# DANH MỤC HÌNH VẼ

[Hình 1.1 Lịch sử của mô hình ngôn ngữ 1](#_Toc166935179)

[Hình 1.2 Mô hình Transformer 2](#_Toc166935180)

[Hình 1.3 Công thức tính position embedding chẵn 3](#_Toc166935181)

[Hình 1.4 Công thức tính position embedding lẻ 3](#_Toc166935182)

[Hình 1.5 Dữ liệu đầu vào của encoder 3](#_Toc166935183)

[Hình 1.6 Tính vector Query, Key, Value 4](#_Toc166935184)

[Hình 1.7 Công thức Attention 4](#_Toc166935185)

[Hình 1.8 Multi-head attention 5](#_Toc166935186)

[Hình 1.9 Kết quả Z 5](#_Toc166935187)

[Hình 1.10 Cấu trúc BERT 6](#_Toc166935188)

[Hình 1.11 BERT input 7](#_Toc166935189)

[Hình 1.12 Bài toán điền từ trống 7](#_Toc166935190)

[Hình 1.13 Bài toán dự đoán hai câu liên tiếp 8](#_Toc166935191)

[Hình 2.1 Decoder của transformer. 10](#_Toc166935192)

[Hình 2.2 Quá trình huấn luyện GPT 12](#_Toc166935193)

[Hình 2.3 Vấn đề cần dữ liệu 12](#_Toc166935194)

[Hình 2.4 So sánh hiệu suất GPT2 15](#_Toc166935195)

[Hình 2.5 So sánh hiệu suất GPT3 16](#_Toc166935196)

[Hình 3.1 Kiến trúc Seq2Seq 17](#_Toc166935206)

# DANH MỤC BẢNG BIỂU

# DANH MỤC CÁC CHỮ VIẾT TẮT

|  |  |
| --- | --- |
| BERT | Bidirectional Encoder Representation from Transformers |
| SEP | Separator |
| Seq2Seq | Sequence to Sequence |
|  |  |
|  |  |
|  |  |

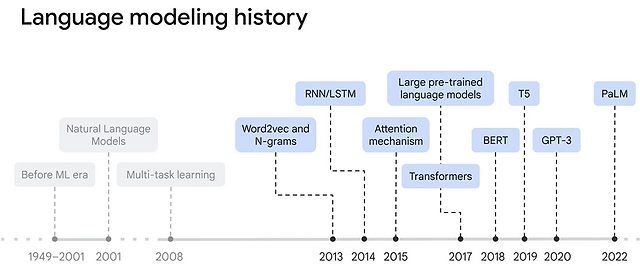
# MÔ HÌNH ENCODER DỰA TRÊN TRANSFORMER

## TRANSFORMER

### Quá trình phát triển

Mô hình ngôn ngữ đã có nhiều bước đột phá lớn qua kể từ lúc bắt đầu phát triển:

* Năm 2013, Word2vec và N-gram được đề xuất giúp lưu trữ các từ dưới dạng vector và tìm hiểu các mối quan hệ ngữ cảnh giữa các từ trong văn bản.
* Năm 2014, RNN và LSTM là hai mô hình mạng neural đặc biệt giúp tăng khả năng xử lý của các vấn đề.
* Năm 2015, cơ chế Attention được giới thiệu, dẫn đến rất nhiều mô hình được xây dựng dựa trên nó. Trong đó Transformers là một trong các mô hình huấn luyện sẵn gặt hái được nhiều thành công với khả năng ứng dụng thực tế rộng rãi.



Hình 1. Lịch sử của mô hình ngôn ngữ

## Transformer

### Cấu trúc

Mô hình Transformer được giới hiệu vào năm 2017 bởi nhà khoa học Ashish Vaswani từ Google (Attention Is All You Need, 2017) với cấu trúc gồm encoder và decoder sử dụng cơ chế Attention để hỗ trợ nâng cao hiệu suất .

A screenshot of a computer

Description automatically generated

Hình 1. Mô hình Transformer

Encoder bao gồm các lớp self-attention và feed forward thực hiện việc biến đổi dữ liệu đầu vào và chuyển các dữ liệu đó đến Decoder.

Decoder bao gồm các lớp self-attention, encoder-decoder attention và feed forward sử dụng dữ liệu nhận vào từ Encoder để xứ lý từng yêu cầu cụ thể.

Transformer mang lại nhiều cái tiến hơn so với các mô hình ngôn ngữ đi trước:

* Vector đại diện cho từ sẽ bao gồm cả ngữ cảnh và khả năng thay đổi ngữ nghĩa của từ trong câu. VD: bank trong river bank và bank robber sẽ được biểu diễn bằng hai vector khác nhau, các mô hình trước Transformer sẽ có thể biểu diễn hai chữ bank này giống nhau.
* Tính song song giúp hoạt động tốt trên GPU và TPU.
* Khả năng xử lý tốt hơn các mô hình trước.
* Xử lý tất cả các token từ cùng lúc.

Do chủ đề nghiên cứu của chương này tập trung vào mô hình Transformer Based Encoder, nên sẽ tập trung vào phần Encoder của Transformer

### Xử lý dữ liệu đầu vào

Input embedding nhận vào các câu theo ngôn ngữ tự nhiên và tạo embedding vector cho từng từ trong câu. Đối với mô hình Transformer mỗi embedding vector của từng từ sẽ đều có kích thước 512 biểu diễn từ đó trong mô hình.

Position Encoding sẽ là embedding vector vị trí của từng từ trong văn bảng. Công thức tính giá trị position Encoding sẽ như sau với dmodel là kích thước vector:

A math equation with a circle and numbers

Description automatically generated with medium confidence

Hình 1. Công thức tính position embedding chẵn

A math equation with numbers and symbols

Description automatically generated with medium confidence

Hình 1. Công thức tính position embedding lẻ

Sau khi đã có input embedding và position embedding ta có thể cộng vector lại đối với từng từ trong văn bản để tạo ra vector đầu vào cho Encoder

A screenshot of a computer

Description automatically generated

Hình 1. Dữ liệu đầu vào của encoder

### Lớp Self Attention

Phân tích ra các vector Query, Key và Value của từng dựa vào ma trận weight tương ứng đã có của mô hình.

A diagram of a algorithm

Description automatically generated with medium confidence

Hình 1. Tính vector Query, Key, Value

Kết hợp các vector Query, Key, Value của tất cả từ thành ma trận Query, Key, Value. Tính ma trận Attention bằng công thức đây:

A math equation with black text

Description automatically generated

Hình 1. Công thức Attention

Kết hợp các ma trận attention của từng bộ giá trị weight thì ta sẽ có multi-head attention. Sau đó nhân với weight 0 để ra ma trận kết quả.

A screenshot of a computer

Description automatically generated

Hình 1. Multi-head attention

A graph of a graph of a number of squares

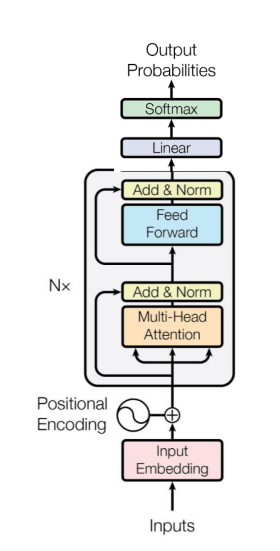
Description automatically generated with medium confidence

Hình 1. Kết quả Z

## BERT

### Tổng quan

Mô hình Bidirectional Encoder Representations from Transformers (BERT) là một mộ hình Encoder dựa trên Transformer. BERT sẽ chỉ xử lý dữ liệu qua các tầng Encoder và trả về kết quả mà không dùng đến Decoder.



Hình 1. Cấu trúc BERT

BERT được phát triển bởi Google vào năm 2018 và vẫn được sử dụng rộng rãi trong nhiều lĩnh vực cho đến nay. Công nghệ Google Search mà chúng ta dùng hiện nay cũng sử dụng BERT.

Có hai biến thể chính là BERT Base (12 lần Encoder, 12 đầu attention) và BERT LARGE (24 lần Encoder, 16 đầu attention). Mỗi biến thể đều được huấn luyện sẵn bằng các bộ từ khổng lồ như Wikipedia và Books Corpus.

Khác với các mô hình ngôn ngữ khác, BERT được huấn luyện dựa hai bài toán điền từ trống trong câu và dự đoán hai câu có phải liên tiếp không. BERT sử dụng cả ngữ cảnh bên trái và bên phải của từ (bidirectional). BERT có thể được tùy chỉnh thêm để xử lý các bài toán khác.

### Xử lý dữ liệu đầu vào

Ngoài việc cần có vector Input Embedding và Position Embedding, BERT còn yêu cầu đầu vào phải có Segment Embedding tượng trưng cho từ đó thuộc câu nào. Ngoài ra chúng ta cần phải chú ý đến token mới [SEP] đây xem như là token để kết thúc câu.

A screenshot of a computer

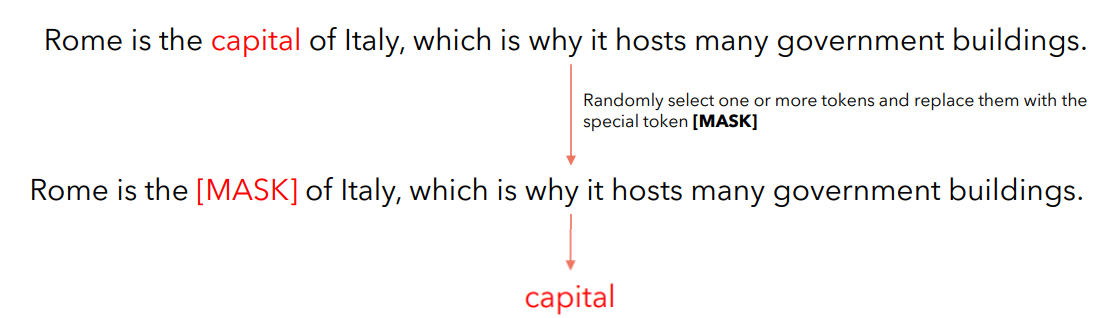
Description automatically generated

Hình 1. BERT input

### Cách huấn luyện sẵn của BERT

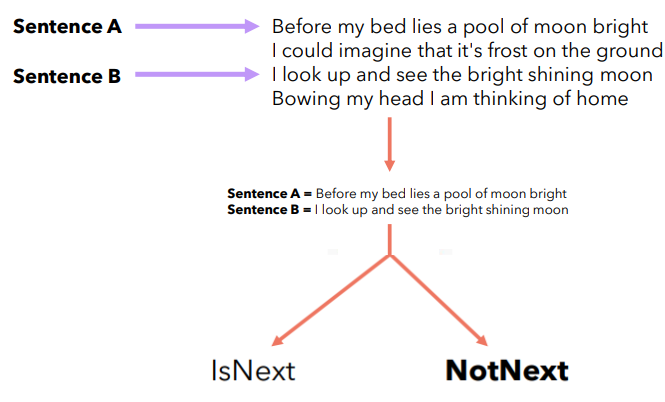
BERT được Google huấn luyện sẵn qua hai bài toán điền từ trống trong câu và dự đoán hai câu liên tiếp.

Về bài toán điền từ trống, Google đã ẩn đi 15% số từ trong bộ dữ liệu và huấn luyện BERT để điền các từ trống này dựa vào ngữ cảnh của câu. Google chọn số dữ liệu ẩn là 15% vì khi ẩn ít hơn sẽ tốn quá nhiều tài nguyên để huấn luyện và khi nhiều hơn sẽ không đủ ngữ cảnh cho câu.



Hình 1. Bài toán điền từ trống

Về bài toán dự đoán hai câu liên tiếp, Google sẽ truyền vào hai câu liên tiếp phân biệt bằng token [SEP], mô hình sẽ dự đoán xem hai câu này có liên tiếp nhau hay không.



Hình 1. Bài toán dự đoán hai câu liên tiếp

### Ứng dụng của BERT

Mô hình và tokenizer của BERT có thể được sử dụng qua các thư viên NLP nổi tiếng như Transformers của Hugging Face. Người dùng có thể sử dụng các thư viện này và tùy chỉnh BERT để thực hiện các bài toán khác như classification, question answering, tagging tokens.

Để thực hiện ta sẽ phải tự tạo một tầng output riêng biệt để kết hợp với mô hình BERT có sẵn. Việc này sẽ giúp tiết kiệm thời gian và tài nguyên vì mô hình BERT đã được huấn luyện sẵn với nhiều hiểu biết về ngôn ngữ.

# MÔ HÌNH DECODER DỰA TRÊN TRANSFORMER

## GPT models

**Định nghĩa:** GPT là viết tắt của "Generative Pre-trained Transformer", là một loạt các mô hình trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP) phát triển bởi OpenAI.

**Mục tiêu:** Mục tiêu chính của GPT là sinh ra văn bản tự nhiên, tức là tạo ra các đoạn văn bản mới dựa trên ngữ cảnh đầu vào.

**Kiến trúc:**

* GPT sử dụng kiến trúc Transformer, nhưng chỉ sử dụng phần decoder của nó.
* Mỗi lớp decoder trong GPT bao gồm các thành phần như self-attention mechanism và feedforward neural network.

**A diagram of a process flow

Description automatically generated**

Hình 2. Decoder của transformer.

* Mô hình GPT thường bao gồm nhiều lớp decoder được xếp chồng lên nhau để tạo ra một mô hình mạnh mẽ và linh hoạt.

**Ứng dụng:** GPT có thể được sử dụng trong nhiều ứng dụng NLP khác nhau, bao gồm tổng hợp văn bản, tạo ra câu chuyện, hoặc trả lời các câu hỏi.

## Xử lí dữ liệu đầu vào cho mô hình GPT

**Tokenization:** Đầu tiên, văn bản đầu vào được chia thành các phần nhỏ hơn gọi là "token". Token có thể là từ hoặc ký tự, tùy thuộc vào cách tokenization được thực hiện.

**Chuyển đổi thành số:** Mỗi token được ánh xạ sang một số nguyên duy nhất, được gọi là "ID". Các từ và ký tự trong văn bản được biểu diễn dưới dạng các số.

**Thêm token đặc biệt:** Token đặc biệt thường được thêm vào đầu và cuối mỗi chuỗi văn bản để đánh dấu vị trí bắt đầu và kết thúc.

**Padding:** Để đảm bảo rằng tất cả các chuỗi đều có độ dài như nhau, các chuỗi được thêm các token padding. Điều này cần thiết vì mô hình GPT yêu cầu các đầu vào có cùng kích thước.

**Tạo mini-batch:** Dữ liệu được chia thành các mini-batch, mỗi mini-batch chứa một số lượng chuỗi văn bản có độ dài giống nhau.

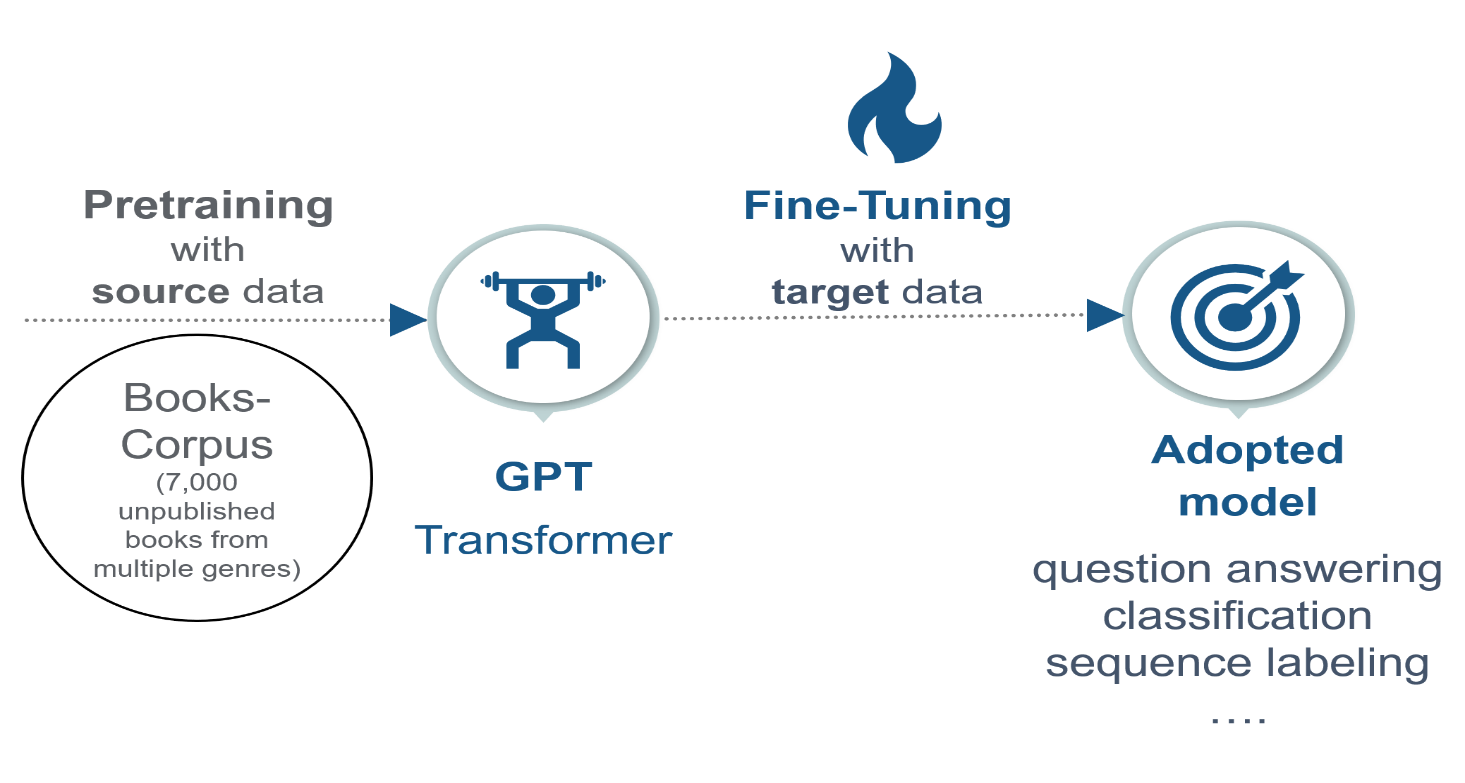
**Chuyển đổi thành tensors:** Cuối cùng, dữ liệu được chuyển đổi thành tensors (mảng đa chiều) để có thể được đưa vào mô hình GPT để huấn luyện hoặc dự đoán.

## Transfer Learning

Là cải thiện khả năng học một cái gì đó mới thông qua việc vận dụng các kiến thức đã được tiếp thu và có liên quan từ những điều đã học trước đó.

Ví dụ như là khi bạn học đi xe đạp thì bạn phải học từ con số 0 đó là học cách cân bằng cách phanh xe, vân vân. Nhưng khi bạn học đi xe máy khi đã biết đi xe đạp thì không cần học lại các thứ đó nữa mà vận dụng nó qua việc học đi xe máy.

Trong mô hình GPT, transfer learning được áp dụng thông qua việc sử dụng Pre-training trên một lượng dữ liệu văn bản. Trong quá trình đó, mô hình học được cách hiểu và tạo ra văn bản tự nhiên. Sau đó Fine-tuning(tinh chỉnh) mô hình này trên các nhiệm vụ cụ thể, chẳng hạn như phân loại văn bản. Trong quá trình này, chúng ta chỉ cần cung cấp một lượng nhỏ dữ liệu gắn nhãn cho nhiệm vụ cụ thể, và mô hình sẽ sử dụng kiến thức đã học từ quá trình pre-training để cải thiện hiệu suất cho nhiệm vụ mới này.



Hình 2. Quá trình huấn luyện GPT

## Một số vấn đề của việc Fine-tuning khi kết hợp GPT models

Vẫn cần quá nhiều dữ liệu:

A black background with yellow text

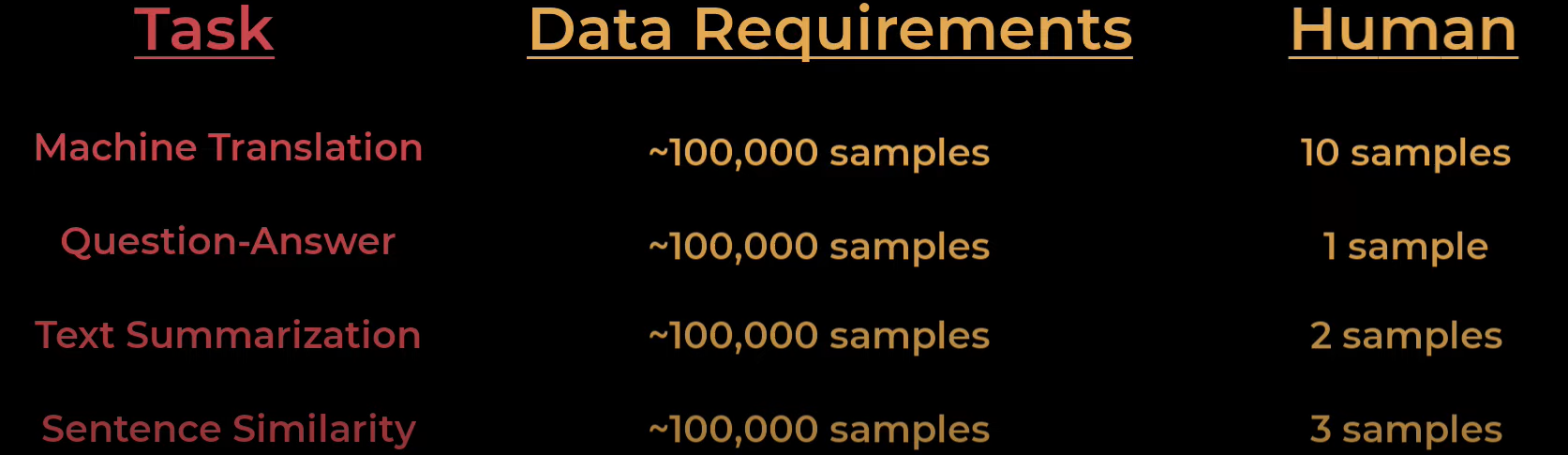
Description automatically generated

Hình 2. Vấn đề cần dữ liệu

Quá dễ bị overfitting:

* **Dữ liệu Huấn luyện có Thay đổi:** Trong quá trình transfer learning, mô hình được huấn luyện trên một tập hợp lớn dữ liệu không gắn nhãn, sau đó fine-tuned trên một tập dữ liệu nhỏ hơn cho một tác vụ cụ thể. Nếu dữ liệu fine-tuning không đại diện cho phân phối của dữ liệu trong quá trình pre-training, mô hình có thể bị overfitting.
* **Số lượng Tham số Lớn:** Các mô hình GPT thường có hàng triệu hoặc thậm chí hàng tỷ tham số. Khi mô hình có quá nhiều tham số so với lượng dữ liệu huấn luyện, có nguy cơ mô hình sẽ học được các mẫu ngẫu nhiên trong dữ liệu huấn luyện thay vì các mẫu tổng quát.
* **Overfitting trên Dữ liệu nhỏ:** Trong transfer learning, fine-tuning mô hình trên một tập dữ liệu nhỏ có thể dễ dàng dẫn đến overfitting. Nếu dữ liệu huấn luyện không đủ lớn hoặc đa dạng, mô hình có thể học những chi tiết không quan trọng hoặc nhiễu trong dữ liệu.
* **Số lượng Epochs:** Việc huấn luyện mô hình quá nhiều epochs có thể dẫn đến overfitting, đặc biệt là khi dữ liệu huấn luyện không đủ lớn. Mô hình có thể học được các mẫu không cần thiết trong dữ liệu, dẫn đến hiện tượng overfitting.

Không giống như cách con người học:



**Không ổn khi xử lí nhiều kiểu ngôn ngữ một lúc:** Vì sự khác biệt về cấu trúc ngôn ngữ, văn hóa và ngữ cảnh giữa các ngôn ngữ khác nhau. Điều này có thể dẫn đến kết quả không ổn định hoặc không chính xác khi áp dụng mô hình cho các ngôn ngữ mới.

A screen shot of a graph

Description automatically generated

## Meta learning

Meta learning, hay còn được gọi là "learning to learn", là một lĩnh vực trong machine learning tập trung vào việc phát triển các thuật toán hoặc mô hình có khả năng học cách học từ dữ liệu và kinh nghiệm trước đó. Ý tưởng cơ bản của meta learning là học cách thích ứng nhanh chóng và hiệu quả với các tác vụ mới mà không cần phải huấn luyện lại từ đầu.

Trong meta learning, mô hình học được cách tự động tìm hiểu cách thực hiện các phương pháp học tối ưu và chiến lược học từ dữ liệu và kinh nghiệm trước đó. Nó có thể bao gồm các khía cạnh như:

* Học cách chọn ra các đặc trưng quan trọng từ dữ liệu.
* Học cách tối ưu hóa quá trình học.
* Học cách chọn ra một mô hình phù hợp cho từng tác vụ cụ thể.
* Học cách thích ứng với các môi trường học mới.

Với meta learning, mô hình không chỉ học được từ dữ liệu huấn luyện của một tác vụ cụ thể, mà còn học được cách sử dụng kiến thức và kinh nghiệm đã học được để nhanh chóng thích ứng với các tác vụ mới. Điều này giúp tăng cường khả năng linh hoạt và hiệu suất của mô hình trong việc giải quyết các tác vụ mới mà nó chưa từng gặp phải trước đó.

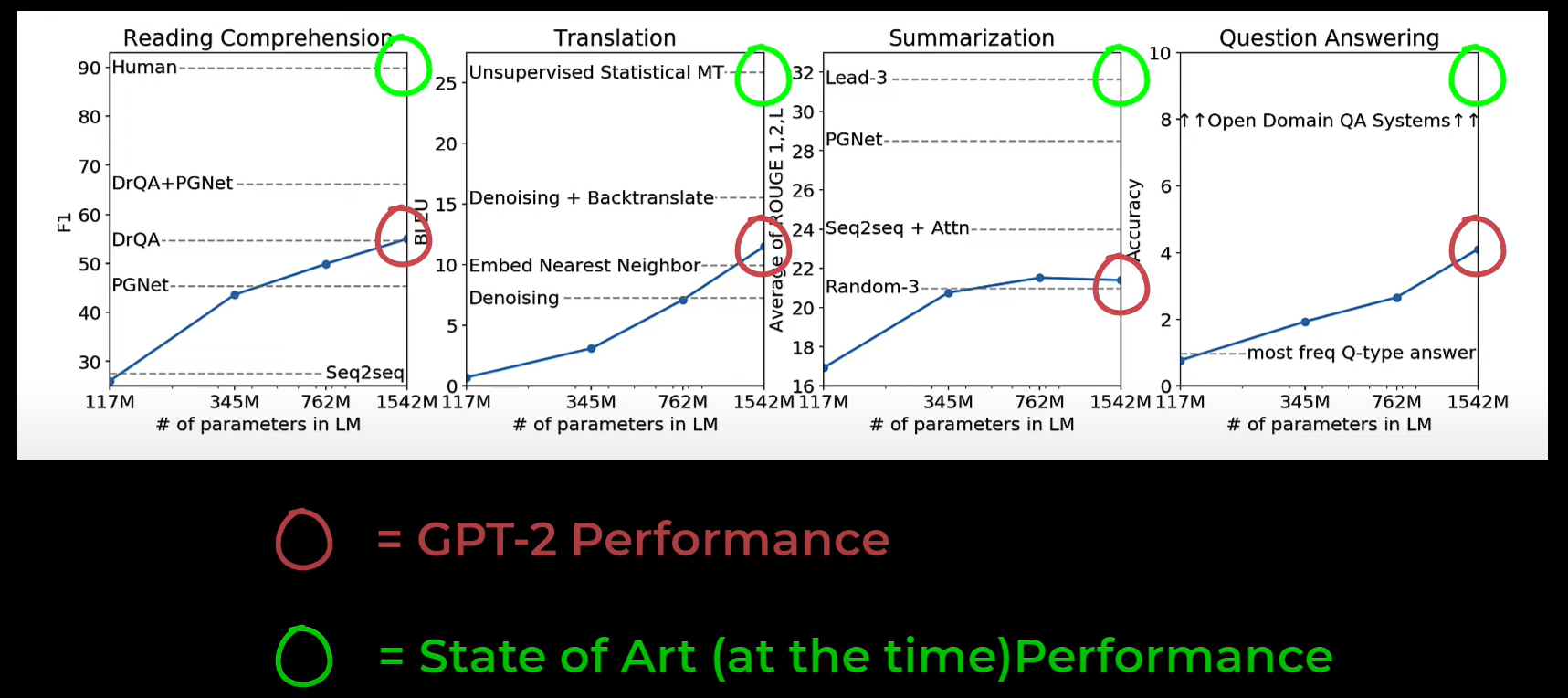
## Meta – learning và GPT

Meta – learning là một giải pháp để giải quyết các vấn đề mà GPT-1 gặp phải. Và sự kết hợp đó tạo nên GPT-2.

GPT-2 khá là giống với phải GPT-1, có cùng pre-traning with language modeling nhưng thay vì fine-tuning chúng ta sẽ sử dụng zero-short learning(học không giám sát). Khi bạn sử dụng zero-shot learning với GPT-2, bạn không cần huấn luyện mô hình trước đó cho một nhiệm vụ cụ thể. Thay vào đó, bạn cung cấp cho mô hình một mô tả ngắn gọn hoặc một loạt các từ khóa liên quan đến nhiệm vụ bạn muốn mô hình thực hiện. Dựa trên thông tin này, mô hình sẽ cố gắng tạo ra văn bản hoặc thực hiện nhiệm vụ đó một cách tự động.

Tuy vậy thì zero-short learning cũng rất khó đối với các mô hình vì vậy cần phải mở rộng kiến trúc để nắm bắt nhiều mẫu ngôn ngữ nhất có thể trong quá trình đào tạo trước. Và đó là lí do GPT được huấn luyện với 1,5 tỉ tham số khác nhau.

Mặc dù cách tiếp cận này không hoạt động tốt trong việc tinh chỉnh một số điểm chuẩn, tuy nhiên việc mở rộng kiến trúc thực sự vẫn giúp ích cho hiệu suất theo một cách nào đó



Hình 2. So sánh hiệu suất GPT2

## GPT-3

Nếu chúng ta vẫn giữ cách cũ và tiếp tục tăng kích tăng kích thước của mô hình thì đó sẽ là sự hình thành của phiên bản GPT-3

GPT-3 là mô hình ngôn ngữ khổng lồ với 175 tỉ tham số. Tuy vẫn giữ các điểm của các bản tiền nhiệm như là pre-trained với các mô hình ngôn ngữ và sau đó fine-tuned bằng meta learning.

Nhưng thay vì chỉ sử dụng zero-short learning như ở GPT-2 thì GPT-3 sử dụng One-shot learning, Zero-short learning và Few-short learning

**One-shot learning (học một mẫu):** Trong GPT-3, One-shot learning là khi một mô hình chỉ cần một ví dụ duy nhất để học một nhiệm vụ cụ thể. Ví dụ: Bạn chỉ cần cho mô hình một ví dụ của một loại hoa để nó nhận biết các loại hoa khác trong tương lai.

**Few-shot learning (học một số mẫu):** Trong GPT-3, Few-shot learning là khi mô hình được cung cấp một số lượng nhỏ các ví dụ để học một nhiệm vụ cụ thể.

Ví dụ: Bạn cung cấp một vài ví dụ của các loại hoa để mô hình học nhận biết chúng.

**Zero-shot learning (Học không mẫu):** Zero-shot learning là khi mô hình cố gắng thực hiện một nhiệm vụ mà nó chưa được huấn luyện trước đó hoặc không có dữ liệu huấn luyện cụ thể cho nhiệm vụ đó. Ví dụ: Nếu bạn muốn mô hình phân loại hình ảnh của các loài chim mà không có dữ liệu huấn luyện cụ thể cho từng loài chim, bạn có thể cung cấp cho mô hình một mô tả về các đặc điểm của từng loài chim và mô hình sẽ cố gắng dự đoán chúng từ mô tả.

A graph with colored lines and numbers

Description automatically generated

Hình 2. So sánh hiệu suất GPT3

Mang lại hiệu năng vượt qua cả mong đợi ở một số yêu cầu nhất định

# MÔ HÌNH ENCODER VÀ DECODER

## Mô hình Seq2Seq (Sequence-to-Sequence)

Seq2Seq (Sequence-to-Sequence) là một kiến trúc mạng nơ-ron nhân tạo được sử dụng rộng rãi trong các tác vụ xử lý ngôn ngữ tự nhiên (NLP) và các bài toán khác liên quan đến việc biến đổi chuỗi dữ liệu đầu vào thành chuỗi dữ liệu đầu ra. Kiến trúc này đặc biệt hiệu quả trong các ứng dụng như dịch máy, tóm tắt văn bản, hỏi đáp tự động, và nhận dạng giọng nói.

A diagram of a computer code

Description automatically generated

Hình 3. Kiến trúc Seq2Seq

## Các thành phần

Mô hình Seq2Seq (Sequence-to-Sequence) gồm ba thành phần chính: Encoder (bộ mã hóa), Decoder (bộ giải mã), và cơ chế Attention (cơ chế chú ý). Mỗi thành phần có vai trò và cấu trúc riêng, góp phần vào việc biến đổi chuỗi dữ liệu đầu vào thành chuỗi dữ liệu đầu ra một cách hiệu quả.

**Encoder (Bộ Mã Hóa):**

* *Nhiệm Vụ*: Mã hóa chuỗi đầu vào thành một vector có kích thước cố định (context vector).
* *Kiến Trúc*: Thường sử dụng các lớp LSTM (Long Short-Term Memory) hoặc GRU (Gated Recurrent Unit) để xử lý các chuỗi đầu vào.
* *Hoạt Động*: Nhận vào một chuỗi (ví dụ, câu có lỗi chính tả) và tạo ra một chuỗi các trạng thái ẩn. Trạng thái ẩn cuối cùng được sử dụng làm context vector.

**Decoder (Bộ Giải Mã):**

* *Nhiệm Vụ:* Giải mã context vector thành chuỗi đầu ra (ví dụ, câu chính xác không có lỗi chính tả).
* *Kiến Trúc:* Cũng sử dụng các lớp LSTM hoặc GRU.
* *Hoạt Động:* Sử dụng context vector từ encoder để tạo ra chuỗi đầu ra từng bước một. Tại mỗi bước, đầu vào của decoder là từ dự đoán từ bước trước đó hoặc từ thực tế trong giai đoạn huấn luyện.

**Attention Mechanism (Cơ Chế Chú Ý):**

* *Nhiệm Vụ:* Giúp decoder tập trung vào các phần khác nhau của chuỗi đầu vào khi tạo ra mỗi từ trong chuỗi đầu ra.
* *Kiến Trúc:* Tạo ra một trọng số chú ý cho mỗi từ đầu vào dựa trên trạng thái ẩn hiện tại của decoder.
* *Hoạt Động:* Tại mỗi bước giải mã, attention mechanism tính toán một trọng số cho mỗi trạng thái ẩn của encoder và tạo ra một vector ngữ cảnh mới (context vector) bằng cách kết hợp các trạng thái ẩn đó theo trọng số.

## Mô hình Encoder-Decoder

Đây là cấu trúc Seq2Seq truyền thống, với Encoder chuyển đổi câu đầu vào thành context vector và Decoder tái tạo câu đầu ra, kết hợp cơ chế tự chú ý (self-attention) để phát hiện và sửa các lỗi.

Phù hợp cho việc phát hiện và sửa lỗi chính tả bằng cách sinh ra câu không có lỗi từ câu có lỗi.

# KẾT LUẬN

Qua quá trình nghiên cứu và thực nghiệm của đề tài, chúng em đã rút ra các kết luận sau:

Mô hình BERT dựa trên encoder của transformer đã trở thành một nền tảng của ngành NLP hiện nay. Bằng cách áp dụng các mô hình này, chúng ta có thể nâng cao hiệu suất của các hệ thống NLP.

Mô hình GPT (Generative Pre-trained Transformer) dựa trên decoder của transformer là một trong những tiến bộ quan trọng trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP). Dựa trên kiến trúc Transformer, GPT có khả năng xử lý và sinh ra văn bản tự nhiên với chất lượng cao.

# TÀI LIỆU THAM KHẢO

**Tiếng Anh**

Vaswani, A. (2017). *Attention Is All You Need.*