TRƯỜNG ĐẠI HỌC THỦY LỢI

KHOA CÔNG NGHỆ THÔNG TIN



BÀI TẬP LỚN

**HỌC PHẦN: ĐỒ ÁN TRÍ TUỆ NHÂN TẠO**

**ĐỀ TÀI: XÂY DỰNG MÔ HÌNH DỊCH BÌNH LUẬN SẢN PHẨM ANH - VIỆT**

Giáo viên hướng dẫn:

Sinh viên/nhóm sinh viên thực hiện:

Hà Nội, tháng 1, năm 2024

**Mục Lục**

[**PHẦN 1: LÝ THUYẾT 2**](#_octyhmypbywb)

[1.1 Dịch Máy 2](#_753rzqbautlk)

[1.2 Mô hình dịch máy sử dụng 5](#_yk9rxx3tfvg4)

[**PHẦN 2: ỨNG DỤNG 8**](#_nme3uwhh4qye)

[2.1 Mô tả tệp dữ liệu 8](#_yltfk3s3w42f)

[2.2 Thiết kết mô hình 8](#_rkqanf8f73e0)

[2.3 Kết luận 19](#_qm3tmce2jerb)

[**Tài Liệu Tham khảo 20**](#_uvlkxkvlbmau)

# **PHẦN 1: LÝ THUYẾT**

## **1.1 Dịch Máy**

* **Định nghĩa:**

Dịch máy là quá trình sử dụng trí tuệ nhân tạo để tự động dịch văn bản từ ngôn ngữ này sang ngôn ngữ khác mà không cần sự tham gia của con người. Phương pháp dịch máy hiện đại không còn dừng lại ở việc dịch bám sát bản gốc để truyền đạt ý nghĩa đầy đủ của văn bản ngôn ngữ gốc bằng ngôn ngữ đích. Mà giờ đây phương pháp này còn phân tích tất cả các thành phần của văn bản, đồng thời nhận biết cả cách thức các từ ảnh hưởng đến nhau.[1]

* **Lịch sử:**

Ý tưởng sử dụng máy tính để dịch tự động ngôn ngữ của con người lần đầu tiên xuất hiện vào đầu những năm 1950. Tuy nhiên, vào thời điểm đó, mức độ phức tạp của bản dịch cao hơn nhiều so với ước tính ban đầu của các nhà khoa học máy tính. Công việc này đòi hỏi sức mạnh xử lý và lưu trữ dữ liệu khổng lồ, vượt quá khả năng của những chiếc máy đời đầu.

Vào đầu những năm 2000, phần mềm, dữ liệu và phần cứng máy tính có khả năng thực hiện hoạt động dịch máy cơ bản. Những nhà phát triển ban đầu đã sử dụng cơ sở dữ liệu thống kê của các ngôn ngữ để hướng dẫn máy tính dịch văn bản. Quá trình này mất rất nhiều thời gian và công sức. Các nhà phát triển đều phải bắt đầu phát triển ngôn ngữ từ đầu với mỗi ngôn ngữ được thêm vào. Kể từ đó, dịch vụ dịch máy đã phát triển cả về tốc độ và độ chính xác, và một số chiến lược dịch máy khác nhau đã xuất hiện.[1]

* **Lợi ích:**

Biên dịch viên sử dụng dịch vụ dịch máy để dịch nhanh hơn và hiệu quả hơn. Sau đây là một số lợi ích của dịch vụ dịch máy mà chúng tôi đưa ra:

* Hỗ trợ dịch tự động

Dịch máy cung cấp một điểm khởi đầu tốt cho biên dịch viên chuyên nghiệp. Nhiều hệ thống quản lý dịch thuật tích hợp một hoặc nhiều mô hình dịch máy vào quy trình làm việc của họ. Họ có chế độ cài đặt để chạy bản dịch tự động, sau đó gửi cho biên dịch viên để tiến hành chỉnh sửa bản dịch máy đó.

* Tốc độ và khối lượng

Dịch vụ dịch máy hoạt động rất nhanh, dịch hàng triệu từ gần như ngay lập tức. Dịch vụ này có thể dịch một lượng lớn dữ liệu, chẳng hạn như cuộc trò chuyện theo thời gian thực hoặc các vụ án tố tụng có quy mô lớn. Dịch vụ này cũng có thể xử lý các tài liệu bằng tiếng nước ngoài, tìm kiếm các thuật ngữ có liên quan và ghi nhớ các thuật ngữ đó cho các ứng dụng trong tương lai.

* Nhiều lựa chọn ngôn ngữ

Có nhiều nhà cung cấp dịch vụ dịch máy lớn hỗ trợ từ 50 đến hơn 100 ngôn ngữ. Hoạt động dịch cũng diễn ra đồng thời cho nhiều ngôn ngữ, điều này rất hữu ích cho việc phát hành sản phẩm trên toàn cầu cũng như việc cập nhật tài liệu.

* Dịch vụ dịch tiết kiệm chi phí

Dịch máy làm tăng hiệu suất và giúp cung cấp bản dịch nhanh hơn, giảm thiểu thời gian đưa ra thị trường. Do sự tham gia của con người vào quá trình dịch ít hơn nên dịch vụ dịch máy cung cấp các bản dịch cơ bản nhưng cũng có giá trị, giúp giảm thiểu cả chi phí và thời gian trả dự án. Ví dụ: trong các dự án lớn, bạn có thể tích hợp dịch vụ dịch máy với hệ thống quản lý nội dung của mình để tự động gắn thẻ và sắp xếp nội dung trước khi dịch nội dung đó sang các ngôn ngữ khác.[1]

* **Trường hợp sử dụng:**

Dịch máy thường được dùng trong một số trường hợp, chẳng hạn như những trường hợp được đưa ra dưới đây:

* Thông tin liên lạc nội bộ

Đối với một công ty hoạt động ở nhiều quốc gia trên thế giới, việc quản lý thông tin liên lạc có thể trở nên khó khăn. Các nhân viên có thể có kỹ năng ngôn ngữ khác nhau và một số có thể chưa đủ kỹ năng về ngôn ngữ để hiểu được ngôn ngữ chính thức của công ty. Dịch máy giúp giảm thiểu hoặc xóa bỏ rào cản ngôn ngữ trong giao tiếp. Mọi người có thể nhanh chóng có bản dịch của văn bản và hiểu thông điệp cốt lõi của nội dung. Bạn có thể sử dụng dịch vụ này để dịch các bài thuyết trình, bản tin công ty và các thông tin liên lạc thông thường khác.

* Thông tin liên lạc bên ngoài công ty

Các công ty sẽ sử dụng dịch vụ dịch máy để giao tiếp hiệu quả hơn với các bên liên quan và khách hàng ở bên ngoài. Ví dụ: bạn có thể dịch các tài liệu quan trọng sang các ngôn ngữ khác cho đối tác và khách hàng toàn cầu. Nếu một cửa hàng trực tuyến hoạt động ở nhiều quốc gia, dịch vụ dịch máy có thể dịch các bài đánh giá sản phẩm để khách hàng có thể đọc được các đánh giá đó bằng ngôn ngữ của họ.

* Phân tích dữ liệu

Một số loại hình dịch máy có thể xử lý hàng triệu nhận xét do người dùng đưa ra và cung cấp bản dịch có độ chính xác cao trong một khoảng thời gian ngắn. Một số công ty sẽ dịch một lượng lớn nội dung được đăng tải trên mạng xã hội và các trang web mỗi ngày, đồng thời dịch các nội dung để phục vụ mục đích phân tích. Ví dụ: các công ty đó có thể tự động phân tích ý kiến của khách hàng được viết bằng nhiều ngôn ngữ.

* Dịch vụ khách hàng trực tuyến

Nhờ có công nghệ dịch máy, các thương hiệu có thể tương tác với khách hàng trên toàn thế giới, bất kể họ nói ngôn ngữ nào.

* Nghiên cứu pháp lý

Bộ phận pháp lý sử dụng dịch vụ dịch máy để chuẩn bị các tài liệu pháp lý ở nhiều quốc gia. Dịch vụ dịch máy giúp chúng ta có sẵn một lượng lớn nội dung để phân tích. Những dữ liệu này có thể sẽ rất khó xử lý nếu chúng được viết bằng các ngôn ngữ khác.[1]

* **Phương pháp dịch máy:**
* Dịch máy thống kê (Statistical Machine Translation - SMT): Đây là phương pháp truyền thống sử dụng các mô hình thống kê để xác định cách dịch từ ngôn ngữ nguồn sang ngôn ngữ đích. Các mô hình SMT dựa trên các đặc trưng thống kê của các cặp câu tương đương trong các tập dữ liệu huấn luyện.
* Dịch máy cơ sở trên quy tắc (Rule-based Machine Translation - RBMT): Phương pháp này dựa trên các quy tắc ngữ pháp và từ điển để thực hiện dịch. Các hệ thống RBMT sử dụng các bộ quy tắc ngữ pháp và từ điển để phân tích và tái tạo cú pháp của câu từ ngôn ngữ nguồn sang ngôn ngữ đích.
* Dịch máy học sâu (Neural Machine Translation - NMT): Được xem là tiến bộ lớn trong dịch máy, NMT sử dụng các mạng nơ-ron học sâu để học cách dịch từ ngôn ngữ nguồn sang ngôn ngữ đích. Mô hình NMT học cách biểu diễn ý nghĩa của các câu và sử dụng các mạng nơ-ron tái tạo các câu tương đương trong ngôn ngữ đích.
* Dịch máy hướng ngữ nghĩa (Example-based Machine Translation - EBMT): Phương pháp này dựa trên các ví dụ cụ thể để dịch các câu. EBMT lưu trữ các cặp câu tương đương trong cơ sở dữ liệu và sử dụng chúng để tạo ra các dự đoán dịch.
* Dịch máy cộng tác (Interactive Machine Translation - IMT): Là một phương pháp kết hợp sự hỗ trợ của con người và máy tính. Hệ thống IMT cho phép người dùng can thiệp trực tiếp vào quá trình dịch để cải thiện chất lượng dịch thuật.

## **1.2 Mô hình dịch máy sử dụng**

1.2.1 Mô hình MarianMT (transformers)

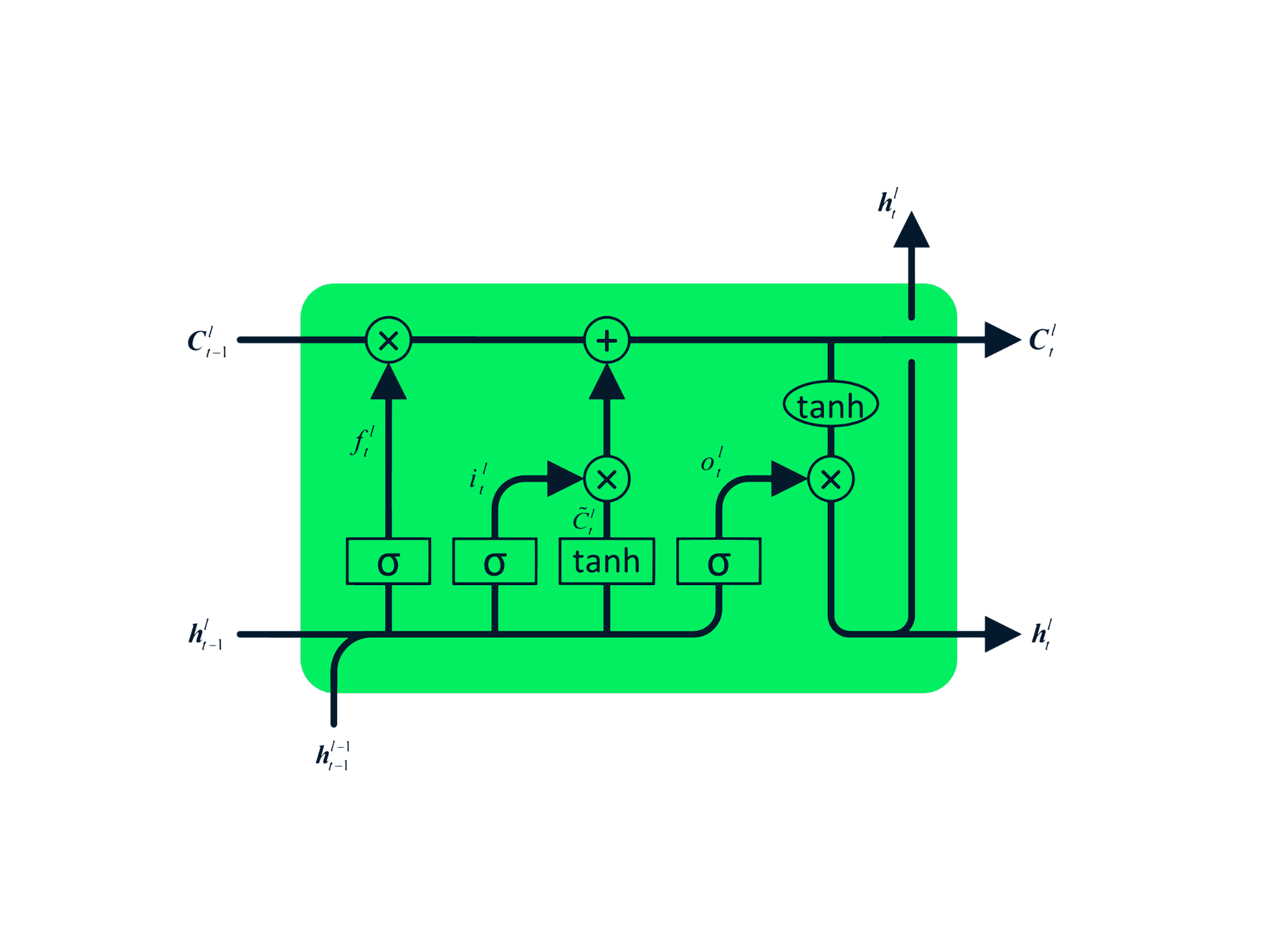
MarianMT (Multilingual Translation) là một mô hình dịch máy nền tảng từ Hugging Face, được xây dựng trên nền tảng của PyTorch và hỗ trợ nhiều ngôn ngữ. Mô hình này là một biến thể của Transformer model, một trong những mô hình dịch máy hiệu quả nhất hiện nay.

1.2.2 Translator (googletrans)

Translator thường liên quan đến việc sử dụng thư viện googletrans trong Python để thực hiện dịch máy bằng dịch vụ dịch của Google. googletrans là một thư viện Python có thể sử dụng để dịch văn bản giữa các ngôn ngữ bằng cách sử dụng API dịch Google Translate.

1.2.3 Mô hình Seq2Seq với mạng LSTM

* **Seq2Seq**
* **Khái niệm:** là một kiến trúc mạng nơ-ron sử dụng trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP) và dịch máy. Kiến trúc này được thiết kế để giải quyết các bài toán chuyển đổi từ một chuỗi đầu vào thành một chuỗi đầu ra. Các bài toán này bao gồm dịch máy, tóm tắt văn bản, phát sinh câu, và nhiều ứng dụng khác.
* **Cấu trúc:**
* Mô hình mã hóa (Encoder): Mã hóa một chuỗi đầu vào thành một vector bao gồm các thông tin cần thiết của chuỗi đó. Thông thường, một mạng LSTM (Long Short-Term Memory) hoặc GRU (Gated Recurrent Unit) được sử dụng để mã hóa, với mỗi từ trong chuỗi đầu vào được biểu diễn dưới dạng vector.
* Mô hình giải mã (Decoder): Nhận vector biểu diễn từ mô hình mã hóa và dự đoán chuỗi đầu ra từ đó. Cũng sử dụng mạng LSTM hoặc GRU, nhưng được huấn luyện để dự đoán chuỗi thay vì chỉ mã hóa.
* **LSTM**
* **Khái niệm**: LSTM là một biến thể của RNN được thiết kế để giải quyết vấn đề mất dần đạo hàm và giúp mạng "nhớ" thông tin từ xa một cách hiệu quả hơn. LSTM sử dụng các cổng (gates) để kiểm soát luồng thông tin vào và ra khỏi các ô trạng thái (cell state), giúp mạng học và quên thông tin một cách hiệu quả.
* **Cấu trúc;**



Hình 1: Sơ đồ LSTM

LSTM(Long Short-Term Memory) có cấu trúc phức tạp gồm:

* **Cổng quên ()**: Cổng này quyết định phần nào của trạng thái ô trước đó (​) sẽ được quên đi. Nó nhận đầu vào là trạng thái ẩn trước đó và đầu vào hiện tại, sau đó sử dụng hàm kích hoạt sigmoid để tạo ra một số trong khoảng từ 0 đến 1, biểu thị tỷ lệ phần nào của trạng thái ô trước đó sẽ được giữ lại.
* **Cổng đầu vào ()**: Cổng này quyết định phần nào của thông tin mới () sẽ được cập nhật vào trạng thái ô hiện tại (). Nó cũng nhận đầu vào từ trạng thái ẩn trước đó và đầu vào hiện tại, sau đó sử dụng hàm kích hoạt sigmoid để xác định phần nào của thông tin mới sẽ được cập nhật.
* **Cập nhật trạng thái ô mới (​)**: Đây là trạng thái ô tạm thời mới được tạo ra dựa trên đầu vào hiện tại và trạng thái ẩn trước đó. Trong LSTM, thông tin mới được tạo ra từ các đầu vào mới và cũ qua hàm kích hoạt tanh.
* **Cập nhật trạng thái ô (​)**: Trạng thái ô mới được tính toán bằng cách kết hợp thông tin từ cổng quên và cổng đầu vào. Nó xác định cách cập nhật trạng thái ô trước đó để tạo ra trạng thái ô mới.
* **Cổng đầu ra ()**: Cổng này quyết định phần nào của trạng thái ô mới () sẽ được chuyển sang trạng thái ẩn mới (). Nó sử dụng hàm kích hoạt sigmoid để xác định phần nào của trạng thái ô mới sẽ được chuyển đến trạng thái ẩn.
* **Trạng thái ẩn ()**: Đây là trạng thái ẩn mới được tính toán dựa trên trạng thái ô mới và cổng đầu ra. Nó đại diện cho kết quả cuối cùng của mạng LSTM tại thời điểm hiện tại.

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

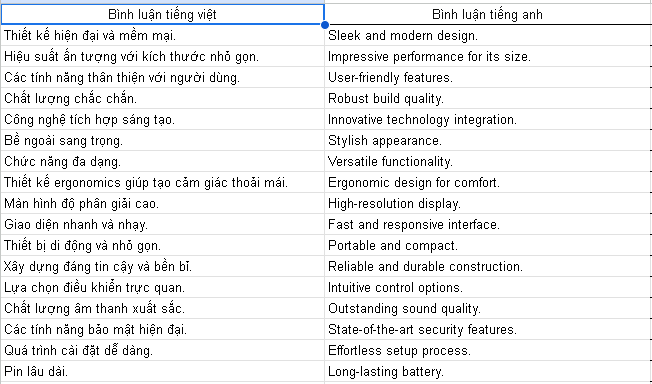
# 

# 

# **PHẦN 2: ỨNG DỤNG**

## **2.1 Mô tả tệp dữ liệu**

* Dữ liệu về bình luận sản phẩm công nghệ gồm 247 mẫu song ngữ Anh - Việt.



Hình 2: Dữ liệu bình luận sản phẩm công nghệ

## **2.2 Thiết kết mô hình**

2.2.1 Bài 1:

Bài 1.1: MarianMTModel (transformers)

* Thư viện và hàm dịch:

from transformers import MarianMTModel, MarianTokenizer

# Khởi tạo mô hình và tokenizer cho mô hình MarianMT

model\_name = "Helsinki-NLP/opus-mt-en-vi"

tokenizer = MarianTokenizer.from\_pretrained(model\_name)

model = MarianMTModel.from\_pretrained(model\_name)

def translate\_with\_marianmt(text, target\_lang='vi'):

# Tokenize câu văn vào định dạng phù hợp cho mô hình

inputs = tokenizer(text, return\_tensors="pt", truncation=True, padding=True)

# Dịch câu văn sang ngôn ngữ đích

translated = model.generate(\*\*inputs, max\_length=100, num\_beams=4, early\_stopping=True)

# Giải mã văn bản đã dịch

translated\_text = tokenizer.decode(translated[0], skip\_special\_tokens=True)

return translated\_text

* Khởi tạo mô hình và tokenizer:
* Mô hình và tokenizer được khởi tạo từ mô hình MarianMT đã được huấn luyện trước đó với tên Helsinki-NLP/opus-mt-en-vi.
* MarianTokenizer.from\_pretrained(model\_name) được sử dụng để tạo tokenizer từ mô hình đã được huấn luyện.
* MarianMTModel.from\_pretrained(model\_name) được sử dụng để tạo mô hình dịch từ ngôn ngữ Anh sang ngôn ngữ Việt.
* Hàm translate\_with\_marianmt(text, target\_lang='vi'): Đây là hàm chính để dịch văn bản. Tham số: text: Văn bản cần dịch từ tiếng Anh sang tiếng Việt. target\_lang='vi': Ngôn ngữ đích là tiếng Việt.
* Bước 1: Tokenize văn bản: tokenizer(text, return\_tensors="pt", truncation=True, padding=True): Dùng tokenizer để chuyển đổi câu văn text thành định dạng phù hợp cho mô hình, và trả về tensors của PyTorch (return\_tensors="pt"). truncation=True, padding=True đảm bảo rằng các câu văn sẽ được cắt ngắn hoặc bổ sung các token padding để có độ dài thích hợp cho mô hình.
* Bước 2: Dịch văn bản: model.generate(\*\*inputs, max\_length=100, num\_beams=4, early\_stopping=True): Sử dụng mô hình để tạo ra câu văn dịch từ tensors inputs đã được tokenize. max\_length=100: Độ dài tối đa của câu văn dịch. num\_beams=4: Số lượng "beams" sử dụng trong quá trình dịch để cải thiện chất lượng. early\_stopping=True: Dừng sớm khi có kết quả dịch hợp lý.
* Bước 3: Giải mã văn bản đã dịch: tokenizer.decode(translated[0], skip\_special\_tokens=True): Sử dụng tokenizer để chuyển đổi lại chuỗi tokens đã được dịch từ tensors thành văn bản tiếng Việt, loại bỏ các token đặc biệt như token kết thúc (<eos>).
* Trả về kết quả dịch: Hàm trả về chuỗi văn bản tiếng Việt đã được dịch từ văn bản tiếng Anh ban đầu.
* Sử dụng hàm dịch dữ liệu:

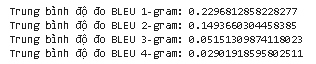
# Tao df mới

data\_transformers = pd.DataFrame({})

# Áp dụng mô hinh

data\_transformers['việt'] = data['Anh'].apply(translate\_with\_marianmt)

* Khởi tạo DataFrame mới: Bạn đã tạo DataFrame mới data\_transformers, nhưng nó không có dữ liệu từ data.
* Áp dụng mô hình dịch: Sử dụng phương thức apply của pandas để áp dụng hàm translate\_with\_marianmt cho từng mẫu văn bản trong cột 'Anh' của DataFrame data, và lưu kết quả vào cột 'việt' của data\_transformers.
* Kết quả đánh giá:



Hình 3: Kết quả MarianMTModel

Bài 1.2: Translator (googletrans)

* Thư viện và hàm dịch:

from googletrans import Translator

# hàm dịch

def translate\_text(text, target\_language='vi'):

translator = Translator()

translation = translator.translate(text, dest=target\_language)

return translation.text

* Import Translator từ thư viện googletrans.
* Định nghĩa hàm translate\_text(text, target\_language='vi') để dịch văn bản text sang ngôn ngữ target\_language (mặc định là tiếng Việt).
* Tạo một đối tượng Translator().
* Gọi phương thức translate(text, dest=target\_language) của đối tượng translator để dịch văn bản text sang ngôn ngữ target\_language.
* translation.text trả về văn bản đã được dịch.
* Sử dụng hàm dịch dữ liệu:

# Tạo df mới

data\_googletrans = pd.DataFrame({})

# Áp dụng

data\_googletrans['việt'] = data['Anh'].apply(translate\_text)

* Khởi tạo DataFrame mới: Bạn đã tạo DataFrame mới data\_googletrans, nhưng nó không có dữ liệu từ data.
* Áp dụng mô hình dịch: Sử dụng phương thức apply của pandas để áp dụng hàm translate\_test cho từng mẫu văn bản trong cột 'Anh' của DataFrame data, và lưu kết quả vào cột 'việt' của data\_googletrans.
* Kết quả đánh giá:



Hình 4: Kết quả Translator

2.2.2 Bài 2:

* Tiền xử lý dữ liệu: Tokenize và thêm các token đặc biệt cho văn bản tiếng Anh và tiếng Việt.

from pyvi import ViTokenizer

eos = '<eos>'

bos = '<bos>'

def tokenize\_vn\_text(text):

return [bos] + ViTokenizer.tokenize(text).split() + [eos]

from nltk.tokenize import word\_tokenize

def tokenize\_eng\_text(text):

return [bos] + word\_tokenize(text) + [eos]

# Tokenizer và thêm đầu cuối

data\_xaydung['Việt'] = data\_xaydung['Việt'].apply(tokenize\_vn\_text)

data\_xaydung['Anh'] = data\_xaydung['Anh'].apply(tokenize\_eng\_text)

* Tokenize tiếng Việt (tokenize\_vn\_text):
* Sử dụng ViTokenizer.tokenize(text) từ thư viện pyvi để tokenize văn bản tiếng Việt. Phương thức này sẽ phân tích câu thành các từ và trả về một chuỗi đã được tokenize.
* Sau đó, kết quả từ ViTokenizer.tokenize(text).split() được thêm vào list kết quả, với thêm token bos vào đầu và token eos vào cuối.
* Tokenize tiếng Anh (tokenize\_eng\_text):
* Sử dụng word\_tokenize(text) từ thư viện nltk để tokenize văn bản tiếng Anh. Phương thức này sẽ tách câu thành các từ dựa trên các quy tắc ngữ pháp của tiếng Anh.
* Tương tự như trên, kết quả từ word\_tokenize(text) được thêm vào list kết quả, với thêm token bos vào đầu và token eos vào cuối.
* Áp dụng cho dữ liệu:
* Sau khi định nghĩa hai hàm tokenize\_vn\_text và tokenize\_eng\_text, bạn áp dụng chúng cho dữ liệu của mình.
* data\_xaydung['Việt'] = data\_xaydung['Việt'].apply(tokenize\_vn\_text): Áp dụng hàm tokenize\_vn\_text để tokenize các câu tiếng Việt trong cột 'Việt' của DataFrame data\_xaydung.
* data\_xaydung['Anh'] = data\_xaydung['Anh'].apply(tokenize\_eng\_text): Áp dụng hàm tokenize\_eng\_text để tokenize các câu tiếng Anh trong cột 'Anh' của DataFrame data\_xaydung.
* Tokenizer: Sử dụng Tokenizer từ Keras để chuyển đổi văn bản thành chuỗi số và tạo từ điển từ vựng cho cả hai ngôn ngữ.

X\_eng = data\_xaydung['Anh'].values

X\_vn = data\_xaydung['Việt'].values

# Tokenizer và tạo từ điển cho tiếng Anh

eng\_input\_tokenize = X\_eng.copy()

eng\_tokenizer = Tokenizer(num\_words=5000)

eng\_tokenizer.fit\_on\_texts(eng\_input\_tokenize)

eng\_vocabulary = eng\_tokenizer.word\_index

eng\_size = len(eng\_vocabulary)

# Tokenizer và tạo từ điển cho tiếng Việt

vn\_input\_tokenize = X\_vn.copy()

vn\_tokenizer = Tokenizer(num\_words=5000)

vn\_tokenizer.fit\_on\_texts(vn\_input\_tokenize)

vn\_vocabulary = vn\_tokenizer.word\_index

vn\_size = len(vn\_vocabulary)

# Từ điển ngược

eng\_vocabulary\_reverse = {}

for key, value in eng\_tokenizer.word\_index.items():

eng\_vocabulary\_reverse[value] = key

eng\_vocabulary\_reverse[0] = ''

vn\_vocabulary\_reverse = {}

for key, value in vn\_tokenizer.word\_index.items():

vn\_vocabulary\_reverse[value] = key

vn\_vocabulary\_reverse[0] = ''

* Chuẩn bị dữ liệu đầu vào:
* X\_eng = data\_xaydung['Anh'].values: Lấy dữ liệu tiếng Anh từ cột 'Anh' của DataFrame data\_xaydung.
* X\_vn = data\_xaydung['Việt'].values: Lấy dữ liệu tiếng Việt từ cột 'Việt' của DataFrame data\_xaydung.
* Tokenizer và tạo từ điển cho tiếng Anh:
* eng\_input\_tokenize = X\_eng.copy(): Tạo một bản sao của dữ liệu tiếng Anh để sử dụng trong tokenizer.
* eng\_tokenizer = Tokenizer(num\_words=5000): Khởi tạo một đối tượng Tokenizer từ Keras với giới hạn tối đa là 5000 từ.
* eng\_tokenizer.fit\_on\_texts(eng\_input\_tokenize): Học từ điển từ dữ liệu tiếng Anh. Phương thức fit\_on\_texts sẽ tạo ra một bảng ánh xạ từ các từ sang các số nguyên duy nhất dựa trên tần suất xuất hiện của từng từ trong dữ liệu.
* Tokenizer và tạo từ điển cho tiếng Việt: Tương tự như tiếng Anh, các bước này được thực hiện cho dữ liệu tiếng Việt:
* vn\_input\_tokenize = X\_vn.copy()
* vn\_tokenizer = Tokenizer(num\_words=5000)
* vn\_tokenizer.fit\_on\_texts(vn\_input\_tokenize)
* Tạo từ điển ngược: Sau khi xây dựng từ điển (word\_index) cho cả tiếng Anh và tiếng Việt, các từ điển ngược được tạo để ánh xạ từ số nguyên trở lại từ vựng.
* eng\_vocabulary\_reverse: Ánh xạ từ số nguyên đến từ vựng tiếng Anh.
* vn\_vocabulary\_reverse: Ánh xạ từ số nguyên đến từ vựng tiếng Việt.
* Padding: Để chuẩn hóa chiều dài của các chuỗi số thành một độ dài nhất định.

# Chuyển đổi văn bản thành chuỗi và padding chúng

eng\_sequence = eng\_tokenizer.texts\_to\_sequences(eng\_input\_tokenize)

engmaxlen = max([len(i) for i in eng\_sequence])

eng\_sequence = pad\_sequences(eng\_sequence, maxlen = engmaxlen, padding = 'post')

vn\_sequence = vn\_tokenizer.texts\_to\_sequences(vn\_input\_tokenize)

vnmaxlen = max([len(i) for i in vn\_sequence])

vn\_sequence = pad\_sequences(vn\_sequence, maxlen = vnmaxlen, padding = 'post')

* Chuyển đổi văn bản thành chuỗi số:
* eng\_tokenizer.texts\_to\_sequences(eng\_input\_tokenize): Sử dụng tokenizer đã được huấn luyện trước đó (eng\_tokenizer) để chuyển đổi các câu văn bản tiếng Anh (eng\_input\_tokenize) thành chuỗi các số nguyên tương ứng với từng từ trong từ điển tiếng Anh.
* vn\_tokenizer.texts\_to\_sequences(vn\_input\_tokenize): Tương tự như trên, áp dụng cho văn bản tiếng Việt (vn\_input\_tokenize).
* Padding chuỗi số:
* max([len(i) for i in eng\_sequence]): Tìm độ dài lớn nhất của các chuỗi số trong danh sách eng\_sequence.
* pad\_sequences(eng\_sequence, maxlen=engmaxlen, padding='post'): Sử dụng hàm pad\_sequences từ keras.preprocessing.sequence để đệm các chuỗi số trong eng\_sequence sao cho có cùng độ dài engmaxlen. Tham số padding='post' cho biết thêm giá trị padding vào cuối của mỗi chuỗi.
* Tương tự, áp dụng cho vn\_sequence.
* Chia dữ liệu: Chia tập dữ liệu thành tập huấn luyện và tập kiểm tra (9:1).

# Chia dữ liệu thành tập huấn luyện và kiểm tra

split\_ratio = 0.9

split\_index = round(len(vn\_sequence) \* split\_ratio)

trainX = eng\_sequence[:split\_index]

trainY = vn\_sequence[:split\_index]

testX = eng\_sequence[split\_index:]

testY = vn\_sequence[split\_index:]

# Tăng kích thước tập huấn luyện lên gấp 3 lần

trainX = np.tile(trainX, (3, 1))

trainY = np.tile(trainY, (3, 1))

* Chia dữ liệu thành tập huấn luyện và kiểm tra:
* split\_ratio = 0.9: Tỷ lệ dùng để chia dữ liệu. Ở đây, 90% dữ liệu được sử dụng cho tập huấn luyện và 10% còn lại được sử dụng cho tập kiểm tra.
* split\_index = round(len(vn\_sequence) \* split\_ratio): Tính chỉ số để chia tập dữ liệu dựa trên tỷ lệ đã cho.
* trainX = eng\_sequence[:split\_index]: Lấy các chuỗi số của tiếng Anh từ đầu đến chỉ số split\_index để làm tập huấn luyện trainX.
* trainY = vn\_sequence[:split\_index]: Tương tự, lấy các chuỗi số của tiếng Việt từ đầu đến chỉ số split\_index để làm tập huấn luyện trainY.
* testX = eng\_sequence[split\_index:]: Lấy các chuỗi số của tiếng Anh từ chỉ số split\_index đến hết để làm tập kiểm tra testX.
* testY = vn\_sequence[split\_index:]: Tương tự, lấy các chuỗi số của tiếng Việt từ chỉ số split\_index đến hết để làm tập kiểm tra testY.
* Tăng kích thước tập huấn luyện:
* trainX = np.tile(trainX, (3, 1)): Sử dụng hàm np.tile để nhân bản trainX ba lần theo chiều dọc. Điều này có nghĩa là mỗi chuỗi trong trainX sẽ được sao chép ba lần.
* trainY = np.tile(trainY, (3, 1)): Tương tự cho trainY.
* Xây dựng mô hình: Sử dụng mô hình Seq2Seq với mạng LSTM cho cả Encoder và Decoder. Mô hình được biên dịch với hàm mất mát categorical\_crossentropy và tối ưu hóa RMSprop.

# Xây dựng mô hình

latent\_dim = 128

# Encoder

encoder\_inputs = Input(shape=(None,))

enc\_emb = Embedding(eng\_size+1, latent\_dim, mask\_zero=True)(encoder\_inputs)

encoder\_lstm = LSTM(latent\_dim, return\_state=True)

encoder\_outputs, state\_h, state\_c = encoder\_lstm(enc\_emb)

encoder\_states = [state\_h, state\_c]

# Decoder

decoder\_inputs = Input(shape=(None,))

dec\_emb\_layer = Embedding(vn\_size+1, latent\_dim, mask\_zero=True)

dec\_emb = dec\_emb\_layer(decoder\_inputs)

decoder\_lstm = LSTM(latent\_dim, return\_sequences=True, return\_state=True)

decoder\_outputs, \_, \_ = decoder\_lstm(dec\_emb, initial\_state=encoder\_states)

decoder\_dense = Dense(vn\_size+1, activation='softmax')

decoder\_outputs = decoder\_dense(decoder\_outputs)

# Compile mô hình

model = Model([encoder\_inputs, decoder\_inputs], decoder\_outputs)

model.compile(optimizer=RMSprop(learning\_rate=0.001), loss='categorical\_crossentropy', metrics=['accuracy'])

* Thiết lập các tham số cơ bản:
* latent\_dim = 128: Kích thước của không gian latent, tức là kích thước của vector đầu ra từ mỗi lớp LSTM.
* Encoder:
* encoder\_inputs = Input(shape=(None,)): Định nghĩa đầu vào cho encoder là một chuỗi số không định dạng (None cho phép độ dài biến đổi).
* enc\_emb = Embedding(eng\_size+1, latent\_dim, mask\_zero=True)(encoder\_inputs): Lớp nhúng (Embedding) để biểu diễn các từ vào không gian vector. eng\_size+1 là kích thước từ điển tiếng Anh cộng thêm một cho token không có từ tương ứng (để làm padding). mask\_zero=True chỉ ra rằng giá trị không phải là padding sẽ được bỏ qua.
* encoder\_lstm = LSTM(latent\_dim, return\_state=True): Lớp LSTM của encoder với latent\_dim units, và return\_state=True để trả về trạng thái ẩn và trạng thái cell cuối cùng.
* encoder\_outputs, state\_h, state\_c = encoder\_lstm(enc\_emb): Thực hiện lan truyền thuận qua LSTM và lấy ra encoder\_outputs là đầu ra của LSTM, state\_h là trạng thái ẩn cuối cùng, và state\_c là trạng thái cell cuối cùng.
* Decoder:
* decoder\_inputs = Input(shape=(None,)): Định nghĩa đầu vào cho decoder là một chuỗi số không định dạng.
* dec\_emb\_layer = Embedding(vn\_size+1, latent\_dim, mask\_zero=True): Lớp nhúng cho decoder, tương tự như encoder nhưng áp dụng cho từ điển tiếng Việt.
* dec\_emb = dec\_emb\_layer(decoder\_inputs): Biểu diễn đầu vào của decoder dưới dạng nhúng.
* decoder\_lstm = LSTM(latent\_dim, return\_sequences=True, return\_state=True): Lớp LSTM của decoder với latent\_dim units, return\_sequences=True để trả về chuỗi đầu ra và return\_state=True để trả về trạng thái ẩn và trạng thái cell.
* decoder\_outputs, \_, \_ = decoder\_lstm(dec\_emb, initial\_state=encoder\_states): Lan truyền thuận qua LSTM của decoder, sử dụng initial\_state từ encoder\_states để khởi tạo trạng thái ẩn và trạng thái cell của decoder.
* decoder\_dense = Dense(vn\_size+1, activation='softmax'): Lớp dense với softmax activation để dự đoán xác suất của từ tiếng Việt tiếp theo.
* decoder\_outputs = decoder\_dense(decoder\_outputs): Áp dụng lớp dense cho các đầu ra của LSTM để có được đầu ra dự đoán.
* Compile mô hình:
* model = Model([encoder\_inputs, decoder\_inputs], decoder\_outputs): Xác định mô hình với các đầu vào là encoder\_inputs và decoder\_inputs, đầu ra là decoder\_outputs.
* model.compile(optimizer=RMSprop(learning\_rate=0.001), loss='categorical\_crossentropy', metrics=['accuracy']): Compile mô hình với RMSprop optimizer, hàm loss là categorical\_crossentropy (phù hợp cho bài toán dự đoán xác suất), và sử dụng độ chính xác (accuracy) làm metrics để đánh giá mô hình.
* Huấn luyện mô hình: Sử dụng generator để sinh ra các batch dữ liệu huấn luyện và kiểm tra, sau đó huấn luyện mô hình.

# Hàm sinh batch

def generate\_batch(X, y, batch\_size):

while True:

for j in range(0, len(X), batch\_size):

encoder\_input\_data = []

decoder\_input\_data = []

decoder\_target\_data = []

for input\_text, target\_text in zip(X[j:j+batch\_size], y[j:j+batch\_size]):

encoder\_input\_data.append(input\_text)

decoder\_input\_data.append(target\_text[:-1]) # đầu vào decoder không bao gồm token cuối cùng

decodertargetdata = to\_categorical(target\_text[1:], num\_classes=vn\_size+1) # mục tiêu decoder không bao gồm token đầu tiên

decoder\_target\_data.append(decodertargetdata)

encoder\_input\_data = np.array(encoder\_input\_data)

decoder\_input\_data = np.array(decoder\_input\_data)

decoder\_target\_data = np.array(decoder\_target\_data)

yield([encoder\_input\_data, decoder\_input\_data], decoder\_target\_data)

# Huấn luyện mô hình

batch\_size = 128

epochs = 200

steps\_per\_epoch = max(1, len(trainX) // batch\_size)

validation\_steps = max(1, len(testX) // batch\_size)

model.fit(generate\_batch(trainX, trainY, batch\_size),

steps\_per\_epoch=steps\_per\_epoch,

epochs=epochs,

validation\_data=generate\_batch(testX, testY, batch\_size),

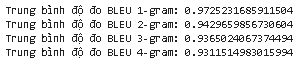
validation\_steps=validation\_steps)

1. Hàm generate\_batch:

* Mục đích: Tạo generator để sinh các batch dữ liệu cho quá trình huấn luyện mô hình. Generator này cho phép huấn luyện mô hình trên dữ liệu lớn mà không cần phải tải toàn bộ dữ liệu vào bộ nhớ.
* Tham số đầu vào:
* X: Dữ liệu đầu vào của encoder (tiếng Anh), là một mảng các chuỗi số đã được mã hóa và padding.
* y: Dữ liệu đầu ra của decoder (tiếng Việt), là một mảng các chuỗi số đã được mã hóa và padding.
* batch\_size: Kích thước của batch dữ liệu.
* Công việc thực hiện:
* Vòng lặp vô hạn (while True) để liên tục sinh batch dữ liệu.
* Với mỗi lần lặp (for j in range(0, len(X), batch\_size)), lấy các mẫu từ j đến j+batch\_size.
* encoder\_input\_data: Lưu trữ các đầu vào cho encoder (các câu tiếng Anh đã được mã hóa và padding).
* decoder\_input\_data: Lưu trữ các đầu vào cho decoder (các câu tiếng Việt đã được mã hóa và padding, loại bỏ token cuối cùng).
* decoder\_target\_data: Lưu trữ các đầu ra mong đợi từ decoder (các câu tiếng Việt đã được mã hóa và padding, loại bỏ token đầu tiên và chuyển đổi thành one-hot vector sử dụng to\_categorical).
* Trả về: Một generator sinh ra các batch dữ liệu. Mỗi lần gọi next() trên generator này sẽ trả về một batch dữ liệu gồm ([encoder\_input\_data, decoder\_input\_data], decoder\_target\_data).

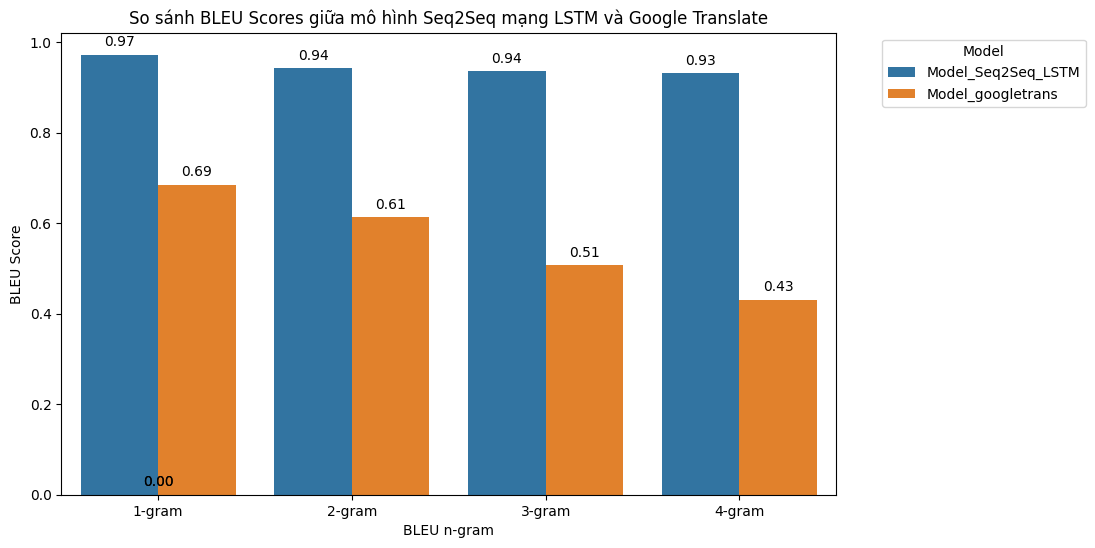
1. Huấn luyện mô hình:

* Thiết lập tham số huấn luyện:
* batch\_size = 128: Kích thước của batch dữ liệu.
* epochs = 200: Số lượng epochs (vòng lặp huấn luyện trên toàn bộ dữ liệu).
* Tính toán số bước mỗi epoch:
* steps\_per\_epoch = max(1, len(trainX) // batch\_size): Số lượng batch huấn luyện mỗi epoch, là phép chia lấy phần nguyên của tổng số mẫu huấn luyện chia cho kích thước batch.
* validation\_steps = max(1, len(testX) // batch\_size): Số lượng batch kiểm tra mỗi epoch trên tập validation, tương tự như trên.
* Huấn luyện mô hình:
* model.fit(...): Gọi phương thức fit trên mô hình để bắt đầu quá trình huấn luyện.
* generate\_batch(trainX, trainY, batch\_size): Truyền generator từ hàm generate\_batch làm dữ liệu huấn luyện.
* generate\_batch(testX, testY, batch\_size): Truyền generator từ hàm generate\_batch làm dữ liệu validation.
* steps\_per\_epoch và validation\_steps: Số lượng bước (batch) sẽ được sử dụng mỗi epoch cho huấn luyện và validation.
* Kết quả đánh giá:



Hình 5: Kết quả mô hình Seq2Seq với mạng LSTM

2.2.3 Nhận xét:



Hình 6: Biểu đồ cột so sánh BLEU

Nhận xét:

* Chung:
* Mô hình Seq2Seq mạng LSTM vượt trội hơn hẳn so với Google Translate ở tất cả các n-gram.
* Điểm số BLEU của Google Translate giảm dần khi n-gram tăng lên, trong khi điểm số của mô hình Seq2Seq mạng LSTM gần như không thay đổi nhiều.
* Ưu điểm:
* Mô hình Seq2Seq mạng LSTM dịch các mẫu trong dữ liệu có độ chính xác cao.
* Google Translate dự dịch các mẫu trong dữ liệu cũng như các mẫu mới tốt.
* Nhược điểm:
* Mô hình Seq2Seq mạng LSTM dịch các mẫu mới ngoài dữ liệu thì độ chính thấp gần như sai hết.
* Google Translate dịch chưa sát với các mẫu trong dữ liệu thường dùng các từ đồng nghĩa hoặc dịch đúng mà đảo thứ tự nên BLEU chưa cao.

## 

## 

## 

## **2.3 Kết luận**

Trong dự án này, chúng em đã thực hiện dịch máy bình luận sản phẩm công nghệ Anh-Việt sử dụng các mô hình transformers, Google Translate, Seq2Seq mạng LSTM . Việc chuẩn bị dữ liệu, huấn luyện mô hình và đánh giá hiệu suất đã được thực hiện một cách cẩn thận và kỹ lưỡng.

Kết quả cho thấy mô hình Seq2Seq mạng LSTM có hiệu suất tốt trong việc dịch bình luận sản phẩm công nghệ từ Anh sang Việt. Mặc dù mô hình có ưu điểm và hạn chế riêng, nhưng vẫn cung cấp các bản dịch có độ chính xác cao trên tập kiểm tra.

Tuy nhiên, còn nhiều cơ hội để cải thiện mô hình, bao gồm việc tối ưu hóa các tham số, mở rộng phạm vi dữ liệu và thử nghiệm các phương pháp tiên tiến hơn.

# **Tài Liệu Tham khảo**

1. <https://aws.amazon.com/vi/what-is/machine-translation/>
2. <https://viblo.asia/p/xay-dung-mo-hinh-dich-may-cho-cap-ngon-ngu-nhat-viet-3P0lPGggZox>
3. <https://huggingface.co/docs/transformers/model_doc/marian>
4. https://py-googletrans.readthedocs.io/en/latest/