nối dư.

- Khối Feed-Forward: Đây là một khối cơ bản, nơi sau khi thực hiện các tính toán trong khối attention ở mỗi lớp, khối tiếp theo là FFN. Bạn có thể hiểu rằng cơ chế attention giúp thu thập thông tin từ các token đầu vào, và FFN xử lý thông tin đó.

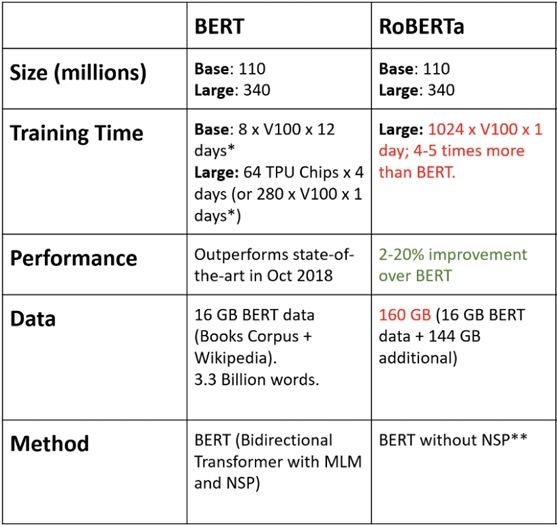
**1.1 Mô hình RoBERTa**

RoBERTa (Robustly Optimized BERT Pretraining Approach) là một biến thể cải tiến của mô hình BERT, được phát triển bởi nhóm nghiên cứu Facebook AI Research (FAIR). Trong khi giữ nguyên kiến trúc Transformer từ BERT, RoBERTa tối ưu hóa quy trình huấn luyện để đạt hiệu suất cao hơn trên các tác vụ xử lý ngôn ngữ tự nhiên (NLP).

1. **Kiến trúc cơ bản**

* **Bộ mã hóa Transformer**:
  + RoBERTa sử dụng kiến trúc Transformer (Vaswani et al., 2017) với các lớp tự chú ý (*self-attention*) và mạng nơ-ron truyền thẳng (*feed-forward neural network*).
  + Hai phiên bản chính:
    - **RoBERTa-base**: 12 lớp, 768 chiều ẩn, 12 đầu chú ý, khoảng 125 triệu tham số.
    - **RoBERTa-large**: 24 lớp, 1024 chiều ẩn, 16 đầu chú ý, khoảng 355 triệu tham số.
  + Tương tự như BERT, RoBERTa được huấn luyện để tạo ra các biểu diễn ngữ cảnh (*contextual embeddings*).

1. **Các cải tiến chính trong RoBERTa**



**Hình 1.1.2: So sánh RoBERTa với BERT**

RoBERTa không thay đổi kiến trúc của BERT nhưng tập trung vào việc tối ưu hóa quy trình huấn luyện với các thay đổi sau:

* **Tăng kích thước tập dữ liệu huấn luyện**:
  + RoBERTa được huấn luyện trên tập dữ liệu lớn hơn (~160GB so với ~16GB của BERT).
  + Nguồn dữ liệu bao gồm: BookCorpus, Wikipedia, Common Crawl News, OpenWebText và Stories.
* **Thay đổi mục tiêu huấn luyện**:
  + Chỉ sử dụng nhiệm vụ *masked language modeling (MLM)* thay vì kết hợp MLM và *next sentence prediction (NSP)* như BERT. Nghiên cứu cho thấy NSP không đóng góp đáng kể vào hiệu quả mô hình.
* **Tăng kích thước batch**:
  + RoBERTa sử dụng kích thước batch lớn hơn (32k tokens so với 16k tokens của BERT), giúp quá trình huấn luyện ổn định hơn.
* **Ngữ cảnh dài hơn**:
  + RoBERTa xử lý ngữ cảnh dài hơn trong mỗi đoạn văn, với độ dài tối đa 512 tokens.
* **Cải thiện mã hóa từ**:
  + Sử dụng *Byte-Pair Encoding (BPE)* với từ điển lớn hơn (~50k từ con) thay vì *WordPiece* như BERT.
* **Thời gian huấn luyện lâu hơn**:
  + RoBERTa trải qua nhiều bước huấn luyện hơn với sự điều chỉnh cẩn thận về tốc độ học (*learning rate*) và giai đoạn khởi động (*warm-up schedules*).

**3. Cải tiến về hiệu suất**

* Nhờ các cải tiến trên, RoBERTa đạt hiệu suất tốt hơn BERT trên nhiều benchmark NLP như GLUE, SQuAD, và RACE.
* Nó được xem như tiêu chuẩn mới trong việc tối ưu hóa các mô hình ngôn ngữ được huấn luyện trước.

**4. Ứng dụng**

* RoBERTa được sử dụng rộng rãi trong các tác vụ NLP như phân loại văn bản, dịch máy, nhận diện thực thể tên (*named entity recognition*), và suy luận ngôn ngữ tự nhiên (*natural language inference*).
* Ngoài ra, nhiều biến thể của RoBERTa (ví dụ như PhoBERT cho tiếng Việt) đã được phát triển để xử lý các ngôn ngữ cụ thể.

**1.2 PhoBERT - Mô hình ngôn ngữ lớn đầu tiên cho tiếng Việt (2020)**

* **Hai phiên bản**: PhoBERTbase và PhoBERTlarge, được xây dựng trên kiến trúc BERT với các cải tiến từ RoBERTa.

**Dữ liệu tiền huấn luyện:**

* Sử dụng tập dữ liệu tiếng Việt lớn (~20GB) gồm Wikipedia (1GB) và tin tức Việt Nam (19GB).
* Dữ liệu được phân tách từ âm tiết sang từ nhờ **RDRSegmenter** (*Rule-based Discriminative Reranking Segmenter*) và được tối ưu hóa bằng **Byte-Pair Encoding (BPE)**.

**Hiệu suất vượt trội:**

* Đạt kết quả tốt nhất (SOTA) trên các tác vụ xử lý ngôn ngữ tự nhiên (NLP), bao gồm:
  + Gán nhãn từ loại (*Part-of-Speech (POS) tagging*).
  + Phân tích phụ thuộc (*Dependency Parsing*).
  + Nhận diện thực thể tên (*Named Entity Recognition (NER)*).
  + Suy luận ngôn ngữ tự nhiên (*Natural Language Inference (NLI)*).
* Vượt trội hơn các mô hình đa ngôn ngữ như **XLM-R**. (*XLM-R (XLM-RoBERTa)* là một mô hình ngôn ngữ đa ngôn ngữ được phát triển bởi Facebook AI, được mở rộng từ RoBERTa và thiết kế để xử lý nhiều ngôn ngữ khác nhau, bao gồm cả những ngôn ngữ ít phổ biến).

**Ứng dụng:**

* Cung cấp công cụ mạnh mẽ cho nghiên cứu và ứng dụng xử lý tiếng Việt.
* Mã nguồn mở trên GitHub.

**Những điểm nổi bật của PhoBERT:**

**Giải quyết hai thách thức lớn**:

1. Tập dữ liệu tiếng Việt trên Wikipedia nhỏ và không đại diện cho ngôn ngữ chung.
2. Khó khăn trong việc phân biệt âm tiết và từ trong tiếng Việt.

**Hiệu quả trên nhiều tác vụ NLP tiếng Việt**:

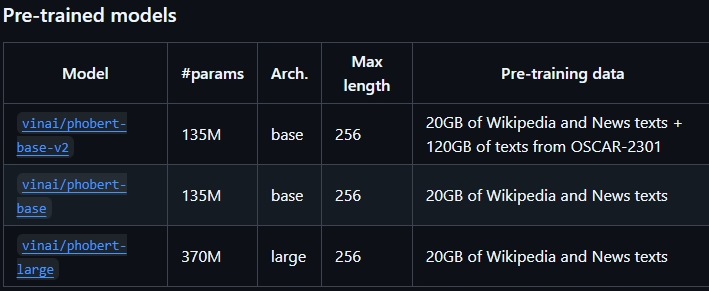
* Cải tiến đáng kể so với các mô hình hiện tại ở tất cả các nhiệm vụ đã thử nghiệm.

**Tối ưu hóa hiệu suất**:

* Sử dụng các kỹ thuật huấn luyện tốt nhất (RoBERTa và BPE).
* Vượt trội hơn các mô hình đa ngôn ngữ dù dùng ít tham số hơn.

**Khả năng phát triển**:

* Là nền tảng vững chắc cho các nghiên cứu và ứng dụng xử lý ngôn ngữ tiếng Việt trong tương lai.



**Hình 1.2: Các phiên bản khác nhau của PhoBERT**

**2. Demo**

a) Nhóm chúng tôi chọn một bài toán trích chọn thông tin mới (Information Extraction): trích xuất thông tin về các khoản đầu tư và startup từ các bài báo kinh tế.

Để tự xây dựng dữ liệu cho tập huấn luyện (train) và tập kiểm tra (test), nhóm chúng tôi đã thực hiện các bước sau:

**Bước 1: Xác định danh sách dữ liệu**

* **Danh sách các startup**, **nhà đầu tư**, và **lĩnh vực**:
  + startups: Danh sách các startup tiềm năng.
  + investors: Danh sách các nhà đầu tư.
  + sectors: Danh sách các lĩnh vực đầu tư.

**Bước 2: Xây dựng hàm sinh dữ liệu**

* **Hàm generate\_investment\_description()**:
  + Sinh ngẫu nhiên một mẫu dữ liệu mô tả các sự kiện đầu tư.
  + Mỗi mẫu sẽ bao gồm các thông tin như:
    - startup (startup nhận đầu tư).
    - investor (nhà đầu tư).
    - sector (lĩnh vực hoạt động của startup).
    - amount (số tiền đầu tư).
  + Tạo một trong những mẫu miêu tả đầu tư từ các template có sẵn.

**Bước 3: Tạo tập huấn luyện và kiểm tra**

* **Sinh dữ liệu huấn luyện** (80% tổng số mẫu dữ liệu):
  + Sử dụng vòng lặp để sinh ra khoảng 80% số mẫu dữ liệu cho tập huấn luyện.
  + Mỗi lần, gọi hàm generate\_investment\_description() để tạo một mẫu ngẫu nhiên và thêm vào tập dữ liệu huấn luyện.
* **Sinh dữ liệu kiểm tra** (20% còn lại):
  + Sử dụng vòng lặp để sinh ra 20% số mẫu dữ liệu cho tập kiểm tra.
  + Tương tự như trên, nhưng chỉ cho vào tập dữ liệu kiểm tra mà không ảnh hưởng đến tập huấn luyện.

**Bước 4: Lưu dữ liệu vào file JSON**

* **Lưu dữ liệu huấn luyện** vào một file riêng biệt startup\_investment\_train.json.
* **Lưu dữ liệu kiểm tra** vào một file riêng biệt startup\_investment\_test.json.
* Sử dụng json.dump() để ghi dữ liệu vào các file tương ứng.

b) Xây dựng mô hình Transformer (có thể sử dụng mô hình Pretrained) để huấn luyện và đánh giá bài toán trên. Lưu ý: đánh giá trên tập Test với các số liệu Precison, Recall, F1-score.

**1. Các Thư Viện và Mô-đun**

* **Thư viện chính**:
  + transformers: Sử dụng PhoBERT từ thư viện Hugging Face để huấn luyện và đánh giá mô hình.
  + torch: Xử lý tensor và tạo dataset tùy chỉnh.
  + scikit-learn: Tính toán các độ đo đánh giá như Precision, Recall, F1-score.
  + json: Tải dữ liệu từ file JSON.
  + numpy: Xử lý các mảng số học.

**2. Khởi tạo Dataset Tùy chỉnh**

Lớp StartupInvestmentDataset được xây dựng để chuẩn bị dữ liệu theo định dạng phù hợp cho mô hình.

**2.1. Hàm \_\_init\_\_**

* **Dữ liệu đầu vào**:
  + data: Tập dữ liệu (chuỗi văn bản và nhãn thực thể).
  + tokenizer: Bộ mã hóa từ (PhoBERT tokenizer).
  + max\_len: Độ dài tối đa của mỗi chuỗi (128 token).
  + label\_encoder: Chuyển nhãn thực thể sang số nguyên.
* **Các nhãn**:
  + Bao gồm: O (Outside), B-... (Begin), I-... (Inside) cho từng loại thực thể (STARTUP, INVESTOR, SECTOR, AMOUNT).

**2.2. Hàm \_\_getitem\_\_**

Xử lý dữ liệu từng mẫu:

1. **Token hóa chuỗi văn bản**: Biến chuỗi thành ID token và đệm (padding) nếu ngắn hơn max\_len.
2. **Khởi tạo nhãn**:
   * Mặc định tất cả các token là O (không thuộc thực thể nào).
3. **Gán nhãn BIO**:
   * Với mỗi thực thể trong entities, xác định vị trí thực thể trong chuỗi.
   * Chuyển thành token tương ứng và gán nhãn bắt đầu (B-) và bên trong (I-) của thực thể.

**2.3. Hàm \_\_len\_\_**

* Trả về số lượng mẫu trong tập dữ liệu.

**3. Hàm Tính Độ Đo Hiệu Suất (compute\_metrics)**

* **Đầu vào**: Kết quả dự đoán và nhãn thực tế.
* **Xử lý**:
  + Lấy nhãn thực tế và nhãn dự đoán, bỏ qua các token padding (-100).
  + Tính toán độ đo **Precision**, **Recall**, và **F1-score** theo từng nhãn.
* **Đầu ra**: Trả về kết quả đánh giá.

**4. Tải Dữ Liệu Từ JSON**

Hàm load\_dataset đọc dữ liệu từ file JSON:

* **Định dạng dữ liệu**:
  + Dữ liệu huấn luyện (train) và kiểm tra (test), mỗi phần chứa:
    - text: Chuỗi văn bản.
    - entities: Các thực thể (startup, investor, sector, amount).

**5. Hàm main - Quy Trình Chính**

**5.1. Cấu Hình**

* **Mô hình PhoBERT**: Sử dụng phiên bản vinai/phobert-base.
* Các thông số huấn luyện:
  + Độ dài tối đa (max\_len): 128.
  + Kích thước batch (batch\_size): 16.
  + Số epoch: 5.
  + Learning rate: 2×10^(-5).

**5.2. Tạo Dataset**

* Gọi lớp StartupInvestmentDataset cho tập train và test.

**5.3. Cấu Hình Huấn Luyện**

* **TrainingArguments**:
  + Số epoch, batch size, learning rate, và chiến lược lưu mô hình tốt nhất.
  + Chiến lược đánh giá sau mỗi epoch (evaluation\_strategy='epoch').
* **Trainer**:
  + Định nghĩa mô hình PhoBERT, dữ liệu, và hàm tính toán độ đo.

**5.4. Huấn Luyện và Đánh Giá**

1. **Huấn luyện mô hình**:
   * trainer.train(): Huấn luyện mô hình trên tập dữ liệu huấn luyện.
2. **Đánh giá mô hình**:
   * trainer.evaluate(): Đánh giá mô hình trên tập kiểm tra (test).
   * In kết quả Precision, Recall, và F1-score.

**6. Tóm Tắt Quy Trình**

1. Tải dữ liệu JSON và chuẩn bị tập dữ liệu.
2. Token hóa văn bản và ánh xạ nhãn BIO.
3. Huấn luyện mô hình PhoBERT trên tập huấn luyện.
4. Đánh giá trên tập kiểm tra với các độ đo hiệu suất.