

TRƯỜNG ĐẠI HỌC CẦN THƠ  
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



ĐỒ ÁN MÔN DEEP LEARNING - CT282 01

*Đề Tài*

XÂY DỰNG MÔ HÌNH SEQ2SEQ  
SINH CÂU TRUY VẤN SQL TỪ YÊU CẦU

*Sinh viên thực hiện:*

*Trần Gia Huy*

*B2016968*

*Kim Minh Thắng*

*B2007210*

*Cần Thơ, tháng 11 năm 2023*

# Mục lục

<b>1</b>	<b>Giới thiệu</b>	<b>3</b>
1.1	Bối cảnh đề tài . . . . .	3
1.2	Đặt vấn đề . . . . .	4
1.3	Hướng tiếp cận . . . . .	4
<b>2</b>	<b>Tập dữ liệu fine-tune</b>	<b>4</b>
2.1	Giới thiệu . . . . .	4
2.2	Tập dữ liệu b-mc2/sql-create-context . . . . .	5
2.3	Tập dữ liệu Clinton/Text-to-sql-v1 . . . . .	5
<b>3</b>	<b>Kiến trúc Transformer</b>	<b>5</b>
3.1	Tổng Quan . . . . .	5
3.2	Hai khối Encoder và Decoder . . . . .	6
3.3	Embedding và Positional Encoding . . . . .	6
3.3.1	Input Embedding và Output Embedding . . . . .	6
3.3.2	Positional Encoding . . . . .	7
3.4	Khối Multi-Headed Attention . . . . .	7
3.4.1	Scaled Dot-Product Attention . . . . .	7
3.4.2	Multi-headed Attention . . . . .	8
3.5	Add và LayerNormalization . . . . .	8
3.5.1	Add . . . . .	8
3.5.2	LayerNormalization . . . . .	9
3.6	Position-wise Feed Forward Network . . . . .	9
<b>4</b>	<b>Mô hình T5 Small</b>	<b>9</b>
4.1	Giới thiệu . . . . .	9
4.2	Pre-training T5 . . . . .	10
4.3	Tham số . . . . .	10
<b>5</b>	<b>Fine-tune mô hình sinh truy vấn SQL từ yêu cầu</b>	<b>10</b>
5.1	Tiền xử lý dữ liệu . . . . .	10
5.2	Fine-tune mô hình . . . . .	11
5.2.1	Thuật toán Lan Truyền Ngược (Backpropagation) . . . . .	11
5.2.2	Gradient Descent - Adam . . . . .	12
5.2.3	Hàm lỗi CrossEntropyLoss . . . . .	13
5.2.4	Fine-tune . . . . .	13
5.3	Đánh giá mô hình . . . . .	14
<b>6</b>	<b>Triển khai mô hình</b>	<b>14</b>

## Danh sách hình vẽ

1	Quy trình hoạt động của các LLM. Nguồn ảnh: <b>Analytics Yogi</b> . . . . .	3
2	Fine-tune LLM trên một downstream task. Nguồn ảnh: <b>Labellerr</b> . . . . .	4
3	Tổng quan kiến trúc Transformer. Nguồn ảnh: <b>Lena Voita</b> . . . . .	6
4	Chồng các khối Encoders và Decoders. Nguồn ảnh: <b>Jay Alammarr</b> . . . . .	6
5	Tầng Input và Output của Transformers. Nguồn ảnh: <b>Jalamar</b> . . . . .	7
6	Khối ScaleDotProductAttention. Nguồn ảnh: <b>PapersWithCode</b> . . . . .	7
7	Multi-headed Attention. Nguồn ảnh: <b>PapersWithCode</b> . . . . .	8
8	Tầng Add và LayerNormalization. Nguồn ảnh: <b>PyLessons</b> . . . . .	8
9	Khối Position-wise Feed Forward . Nguồn ảnh: <b>PyLessons</b> . . . . .	9
10	Mô hình T5 trong bài toán Text to Text . Nguồn ảnh: <b>Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer</b> . . . . .	9
11	Self-supervised Training với T5 . Nguồn ảnh: <b>Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer</b> . . . . .	10
12	Một vài dòng dữ liệu fine-tune trên custom dataset. . . . .	11
13	Tokenize chuỗi đầu vào. . . . .	11
14	Sơ đồ thuật toán Backpropagation. Nguồn ảnh: <b>SideSalad.com</b> . . . . .	12
15	Mình hoạt thuật toán gradient descent. Nguồn ảnh: <b>MLmuse</b> . . . . .	13
16	Biểu đồ giá trị lỗi (cột) trên tập train và tập test. . . . .	14
17	Biểu đồ giá trị lỗi (đường) trên tập train và tập test. . . . .	14
18	Demo mô hình trên web. . . . .	14

# XÂY DỰNG MÔ HÌNH SEQ2SEQ SINH CÂU TRUY VẤN SQL TỪ YÊU CẦU

Kim Minh Thắng, Trần Gia Huy

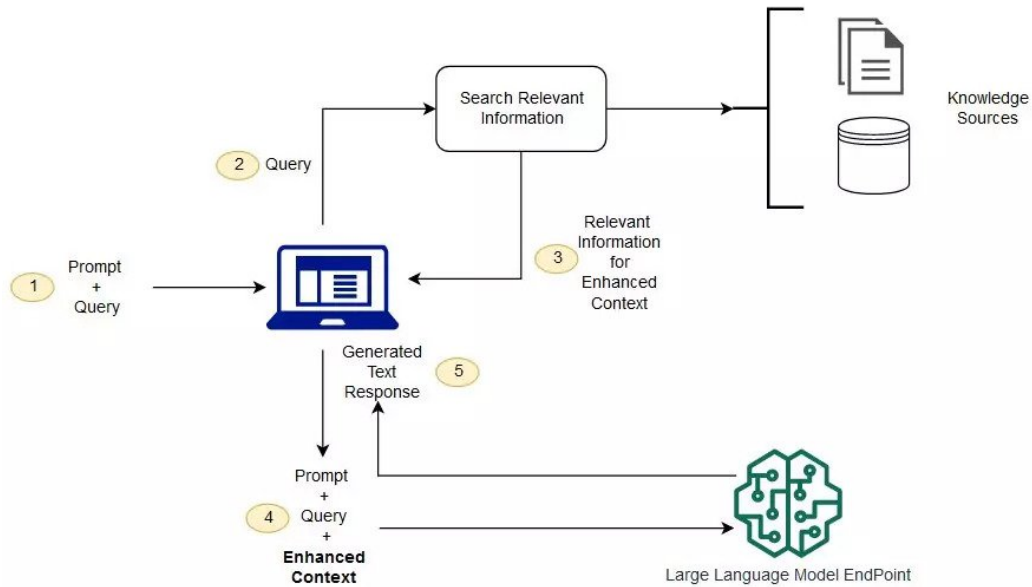
Tháng 11 năm 2023

## 1 Giới thiệu

### 1.1 Bối cảnh đề tài

**Ngôn ngữ truy vấn SQL** đã là một thành phần không thể thiếu của hệ quản trị cơ sở dữ liệu, sử dụng trong gần như mọi hệ thống thông tin hiện tại. Việc sử dụng ngôn ngữ truy vấn SQL là một công việc hằng ngày của các lập trình viên. Tuy nhiên, trong những câu lệnh SQL thường mang tính lặp lại, việc viết lại các câu lệnh này là một công việc tốn thời gian và công sức.

Những năm gần đây đã thể hiện sự phát triển mạnh mẽ của các mô hình **ngôn ngữ lớn (LLM)** - ứng dụng kiến trúc **Transformer** trong các bài toán xử lý ngôn ngữ tự nhiên (NLP).



Hình 1: Quy trình hoạt động của các LLM. Nguồn ảnh: **Analytics Yogi**

Ứng dụng lợi thế của các LLM, đồ án này đề xuất một mô hình **seq2seq** sử dụng kiến trúc **Transformer** để sinh câu truy vấn SQL từ yêu cầu người dùng. Được fine-tune lại từ mô hình **T5** với 2 tập dữ liệu về câu hỏi với khoảng **310 nghìn** câu hỏi và truy vấn SQL tương ứng.

## 1.2 Đặt vấn đề

Mục tiêu của đề tài chính là xây dựng một mô hình seq2seq sử dụng kiến trúc Transformer để sinh câu truy vấn SQL từ yêu cầu người dùng. Cụ thể hơn, mô hình cần sinh ra truy vấn SQL từ hai thông tin:

- **Context:** Ngữ cảnh của câu truy vấn, ở đây chính là chi tiết về các bảng của cơ sở dữ liệu cần cho truy vấn.

VD: *CREATE TABLE Customers (CustomerID int, CustomerName varchar(255));*

- **Question:** Câu hỏi tương trưng cho câu truy vấn, được viết bằng Tiếng Anh.

VD: *What is the name of the customer with ID 1?*

Từ hai thông tin trên, mô hình cần sinh ra truy vấn SQL tương ứng:

- **Answer:** Câu truy vấn SQL được sinh ra từ hai thông tin trên.

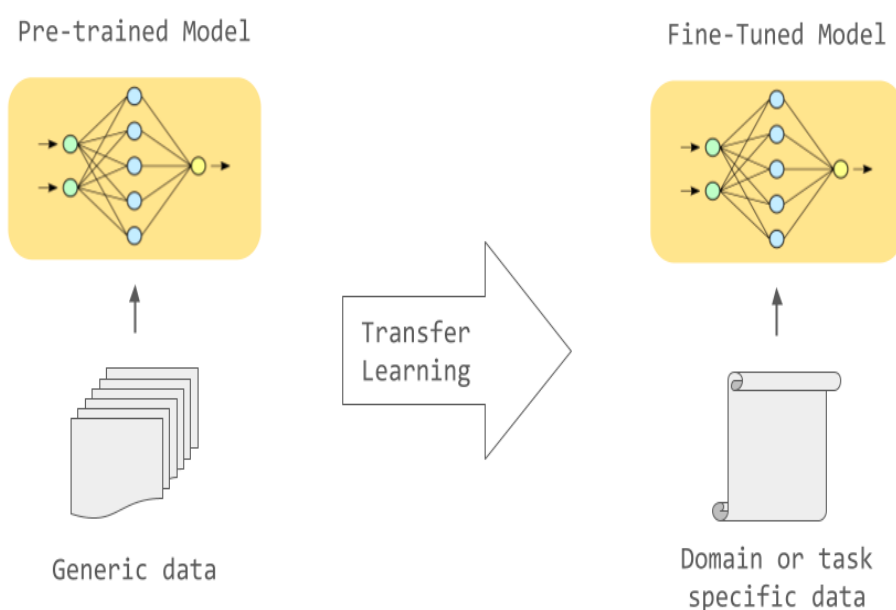
VD: *SELECT CustomerName FROM Customers WHERE CustomerID = 1;*

Mô hình cần đạt được một độ chính xác có thể chấp nhận được, giúp giảm thiểu thời gian và công sức của lập trình viên trong việc viết các câu truy vấn SQL.

## 1.3 Hướng tiếp cận

Để giải quyết vấn đề này, hướng tiếp cận của đề tài chính là fine-tune lại mô hình transformer là **T5** - mô hình đưa tất cả bài toán về dạng text to text.

Fine-tune lại một mô hình ngôn ngữ lớn cần một lượng dữ liệu không nhỏ, sử dụng hai tập dữ liệu **b-mc2/sql-create-context** và **Clinton/Text-to-sql-v1** với tổng cộng khoảng 310 nghìn dòng dữ liệu.



Hình 2: Fine-tune LLM trên một downstream task. Nguồn ảnh: **Labellerr**

## 2 Tập dữ liệu fine-tune

### 2.1 Giới thiệu

Cấu trúc chung của hai tập dữ liệu đều chứa 3 cột thể hiện:

- **Ngữ cảnh câu hỏi:** Các câu lệnh CREATE TABLE để thể hiện những bảng dữ liệu cần dùng trong query.
- **Câu hỏi:** Câu hỏi tương trưng cho câu truy vấn SQL, viết bằng Tiếng Anh.
- **Truy vấn SQL:** Câu truy vấn SQL được sinh ra từ hai thông tin trên.

Ngôn ngữ được sử dụng trong hai tập dữ liệu là Tiếng Anh, các câu lệnh SQL đều được viết đúng cú pháp, rất ít dữ liệu bị trùng lặp và bị thiếu.

## 2.2 Tập dữ liệu b-mc2/sql-create-context

Tập dữ liệu bao gồm **78,577 dòng và 3 cột** 'question': ngôn ngữ tự nhiên cho câu hỏi, 'context' các câu lệnh CREATE TABLE, 'answer': Câu truy vấn SQL.

Được xây dựng từ *WikiSQL* và *Spider*, mục tiêu của tập dữ liệu là dành cho các LLM text sang SQL. Sử dụng các lệnh CREATE TABLE để đưa ra ngữ cảnh cho câu lệnh rõ ràng hơn mà không cần đưa vào dữ liệu thật, tránh lộ thông tin riêng, giảm độ dài.

Chi tiết tập dữ liệu đọc thêm tại: [1]

## 2.3 Tập dữ liệu Clinton/Text-to-sql-v1

Tập dữ liệu bao gồm **262.000 dòng và 5 cột**

- **instruction:** Câu hỏi để sinh ra truy vấn SQL.
- **input:** các câu lệnh CREATE TABLE.
- **response:** Câu truy vấn SQL đích.
- **source:** Nguồn của dòng dữ liệu.
- **text:** Diễn giải đầy đủ, kết hợp giữa cột instruction và input.

Chi tiết tập dữ liệu đọc thêm tại: [2]

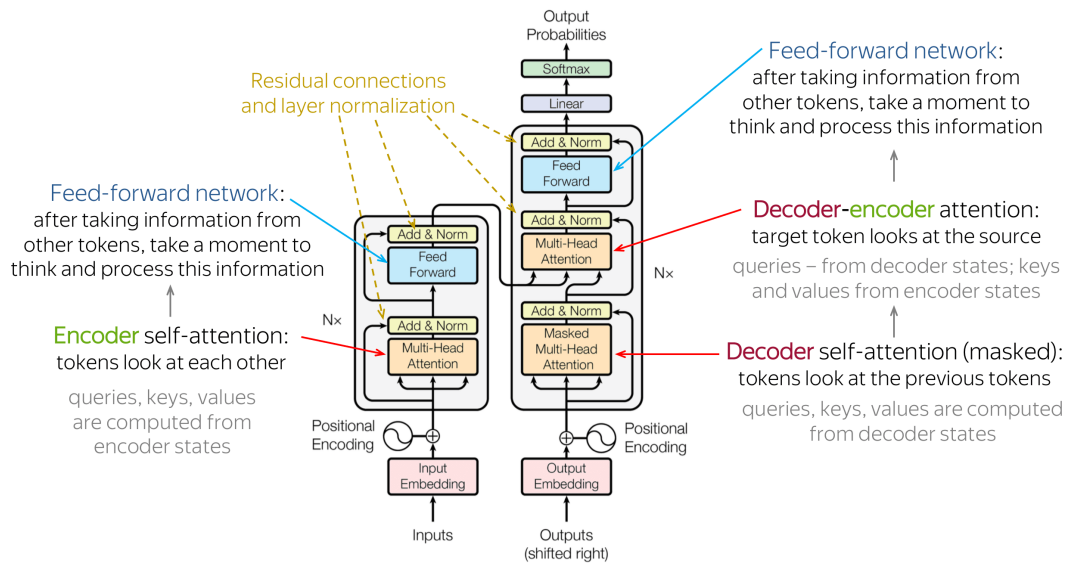
# 3 Kiến trúc Transformer

## 3.1 Tổng Quan

Transformer là một mô hình học sâu được giới thiệu bởi Vaswani và các cộng sự [9] vào năm 2017, đây là mô hình đã đạt được kết quả nổi bật trong nhiều nhiệm vụ xử lý ngôn ngữ tự nhiên (NLP). Transformer được xây dựng dựa trên cơ chế Attention (Attention mechanism, tạm dịch: cơ chế tập trung), cho phép mô hình có khả năng “tập trung” vào các phần khác nhau của đầu vào (Input) trong quá trình học.

Mô hình Transformer sử dụng kiến trúc đa đầu vào (Multi-Head Input) và đa đầu ra (Multi-Head Output) để xử lý đầu vào và đầu ra theo cách đồng thời, tức là không cần xử lý tuần tự từng phần như các mô hình trước đây. Điều này giúp giải quyết vấn đề độ dài phụ thuộc trong ngôn ngữ (Long-range Dependency) và giúp mô hình có khả năng hiểu ngữ cảnh (Contextual Understanding) của dữ liệu đầu vào (Input).

Mô hình Transformer đã được ứng dụng rộng rãi trong nhiều tác vụ xử lý ngôn ngữ tự nhiên, bao gồm dịch máy, phân loại văn bản, dự đoán từ, tóm tắt văn bản, hỏi đáp, và nhiều tác vụ ngôn ngữ tự nhiên khác. BERT (Bidirectional Encoder Representations from Transformers), GPT (Generative Pre-trained Transformer), và T5 (Text-to-Text Transfer Transformer) là những mô hình NLP nổi tiếng được xây dựng trên kiến trúc của mô hình Transformer.



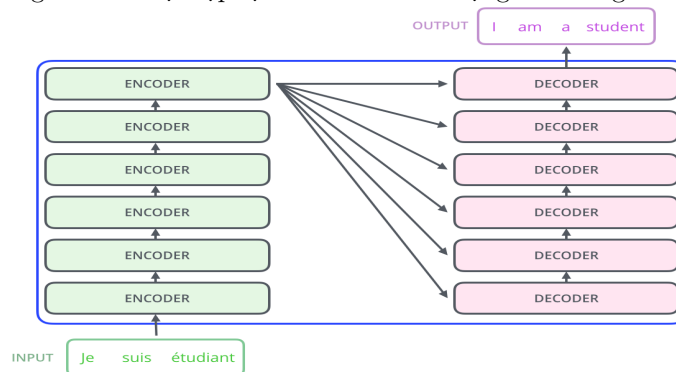
Hình 3: Tổng quan kiến trúc Transformer. Nguồn ảnh: **Lena Voita**

- Mô hình Transformer sử dụng định dạng mã hóa - giải mã (Encoder - Decoder) tương tự như Seq2Seq.
- Những khối được đề xuất trong Transformer, bao gồm Scaled Dot-Product Attention và Multi-Head Attention, là trọng tâm của kiến trúc này, và chúng được xếp hàng loạt và thực hiện song song (parallel) trong mô hình.
- Khác với kiến trúc của RNN, transformer, không có cấu trúc BPTT tương tự và tính toán có thể được thực hiện song song, do đó mô hình có khả năng hoạt động hiệu quả hơn so với kiến trúc RNN.

### 3.2 Hai khối Encoder và Decoder

Đầu vào và đầu ra của Encoder có cùng kích thước. Do đó, cấu trúc Encoder có thể được lặp lại nhiều lần để sử dụng một cách dễ dàng.

Tương tự, Decoder cũng có thể được lặp lại nhiều lần dưới dạng khối để giải mã đầu ra.



Hình 4: Chồng các khối Encoders và Decoders. Nguồn ảnh: **Jay Alammar**

### 3.3 Embedding và Positional Encoding

#### 3.3.1 Input Embedding và Output Embedding

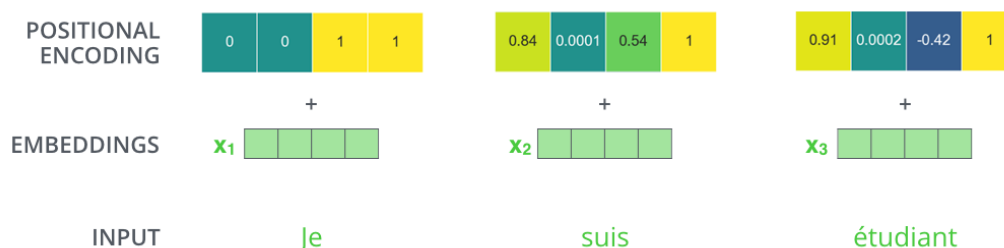
Trong Transformer, Input Embedding và Output Embedding đều là tầng Embedding để chuyển đổi đầu vào thành vector với số thực.

Đây là tầng đầu tiên trong kiến trúc Transformer, nhận đầu vào là một one-hot-encoded vector là một vector có độ dài bằng với số từ trong từ điển,

Mỗi từ đi vào sẽ được tầng này tạo ra một vector đặc, có số chiều nhỏ hơn ban đầu.

### 3.3.2 Positional Encoding

Kết quả của tầng Input Embedding hoặc Output Embedding sẽ được đi qua một phép tính Positional Encoding.



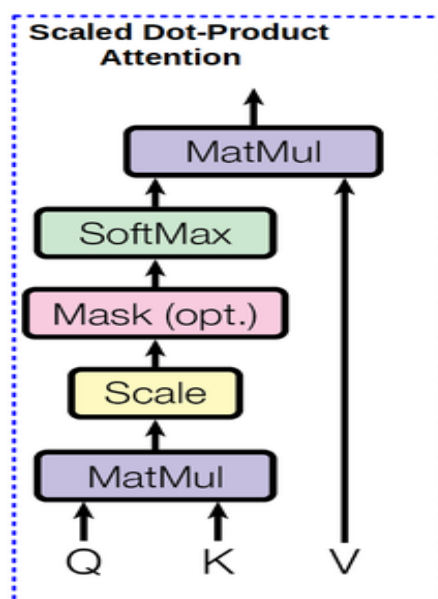
Hình 5: Tầng Input và Output của Transformers. Nguồn ảnh: **Jalamar**

Do Transformer không tính toán tuần tự mà xử lý một cách song song, nên nó không có khả năng nhận biết được vị trí của từ trong câu. Mục tiêu của phép tính Positional Encoding là để giữ lại vị trí cho câu input, không làm mất thứ tự và ngữ nghĩa câu.

## 3.4 Khối Multi-Headed Attention

### 3.4.1 Scaled Dot-Product Attention

Attention là một cơ chế cho phép mô hình tập trung vào những phần quan trọng của đầu vào (Input) trong quá trình học. Cơ chế Attention được sử dụng trong nhiều mô hình NLP, bao gồm cả mô hình Transformer.



Hình 6: Khối ScaleDotProductAttention. Nguồn ảnh: **PapersWithCode**

Đầu vào của Scaled Dot-Product Attention là 3 ma trận Q, K, V có cùng số chiều.

Đầu ra của Scaled Dot-Product Attention là ma trận có cùng số hàng với ma trận Q và cùng số cột với ma trận V thể hiện trọng số attention của nó.

Dữ liệu trong khối này sẽ được xử lý như sau:



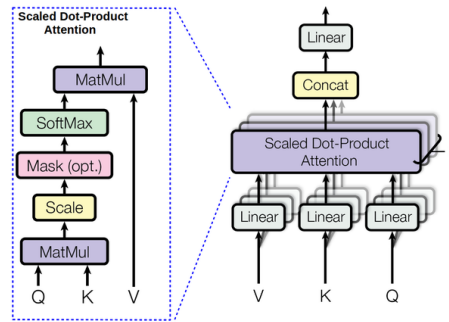
- Hai ma trận Q và K sẽ được nhân lại với nhau, sau đó chia cho căn bậc hai của số chiều của ma trận K (tầng Scale) nhằm ngăn giá trị tăng lên quá lớn.
- Sau tầng Scale, nếu có tầng Mask thì sẽ được thực hiện để ngăn chặn Attention đến các kết nối không hợp lệ (illegal connection).
- Sau đó, ma trận sau khi được nhân sẽ được đưa vào hàm softmax để chuẩn hóa giá trị Attention. Giúp mô hình tự tin hơn với các trọng số.
- Cuối cùng, ma trận sau khi được chuẩn hóa sẽ được nhân với ma trận V để tạo ra đầu ra của khối Scaled Dot-Product Attention. Đây chính là trọng số Attention của khối.

### 3.4.2 Multi-headed Attention

Thay vì chỉ sử dụng một khối Scaled Dot-Product Attention, Transformer sử dụng nhiều khối Scaled Dot-Product Attention song song (parallel) để tăng khả năng học của mô hình, hiểu ngữ nghĩa theo nhiều khối.

Đầu ra của mỗi khối Scaled Dot-Product Attention sẽ được nối với nhau và đi qua một tầng Linear để tạo ra đầu ra của khối Multi-Headed Attention.

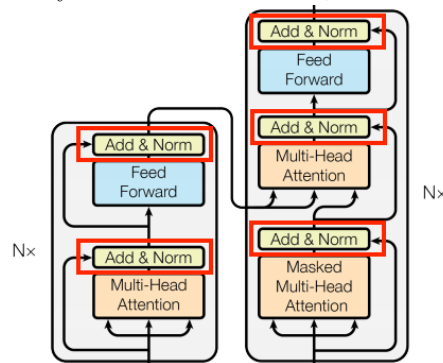
Kết quả của khối là một ma trận có cùng số hàng với ma trận Q và cùng số cột với ma trận V thể hiện trọng số attention của nó.



Hình 7: Multi-headed Attention. Nguồn ảnh: **PapersWithCode**

## 3.5 Add và LayerNormalization

Trong Transformer, Add & Norm được sử dụng để kết hợp thông tin từ các tầng khác nhau trong mạng. Trong quá trình này, đầu ra của một tầng sẽ được cộng với đầu vào ban đầu của tầng đó (skip-connection), sau đó chuẩn hóa lại với Layer Normalization để tạo ra đầu ra cuối cùng.



Hình 8: Tầng Add và LayerNormalization. Nguồn ảnh: **PyLessons**

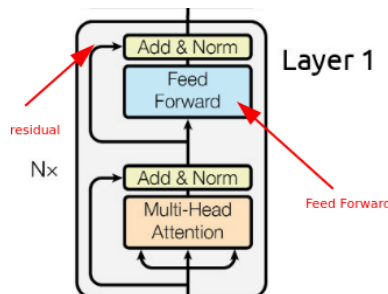
### 3.5.1 Add

Trong bước này, đầu ra của lớp sẽ được cộng với vector đầu vào ban đầu. Việc cộng này giúp cập nhật thông tin từ các phần khác nhau của kiến trúc và giúp tránh hiện tượng mất thông tin (Vanishing Gradient) trong quá trình huấn luyện.

### 3.5.2 LayerNormalization

Sau khi được cộng với đầu vào ban đầu, đầu ra của lớp sẽ được chuẩn hóa lại với Layer Normalization. Layer Normalization là một phép chuẩn hóa dữ liệu đầu ra của một tầng theo của ma trận đầu ra. Quá trình chuẩn hóa giúp cải thiện tính ổn định của mô hình

### 3.6 Position-wise Feed Forward Network

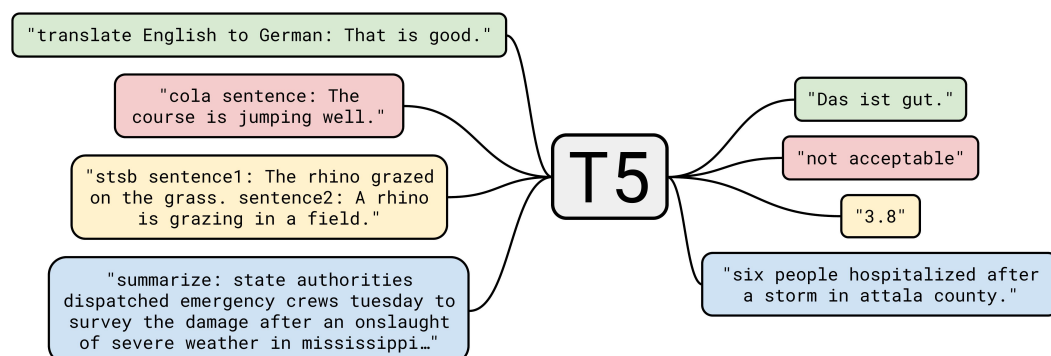


Hình 9: Khối Position-wise Feed Forward . Nguồn ảnh: **PyLessons**

Các vector đầu vào (là các vector đại diện cho từ) sẽ được truyền qua tầng Fully Connected Layer với hàm kích hoạt ReLU, và cuối cùng đi qua một tầng Fully ConnectedLayer nữa. Đầu ra của tầng thứ hai cũng chính là đầu ra của Position-wise Feed-Forward.

Mục đích chính là xử lý tiếp attention output từ tầng trước đó, giúp mô hình có thể học được các mối quan hệ giữa các từ trong câu.

## 4 Mô hình T5 Small



Hình 10: Mô hình T5 trong bài toán Text to Text . Nguồn ảnh: **Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer**

### 4.1 Giới thiệu

Mô hình **T5**, hay **Text-To-Text Transfer Transformer**, là một kiến trúc transformer nổi tiếng trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP). Mô hình này được giới thiệu bởi Google Research và ra mắt lần đầu tiên trong bài báo "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer" vào năm 2019.

Mục tiêu chính của T5 là đơn giản hóa các nhiệm vụ xử lý ngôn ngữ tự nhiên thành một định dạng chung gọi là **"text-to-text"**. Điều này có nghĩa là mô hình được huấn luyện để xử lý mọi công việc NLP thông qua việc chuyển đổi từ đầu vào thành đầu ra dưới dạng văn bản. Điều này bao gồm cả các nhiệm vụ như dịch máy, tóm tắt văn bản, phân loại văn bản, và nhiều nhiệm vụ khác.

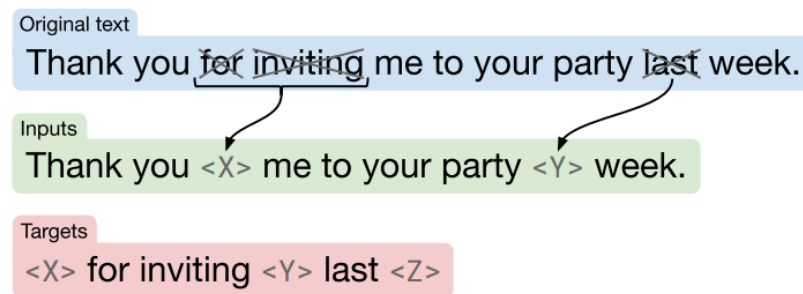
Được huấn luyện bằng phương pháp **teacher-forcing** trên một lượng lớn dữ liệu, trong đó có tập dữ liệu C4 - một tập dữ liệu lớn được thu thập ở các trang web trên Internet. Mô hình được huấn luyện trên nhiều tác vụ khác nhau, nhờ vậy, có khả năng đáp ứng cho transfer learning.

## 4.2 Pre-training T5

Việc pre-training của mô hình T5 được thực hiện với cả **Self-supervised Learning** và **Supervised Learning**.

- Với **Self-supervised Learning**, mô hình được huấn luyện bằng cách sử dụng các **Corrupted Token**.

Bằng cách ngẫu nhiên xóa bỏ 15% các token và thay chúng thành các "sentinel" token, ta sẽ tạo được các câu bị chỉnh sửa. Những câu bị chỉnh sửa sẽ là input cho khối encoder, câu gốc sẽ là input cho khối decoder, và các token bị bỏ ra giới hạn bởi các "sentinel token" sẽ là nhãn



Hình 11: Self-supervised Training với T5 . Nguồn ảnh: **Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer**

- Với **Supervised Learning**, mô hình được huấn luyện trên các tác vụ downstream được cung cấp bởi chuẩn GLUE và SuperGLUE (được đưa về dạng text to text).

## 4.3 Tham số

Với mô hình **T5 Small**, kiến trúc của nó sẽ dựa trên Transformer với:

- 6 khối encoders và 6 khối decoders
- 8 headed-attention
- Chiều dài của feed-forward layer là 2048
- Chiều vector đầu ra là 512
- Số lượng tham số là khoảng 60 triệu tham số.

## 5 Fine-tune mô hình sinh truy vấn SQL từ yêu cầu




Để có được mô hình sinh truy vấn SQL từ chuỗi câu hỏi và chuỗi ngữ cảnh, mục tiêu đề tài là fine-tune lại **mô hình T5** với hai dataset **sql-create-context** (78 nghìn dòng) và **Text-to-sql-v1** (262 nghìn dòng)

### 5.1 Tiền xử lý dữ liệu

Đầu tiên, dữ liệu được load từ huggingface datasets, sau đó được tiền xử lý theo các bước sau:

- Loại bỏ các dòng dữ liệu bị thiếu.
- Loại bỏ các dòng dữ liệu bị trùng lặp (dựa vào question).
- Loại bỏ các cột dữ liệu không cần thiết và merge hai tập dữ liệu lại.

Cuối cùng, ta thu được một dataframe có shape là (340785, 3), không có dữ liệu null và trùng lặp.

question string · lengths	answer string · lengths	context string · lengths
 A histogram showing the distribution of question string lengths. The x-axis is labeled 'string · lengths' and has a major tick at 12. The y-axis represents frequency. The distribution is right-skewed, with most questions being short (lengths 12-20) and a few longer ones. The total count is 244.	 A histogram showing the distribution of answer string lengths. The x-axis is labeled 'string · lengths' and has a major tick at 18. The y-axis represents frequency. The distribution is right-skewed, with most answers being short (lengths 18-25) and a few longer ones. The total count is 557.	 A histogram showing the distribution of context string lengths. The x-axis is labeled 'string · lengths' and has a major tick at 27. The y-axis represents frequency. The distribution is right-skewed, with most contexts being short (lengths 27-35) and a few longer ones. The total count is 489.
How many heads of the departments are older than 56 ?	SELECT COUNT(*) FROM head WHERE age > 56	CREATE TABLE head (age INTEGER)
List the name, born state and age of the heads of departments ordered by age.	SELECT name, born_state, age FROM head ORDER BY age	CREATE TABLE head (name VARCHAR, born_state VARCHAR, age VARCHAR)
List the creation year, name and budget of each department.	SELECT creation, name, budget_in_billions FROM department	CREATE TABLE department (creation VARCHAR, name VARCHAR, budget_in_billions VARCHAR)
What are the maximum and minimum budget of the departments?	SELECT MAX(budget_in_billions), MIN(budget_in_billions) FROM department	CREATE TABLE department (budget_in_billions INTEGER)

Hình 12: Một vài dòng dữ liệu fine-tune trên custom dataset.

Tiếp theo ta sẽ chỉ ra cách tokenize dữ liệu về dạng vector để đưa vào mô hình:

- Đầu tiên, ta sẽ thêm vào một prefix là 'query for: ' để đánh dấu bắt đầu câu hỏi, thêm prefix là 'tables: ' để đánh dấu là ngữ cảnh.
- Với mỗi từ, xây dựng từ điển và đánh mỗi từ với 1 id riêng biệt
- Tokenize chuỗi đầu vào thành input-ids, thêm token '</s>' ở cuối để đánh dấu hết câu, thêm padding token để đảm bảo độ dài của input là như nhau (200 tokens).
- Tokenize chuỗi đầu đích thành decoder-input-ids, dịch tất cả các token sang phải, thêm start: token '<s>' ở đầu, thêm token '</s>' ở cuối để đánh dấu hết câu, thêm padding token để đảm bảo độ dài của input là như nhau (128 tokens).
- Khởi tạo attention mask, đánh dấu các token padding là 0, các token thật là 1.

Như vậy với các chuỗi là câu hỏi, ngữ cảnh và truy vấn SQL, ta đã có thể tokenize chúng thành các vector input-ids, decoder-input-ids, attention mask và decoder-attention-mask.

Ví dụ với chuỗi đầu vào là

query for: what are the classes that having name start with 'A-' tables: `CREATE TABLE classes (id INT, name TEXT);` , ta sẽ tokenize nó thành:

[illegible]

Hình 13: Tokenize chuỗi đầu vào.

### Xây dựng Dataloader và phân chia tập dữ liệu:

- Tách tập dữ liệu thành 2 phần: 80% train và 20% test, trộn ngẫu nhiên các dòng cho hai tập
- Xây dựng lớp Dataloader, hỗ trợ quá trình train và test. Giúp lấy ra dòng dữ liệu theo index, trả về input-ids, decoder-input-ids, attention mask và decoder-attention-mask.

## 5.2 Fine-tune mô hình

### 5.2.1 Thuật toán Lan Truyền Ngược (Backpropagation)

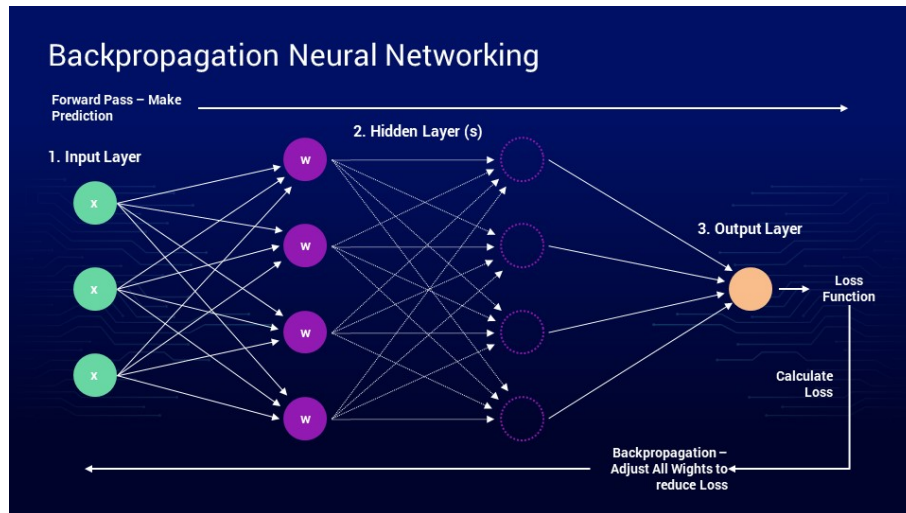
Thuật toán lan truyền ngược là một thuật toán quan trọng trong học máy, được sử dụng để cập nhật các trọng số của mạng nơ-ron nhân tạo. Thuật toán này sẽ tính toán đạo hàm của hàm lỗi (L) theo từng trọng số (w):

Giả sử  $L$  là hàm lỗi (loss function) và  $w_i$  là trọng số thứ  $i$  trong mô hình. Đạo hàm của hàm lỗi theo  $w_i$  được ký hiệu là  $\frac{\partial L}{\partial w_i}$ .

Thuật toán Lan Truyền Ngược (Backpropagation) được sử dụng kết hợp với một thuật toán tối ưu hóa như Gradient Descent để cập nhật các trọng số.

Trong quá trình huấn luyện mô hình sử dụng thuật toán lan truyền ngược, ta cần thực hiện 3 bước sau để tính được đạo hàm của hàm lỗi theo từng trọng số:

- Forward Pass: Tính toán giá trị của hàm lỗi (L) dựa trên các trọng số (w) hiện tại.
- Compute Local Gradients: Tính giá trị đạo hàm tại mỗi nút (node) trong mạng nơ-ron.
- Backward Pass: Tính toán đạo hàm của hàm lỗi (L) theo từng trọng số (w) hiện tại.



Hình 14: Sơ đồ thuật toán Backpropagation. Nguồn ảnh: [SideSalad.com](https://www.side salad.com)

Ở bước cuối cùng, sau khi có được giá trị hàm lỗi và đạo hàm tại mỗi nút mạng, ta tiến hành tính đạo hàm của hàm lỗi theo từng trọng số (w) hiện tại. Kết hợp với *Chain Rule* có quy tắc như sau: Giả sử  $f(x)$  và  $g(x)$  là hai hàm có đạo hàm. Quy tắc chuỗi cho biết rằng đạo hàm của hàm hợp  $h(x) = f(g(x))$  được tính theo công thức:

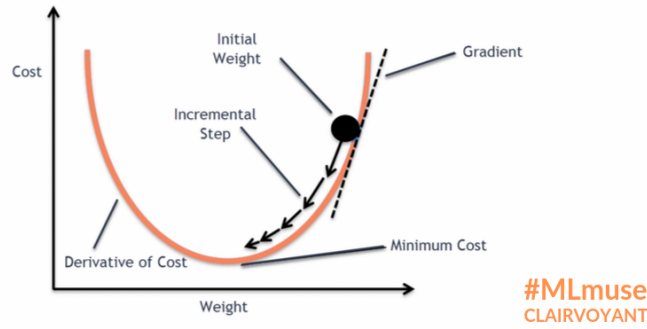
$$(h \circ g)'(x) = f'(g(x)) \cdot g'(x)$$

Trong ký hiệu toán học:

$$\frac{d}{dx}[f(g(x))] = f'(g(x)) \cdot g'(x)$$

### 5.2.2 Gradient Descent - Adam

Trong các bài toán tối ưu hàm lỗi, chúng ta thường tìm giá trị nhỏ nhất của 1 hàm số nào đó, mà hàm số đạt giá trị nhỏ nhất khi đạo hàm bằng 0. Nhưng đối với các hàm số nhiều biến thì đạo hàm rất phức tạp, thậm chí là bất khả thi. Nên thay vào đó người ta tìm điểm gần với điểm cực tiểu nhất và xem đó là nghiệm bài toán. Đây chính là ý tưởng của phương pháp **Gradient Descent**.



Hình 15: Minh hoạt thuật toán gradient descent. Nguồn ảnh: **MLmuse**

Phương pháp gradient descent được sử dụng để tối ưu hóa hàm lỗi  $L$  trong mô hình máy học. Công thức cập nhật trọng số trong mỗi bước gradient descent có thể được biểu diễn như sau:

$$w_i = w_i - \alpha \frac{\partial L}{\partial w_i}$$

Để đi về phía có giá trị cực tiểu, ta cần cập nhật trọng số theo hướng ngược lại với đạo hàm của hàm lỗi. Do vậy, có được công thức trên với  $\alpha$  là learning rate.

### 5.2.3 Hàm lỗi CrossEntropyLoss

Trong huấn luyện mô hình phân loại, hàm lỗi Cross Entropy Loss được sử dụng để đánh giá độ chính xác của mô hình. Có vai trò quan trọng trong việc đánh giá mô hình cho bài toán phân loại nhiều lớp.

Hàm Cross Entropy Loss cho bài toán phân loại nhiều lớp:

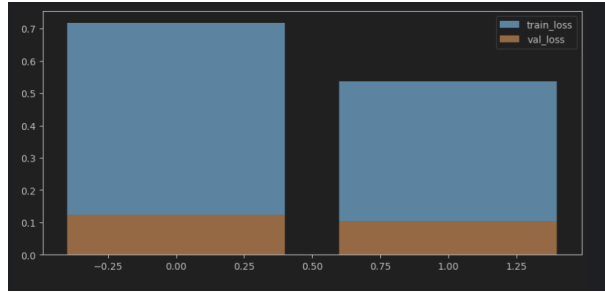
$$L(\mathbf{y}, \hat{\mathbf{y}}) = - \sum_{i=1}^N y_i \log(\hat{y}_i)$$

### 5.2.4 Fine-tune

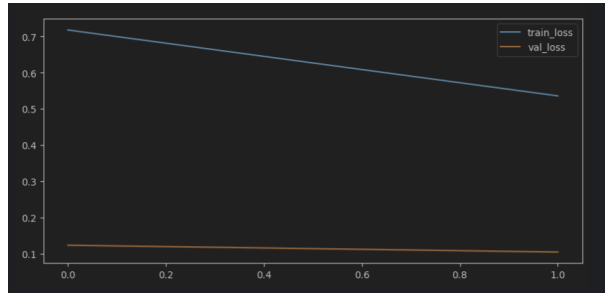
Do thiếu tài nguyên tính toán, nên việc huấn luyện cho mô hình chỉ được giữ ở mức tối thiểu. Các tham số được lựa chọn để fine-tune lại mô hình như sau:

- **Epoch:** 2
- **Batch size:** 32
- **Learning rate:** 3e-5
- **Optimizer:** AdamW
- **Loss function:** CrossEntropyLoss

Hai biểu đồ thể hiện giá trị lỗi trên tập train và tập test của mô hình:



Hình 16: Biểu đồ giá trị lỗi (cột) trên tập train và tập test.



Hình 17: Biểu đồ giá trị lỗi (đường) trên tập train và tập test.

Trong quá trình train, những padding token sẽ được đánh dấu là 100 (ở bước tiền xử lý). Với giá trị này, hàm CrossEntropyLoss sẽ không tính toán đạo hàm của các token padding, giữ cho mô hình không học được thông tin từ các token padding.

### 5.3 Đánh giá mô hình

## 6 Triển khai mô hình

Mô hình được demo trên nền tảng web, hỗ trợ sinh câu lệnh từ một chuỗi context và một chuỗi input Công nghệ sử dụng:

- **Frontend:** ReactJS
- **Backend:** FastAPI + Python

**Text2SQL**

Your create table SQL

```
1 CREATE TABLE classes (id INT, name TEXT);
```

how many classes that having name start with 'A'

Submit

```
1 SELECT COUNT(*) FROM CLASSES WHERE NAME LIKE 'A-%'
```

Hình 18: Demo mô hình trên web.

## References

- [1] b-mc2. *sql-create-context*. HuggingFace. 2022. URL: <https://huggingface.co/datasets/b-mc2/sql-create-context>.
- [2] Clinton. *Text-to-sql-v1*. HuggingFace. 2023. URL: <https://huggingface.co/datasets/Clinton/Text-to-sql-v1>.