TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CUỐI KỲ MÔN XỬ LÝ NGÔN NGỮ TỰ NHIÊN**

**MACHINE TRANSLATION VỚI BIRECTIONAL LSTM VÀ ATTENTION**

*Người hướng dẫn*: **TS LÊ ANH CƯỜNG**

*Người thực hiện*: **LÝ NGUYỄN QUỐC TRUNG – 520H693**

**DƯƠNG MỸ QUÂN – 520H0280**

Lớp **: 20H50304**

Khoá  **: 24**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CUỐI KỲ MÔN XỬ LÝ NGÔN NGỮ TỰ NHIÊN**

**MACHINE TRANSLATION VỚI BIRECTIONAL LSTM VÀ ATTENTION**

Người hướng dẫn: **TS LÊ ANH CƯỜNG**

Người thực hiện: **LÝ NGUYỄN QUỐC TRUNG – 520H0693**

**DƯƠNG MỸ QUÂN – 520H0280**

Lớp **: 20H50304**

Khoá  **: 24**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

LỜI CẢM ƠN

Em xin chân thành cảm ơn sự giúp đỡ nhiệt tình của các thầy cô giáo và các

bạn trong lớp đã hỗ trợ em trong suốt học kì và hoàn thành bài báo cáo này.

Em xin chân thành cảm ơn những kiến thức của thầy Lê Anh Cường và sự giúp đỡ nhiệt tình của thầy, em xin chân thành cảm ơn thầy đã luôn giúp đỡ em hoàn thành tốt bài báo cáo này. Cảm ơn các thầy đã tận tình giảng dạy, giáo dục, trang bị cho chúng em những kiến thức cần thiết để em có thể hoàn thành bài báo cáo này.

Em xin cảm ơn trường Đại học Tôn Đức Thắng đã cho tôi một môi trường giáo dục hiện đại và phát triển.Với sự chăm chỉ và cố gắng, em đã hoàn thành bài báo cáo này. Nhưng chắcchắn, báo cáo này không tránh khỏi sai sót. Em rất mong nhận được những ý kiến đóng góp của các bạn để em ngày càng hoàn thiện hơn.

Xin chân thành cảm ơn!

*TP. Hồ Chí Minh, ngày 4 tháng 5 năm 2023*

*Tác giả*

*(Ký tên và ghi rõ họ tên)*

*Lý Nguyễn Quốc Trung Dương Mỹ Quân*

**BÁO CÁO ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của thầy Lê Anh Cường. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Báo cáo cuối kỳ còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Báo cáo cuối kỳ của mình**. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 4 tháng 5 năm 2023*

*Tác giả*

*(Ký tên và ghi rõ họ tên)*

*Lý Nguyễn Quốc Trung Dương Mỹ Quân*

TÓM TẮT

Cùng với sự phát triển và tiến bộ ngày càng rõ rệt của trí tuệ nhân tạo đã giúp con người giảm bớt đi các công việc tính toán bằng trí tuệ con người, thay vào đó các cộng cụ, các trợ lý ảo đã hỗ trợ một phần nào đó trong cuộc sống của chúng ta. Ví dụ rõ rệt trong bài toán dịch ngôn ngữ của lĩnh vực xử lý ngôn ngữ, trước kia khi cần dịch 1 đoạn văn bản từ ngôn ngữ này sang ngôn ngữ khác, ta cần phải nắm rõ được bộ từ vựng khổng lồ của hai ngôn ngữ mà ta cần dịch và các kiến thức về ngữ pháp, kiến trúc câu, … Việc này làm ảnh hướng đến tốc độ dịch và lượng kiến thức khổng lồ phải học trong việc dịch ngôn ngữ.

Bài toán **Machine Translation** là một trong những bài toán quan trọng trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP), liên quan đến việc dịch một văn bản từ một ngôn ngữ sang một ngôn ngữ khác một cách tự động. Mục tiêu của bài toán này là tạo ra một hệ thống tự động dịch thuật đủ chính xác để có thể sử dụng cho các mục đích như đọc tài liệu, giao tiếp và kinh doanh giữa các quốc gia khác nhau.

Các phương pháp giải quyết bài toán này bao gồm cả các phương pháp dựa trên quy tắc và các phương pháp dựa trên dữ liệu. Các phương pháp dựa trên quy tắc sử dụng các luật ngữ pháp và cú pháp để dịch các câu, trong khi các phương pháp dựa trên dữ liệu sử dụng các mô hình học máy để học cách dịch từ các bộ dữ liệu đã cho.

Machine Translation được sử dụng rộng rãi trong các ứng dụng thực tiễn, bao gồm các công cụ dịch thuật trực tuyến, hệ thống hỗ trợ phiên dịch, công cụ đọc và tạo nội dung đa ngôn ngữ, và các ứng dụng trong các tổ chức và doanh nghiệp đa quốc gia.

Machine Translation ra đời nhằm để giải quyết được việc phải nghiên cứu và tính toán quá nhiều bằng trí tuệ con người, thì các nhà khoa học về lĩnh vực xử lý ngôn ngữ tự nhiên đã giới thiệu về các mô hình Sequence to Sequence để giải quyết các bài toán về dịch ngôn ngữ một các tự động và có tốc độ tính toán cao, một số mô hình phổ biến giải quyết các bài toán Seq2Seq là Recurrent Neural Network (RNN), Long-Short Term Memory (LSTM), … Các mô hình này đã tạo nên sự thành công và đột phá trong lĩnh vực về xử lý ngôn ngữ như Google Translation, Language Translator của Quillbot AI, …

Tuy nhiên, Machine Translation vẫn là một bài toán thách thức trong NLP do sự phức tạp và đa dạng của ngôn ngữ, từ vựng và cú pháp giữa các ngôn ngữ khác nhau. Do đó, việc phát triển các mô hình Machine Translation đòi hỏi sự kết hợp giữa kiến thức về xử lý ngôn ngữ tự nhiên, học máy và toán học.

Trong bài toán Machine Translation này, chúng ta sẽ sử dụng mô hình Seq2Seq bao gồm 2 khối Encoder và Decoder. Khối Encoder nhằm mục đích mã hoá câu văn bản ban đầu thành vector, sau đó kết quả của quá trình Encoder sẽ đi qua khối Decoder để làm đầu vào để sinh ra câu văn bản được dịch. Hai khối Encoder và Decoder sẽ sử dụng mô hình LSTM và cơ chế Attention để mã hoá và sinh chuỗi, trong đó cơ chế Attention giúp cho mô hình sinh chuỗi tốt hơn LSTM thông thường.

Bài toán sử dụng mô hình LSTM thay vì RNN gốc, điều này sẽ giúp dịch các văn bản dài tốt hơn, tránh việc Vanishing Problem trong mô hình RNN khi mã hoá các câu đầu vào quá dài, điều này làm ảnh hưởng đến kết quả của quá trình Decoder.

MỤC LỤC

[TÓM TẮT iii](#_Toc135128164)

[DANH MỤC HÌNH VẼ vii](#_Toc135128165)

[DANH MỤC CÁC CHỮ VIẾT TẮT ix](#_Toc135128166)

[CHƯƠNG 1 – TEXT GENERATION 1](#_Toc135128167)

[1.1 Text generation là gì? Và mở rộng bài toán 1](#_Toc135128168)

[1.1.1 Text generation 1](#_Toc135128169)

[1.1.2 Dữ liệu time-series 3](#_Toc135128170)

[1.2 Recurrent Neural Network (RNN) 4](#_Toc135128171)

[1.2.1 Kiến trúc mô hình RNN 4](#_Toc135128172)

[1.2.2 Các kiến trúc mô hình RNN 6](#_Toc135128173)

[1.2.3 RNN cell 7](#_Toc135128174)

[1.2.4 Backpropagation through time (BPTT) 8](#_Toc135128175)

[1.2.5 Vanishing Problem 10](#_Toc135128176)

[1.3 Long-Short Term Memory (LSTM) 11](#_Toc135128177)

[1.3.1 Kiến trúc mô hình 11](#_Toc135128178)

[1.3.2 LSTM cell 12](#_Toc135128179)

[1.3.3 Forget gate, input gate và output gate 13](#_Toc135128180)

[1.4 Ứng dụng mô hình LSTM cho bài toán Machine Translation 14](#_Toc135128181)

[1.4.1 Machine Translation 14](#_Toc135128182)

[1.4.2 Phương pháp giải quyết bài toán 15](#_Toc135128183)

[CHƯƠNG 2 – THỰC NGHIỆM 17](#_Toc135128184)

[2.1 Mô tả bài toán 17](#_Toc135128185)

[2.2 Xử lý dữ liệu đầu vào 17](#_Toc135128186)

[2.3 Chuẩn hoá dữ liệu 20](#_Toc135128187)

[2.4 Xây dựng mô hình và cải tiến mô hình 22](#_Toc135128188)

[CHƯƠNG 3 – HUẤN LUYỆN MÔ HÌNH VÀ KẾT QUẢ 24](#_Toc135128189)

[3.1 Huán luyện mô hình 24](#_Toc135128190)

[3.2 BLEU score 25](#_Toc135128191)

[3.3 Kết quả 28](#_Toc135128192)

[CHƯƠNG 4 – CẢI TIẾN MODEL 30](#_Toc135128193)

[4.1 Bidirectional LSTM 30](#_Toc135128194)

[4.1.1 Bidirectional 30](#_Toc135128195)

[4.1.2 Cài đặt 31](#_Toc135128196)

[4.2 Kết quả 32](#_Toc135128197)

[TÀI LIỆU THAM KHẢO 35](#_Toc135128198)

DANH MỤC HÌNH VẼ

[Hình 1.1 Mô hình GPT-2 và GPT-3 1](#_Toc135128199)

[Hình 1.2 Text generation bằng ChatPGT 2](#_Toc135128200)

[Hình 1.3 Dữ liệu dạng time-series 3](#_Toc135128201)

[Hình 1.4 Tính time-series trong dữ liệu text 4](#_Toc135128202)

[Hình 1.5 Kiến trúc mô hình RNN 4](#_Toc135128203)

[Hình 1.6 Mô hình FFNN 5](#_Toc135128204)

[Hình 1.7 Input của mô hình RNN 5](#_Toc135128205)

[Hình 1.8 Ứng dụng mô hình RNN trong Machine Translation 6](#_Toc135128206)

[Hình 1.9 Ứng dụng mô hình RNN trong chatbot 6](#_Toc135128207)

[Hình 1.10 Các dạng bài toán mô hình RNN 6](#_Toc135128208)

[Hình 1.11 Hidden state của mô hình RNN 8](#_Toc135128209)

[Hình 1.12 Backpropagation through time trong RNN 9](#_Toc135128210)

[Hình 1.13 Kiến trúc mô hình LSTM 11](#_Toc135128211)

[Hình 1.14 Hidden state của mô hình LSTM 13](#_Toc135128212)

[Hình 1.15 Bài toán Machine Translation 14](#_Toc135128213)

[Hình 1.16 Google Translate 15](#_Toc135128214)

[Hình 1.17 Kiến trúc chung của bài toán Machine Translation 15](#_Toc135128215)

[Hình 1.18 Kiến trúc tổng quát 16](#_Toc135128216)

[Hình 2.1 Machine Translation Problem 17](#_Toc135128217)

[Hình 2.2 Dữ liệu cho bài toán 18](#_Toc135128218)

[Hình 2.3 Một ví dụ trong bộ dataset 18](#_Toc135128219)

[Hình 2.4 Kích thước bộ train, test và validation 19](#_Toc135128220)

[Hình 2.5 Language Class 19](#_Toc135128221)

[Hình 2.6 English language 20](#_Toc135128222)

[Hình 2.7 Một ví dụ chuyển câu đầu vào thành vector 20](#_Toc135128223)

[Hình 2.8 Chuyển dữ liệu thành vector 21](#_Toc135128224)

[Hình 2.9 Kiến trúc mô hình cho bài toán dịch máy 22](#_Toc135128225)

[Hình 2.10 Tạo hai khối Encoder và Decoder 23](#_Toc135128226)

[Hình 2.11 Model Summary 24](#_Toc135128227)

[Hình 3.1 EarlyStopping và ModelCheckPoint 24](#_Toc135128228)

[Hình 3.2 Kết quả train mô hình 25](#_Toc135128229)

[Hình 3.3 Precision 1-gram 27](#_Toc135128230)

[Hình 3.4 Precision 2-gram 27](#_Toc135128231)

[Hình 3.6 Kết quả dự đoán trên bộ test 29](#_Toc135128232)

[Hình 3.7 BLEU score trên bộ train dataset 29](#_Toc135128233)

[Hình 3.8 BLEU score trên bộ test dataset 30](#_Toc135128234)

[Hình 4.1 Kiến trúc LSTM 30](#_Toc135128235)

[Hình 4.2 Kiến trúc Bidirectional LSTM 31](#_Toc135128236)

[Hình 4.3 Khối Encoder và Decoder trong BiLSTM 31](#_Toc135128237)

[Hình 4.4 Model Summary 32](#_Toc135128238)

[Hình 4.5 Kết quả dự đoán trên bộ test 33](#_Toc135128239)

[Hình 4.6 BLEU score trên bộ train dataset 33](#_Toc135128240)

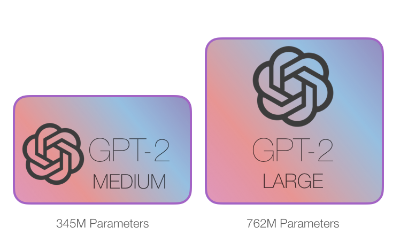
[Hình 4.7 BLEU score trên bộ train dataset 34](#_Toc135128241)

DANH MỤC CÁC CHỮ VIẾT TẮT

|  |  |
| --- | --- |
| RNN | Recurrent Neural Network |
| LSTM | Long Short-Term Memory |
| GPT | Generative Pretrain Transformer |
| AI | Artificial Intelligence |
| FFNN | Feed-Forward Neural Network |
|  |  |

CHƯƠNG 1 – TEXT GENERATION

Với sự phát triển ngày càng tiến bộ của các mô hình AI về sáng tạo nội dung, các bài toán về sinh văn bản tự động hoặc tự động trả lời đang là vấn đề được báo chí nhắc đến ngày càng nhiều. Đặc biệt là sau khi mô hình GPT-2 và GPT-3 ra đời, mô hình Generative Pretrain Transformer (GPT) đã tạo ra sự đột phá trong lĩnh vực xử lý ngôn ngữ tự nhiên trong các nghiên cứu text generation, machine translation, chatbot.

****

Hình 1.1 Mô hình GPT-2 và GPT-3

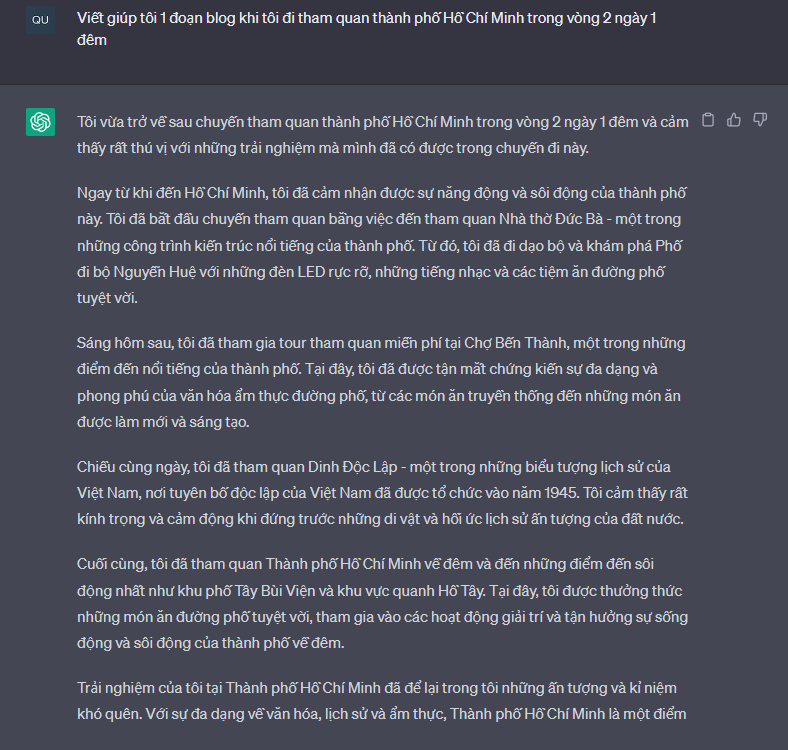
Các mô hình này đã cho thấy được sự cải tiến và tiến bộ ngày càng nhiều khi tạo ra các văn bản không thể phân biệt đâu là người và đâu là máy viết. Vậy text generation là gì? Tại sao nó lại có tầm quan trọng đến vậy?

1.1 Text generation là gì? Và mở rộng bài toán

1.1.1 Text generation

**Text generation** là một bài toán quan trọng trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP), bài toán đặt ra thách thức là tạo ra các đoạn văn bản mới và có ý nghĩa từ các nguồn dữ liệu có sẵn. Bài toán này có thể được ứng dụng trong nhiều lĩnh vực, từ tạo ra các câu trả lời tự động trong các hệ thống trò chuyện, đến viết bài tự động, tạo ra mô tả sản phẩm hoặc tóm tắt nội dung. Ví dụ, bạn cần viết một đoạn văn bản cho bài viết blog của mình, nhưng làm thế nào để có thể tạo ra một đoạn văn bản đủ dài và có ý nghĩa khi bạn chỉ có một vài thông tin hoặc một vài câu hỏi?

Dưới sự hỗ trợ của AI, bạn hoàn toàn có thể tạo một đoạn văn bản chỉ với một câu yêu cầu cho một con chatbot. Đây là một ví dụ về sinh văn bản bằng ChatGPT, đây là một con chatbot đang là chủ đề hot nhất hiện nay. Chúng ta thử yêu cầu con bot viết 1 bài viết blog chỉ với 1 câu lệnh yêu cầu



Hình 1.2 Text generation bằng ChatPGT

Ta có thể thấy dược con chatbot ChatGPT hoàn toàn có thể sinh ra một đoạn văn bản hoàn có ý nghĩa và đầy đủ thông tin mà ta đã yêu cầu. Đây là một trong những thành công lớn trong lĩnh vực xử lý ngôn ngữ tự nhiên.

Đối với các bài toán text generation sử dụng Language Model trước kia, nó cho ta thấy được các mô hình n-grams còn nhiều khuyết điểm và hạn chế để phục vụ nhu cầu xử lý ngôn ngữ của con người. Vì các mô hình Language Model ngày trước, điển hình là n-grams hoặc mô hình ANN để sinh văn bản khi tạo ra các đoạn văn bản dài, đoạn văn bản thường sẽ không có ý nghĩa khi sinh quá dài, hoặc nó chỉ phục vụ cho một vài bài toán nhỏ.

Để cải tiến được các bài toàn về text generation, các nhà khoa học nghiên cứu về lĩnh vực ngôn ngữ đã nghiên cứu và công bố các mô hình deep learning như RNN, LSTM, Transformer, BERT, … phục vụ cho bài toán xử lý dữ liệu dạng text hay còn gọi là dạng dử liệu sequence dưới nhiều tác vụ khác nhau. Trong các bài toán về ngôn ngữ hoặc text hầu hết đều phải xem chúng thuộc dạng dữ liệu sequence hoặc time series.

1.1.2 Dữ liệu time-series

Khi xử lý video ta hay gặp khái niệm FPS (frame per second) tức là bao nhiêu frame (ảnh) mỗi giây. Ví dụ 1 FPS với video 30s tức là lấy ra từ video 30 ảnh, mỗi giây một ảnh để xử lý. Mỗi một frame trong video sẽ có lượng thông tin tốt hơn khi chúng ta kết hợp với các frame trước đó thay vì chỉ tính toán trên 1 frame đơn lẻ.

A picture containing screenshot, line, green, diagram

Description automatically generated

Hình 1.3 Dữ liệu dạng time-series

Dữ liệu có thứ tự như các ảnh tách từ video ở trên được gọi là sequence, time-series data.

Trong bài toán dự đoán đột quỵ tim cho bệnh nhân bằng các dữ liệu tim mạch khám trước đó. Input là dữ liệu của những lần khám trước đó, ví dụ i1 là lần khám tháng 1, i2 là lần khám tháng 2, … i8 là lần khám tháng 8. (i1, i2, … i8) được gọi là sequence data.

A picture containing text, screenshot, font, line

Description automatically generated

Hình 1.4 Tính time-series trong dữ liệu text

Dữ liệu về text cũng là dạng dữ liệu về time-series, ta có thể thấy để dự đoán được từ going (hình 1.2), chúng ta phải dựa vào xác suất có điều kiện của các từ đứng trước nó – phương pháp n-grams. Ứng dụng được tính time-series của text, các Language Model ra đời nhằm phục vụ cho việc tạo ra văn bản dựa trên một bộ dữ liệu cho sẵn, mô hình Recurrent Neural Network là một trong những mô hình thành công nhất trong xử lý text lúc bấy giờ.

1.2 Recurrent Neural Network (RNN)

1.2.1 Kiến trúc mô hình RNN

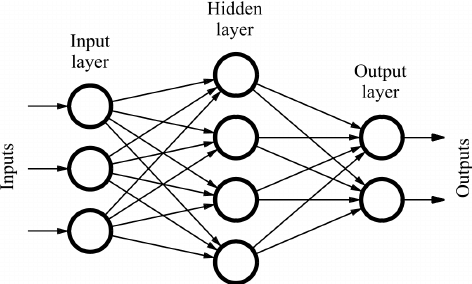
RNN là viết tắt của "Recurrent Neural Network" được công bố và giới thiệu vào năm 1986 là một loại kiến trúc mạng nơ-ron sử dụng phổ biến trong Machine Learning và Deep Learning. RNN được sử dụng hầu hết trong xử lý ngôn ngữ tự nhiên và các bài toán liên quan đến chuỗi dữ liệu.

A picture containing diagram, line

Description automatically generated

Hình 1.5 Kiến trúc mô hình RNN

Mô hình RNN ra đời nhằm mục đích giải quyết các bài toán về dự liệu dạng sequence và time-series, mô hình khắc phục được tính rời rạc và không tuần tự của mô hình Feed-Forward Neural Network (FFNN) ban đầu, vì mô hình FFNN sau khi đi qua mỗi layer, các thứ tự dữ liệu đầu vào sẽ bị xáo trộn làm mất đi tính sequence của bộ dữ liệu.



Hình 1.6 Mô hình FFNN

Khác với mô hình FFNN thông thường, dữ liệu đầu vào khi đi vào mô hình RNN sẽ đi tuần tự từ trái sang phải, điều này giúp cho mô hình vẫn giữ được tính thứ tự của dữ liệu đầu vào và kết quả của cell ở vị trí trước sẽ truyền thông tin làm đầu vào cho cell phía sau. Sau khi đi qua hết các cell, kết quả trả về của mô hình sẽ là một vector đặc trưng chứa tất cả những đặc trưng của các cell phía trước. Cái tên Recurrent (hồi quy) của mô hình cũng bắt nguồn từ đây.

A picture containing screenshot, diagram, rectangle, line

Description automatically generated

Hình 1.7 Input của mô hình RNN

Các ứng dụng phổ biến của mô hình RNN:

* Machine Translation

A picture containing text, screenshot, diagram, line

Description automatically generated

Hình 1.8 Ứng dụng mô hình RNN trong Machine Translation

* Chatbot

A picture containing text, diagram, plan, technical drawing

Description automatically generated

Hình 1.9 Ứng dụng mô hình RNN trong chatbot

1.2.2 Các kiến trúc mô hình RNN

Ứng dụng của mô hình RNN giúp giải quyết được nhiều dạng bài toán time series, dưới đây là 4 dạng bài toán phổ biến mà mô hình RNN giải quyết tốt nhất:

A picture containing screenshot, diagram, design

Description automatically generated

Hình 1.10 Các dạng bài toán mô hình RNN

* One to one: mẫu bài toán cho Neural Network (NN) và Convolutional Neural Network (CNN), 1 input và 1 output, ví dụ với CNN input là ảnh và output là ảnh được segment.
* One to many: bài toán có 1 input nhưng nhiều output, ví dụ: bài toán caption cho ảnh, input là 1 ảnh nhưng output là nhiều chữ mô tả cho ảnh đấy, dưới dạng một câu.
* Many to one: bài toán có nhiều input nhưng chỉ có 1 output, ví dụ bài toán phân loại hành động trong video, input là nhiều ảnh (frame) tách ra từ video, ouptut là hành động trong video
* Many to many: bài toán có nhiều input và nhiều output, ví dụ bài toán dịch từ tiếng anh sang tiếng việt, input là 1 câu gồm nhiều chữ: “I love Vietnam” và output cũng là 1 câu gồm nhiều chữ “Tôi yêu Việt Nam”.

1.2.3 RNN cell

Sau khi đã nắm rõ được kiến trúc chung của mô hình RNN, ta sẽ tiếp tục đi vào phần chính của mô hình RNN đó là **RNN cell**. Mỗi token khi đi vào mô hình RNN thì sẽ đi qua 1 cell hay còn gọi là hidden state của mô hình RNN. Mỗi hidden state sẽ nhận một vector embedding đầu vào của một token đầu vào, kết quả của hidden state trước đó. Ouput của mô hình sẽ là kết quả của hidden state hiện tại và output tại cell hiện tại. Dưới đây là hình ảnh tổng quát của 1 cell RNN.

A picture containing text, diagram, screenshot, plan

Description automatically generated

Hình 1.11 Hidden state của mô hình RNN

Một hidden state của RNN:

* Input: kết quả của hidden phía trước ht-1, vector embedding của token đầu vào xt.
* Output: kết quả của hidden state hiện tại ht, kết quả hidden state sẽ đi qua hàm activation function để tạo ra kết quả yt.

Hidden state ​ (trong một số tài liệu tường ký hiện ). Đây chính là bộ nhớ của mạng.  là tổng hợp thông tin của hidden state trước () cộng với input tại time step t (). Activation function ở đây chủ yếu là tanh hoặc ReLu.

Công thức tổng quát:

(1.1)

Trong đó:

* g1: là hàm activation function thường là hàm Tanh hoặc ReLu
* Whh: là ma trận weight theo hidden state
* Whx: là ma trận weight theo input đầu vào
* Bh: là trọng số bias

Output của từng time step ​ : Tại 1 block của mạng RNN có 2 đầu ra. Trong đó, ​ là tổng hợp thông tin của các state trước để tiếp tục truyền đi trong chuỗi mạng, và ta có thêm​​ là output của từng time step một. Ở đây ​​ thường là hàm softmax.

(1.2)

Trong đó:

* g2: là hàm activation function thường là hàm softmax.
* Why: là ma trận trọng số theo giá trị output đầu ra của hidden state
* ht: là kết quả đầu ra của hidden state hiện tại
* by: là trọng số bias

1.2.4 Backpropagation through time (BPTT)

Có 3 tham số ta cần phải tìm là , , . Để thực hiện gradient descent, ta cần tính:

* =
* =
* =

(Với L là loss function)

A picture containing text, diagram, circle

Description automatically generated

Hình 1.12 Backpropagation through time trong RNN

Việc đào tạo RNN không hề đơn giản, vì chúng truyền ngược gradient qua các lớp và cả theo thời gian. Do đó, trong mỗi bước thời gian, chúng ta phải tổng hợp tất cả các thông tin trước đó cho đến hiện tại, có thể tóm tắt như phương trình:

Trong phương trình này, sự đóng góp của một stage tại bước thời gian k vào độ dốc của toàn bộ hàm mất mát L, tại bước thời gian t=T được tính toán

1.2.5 Vanishing Problem

Thuật ngữ này tiến nghĩa là đạo hàm sẽ tiến về về 0 nhanh theo cấp số nhân, điều này gây khó khăn cho việc tìm hiểu một số phụ thuộc trong thời gian dài. Vấn đề này được gọi là vanishing gradient.

A black text on a white background

Description automatically generated with medium confidence

Sau khi RNN đưa ra vectơ dự đoán h(k), chúng tôi tính toán lỗi dự đoán E(k) và sử dụng thuật toán BPTT để tính toán gradient.

Gradient được sử dụng để cập nhật các tham số mô hình bằng cách:

Giả sử chúng ta có T bước thời gian để training, độ dốc của lỗi trên bước thời gian k được đưa ra bởi:

(1.3)

Trong đó được tính bằng:

Đạo hàm của

(1.4)

Từ (1.3) và (1.4) ta có:

Khi đó bạn sẽ thấy rõ rằng biểu thức cuối cùng có xu hướng biến mất khi k lớn, điều này là do đạo hàm của hàm kích hoạt (tanh hoặc sigmoid) nhỏ hơn 1.

Trong một bước k nào đó:

A picture containing font, white, text, graphics

Description automatically generatedA picture containing font, white, text, symbol

Description automatically generated

1.3 Long-Short Term Memory (LSTM)

1.3.1 Kiến trúc mô hình

Mô hình Long-Short Term Memory (LSTM) được giới thiệu vào năm 1997 là mô hình cải tiến của mô hình RNN truyền thống, sự thành công của mô hình LSTM là khác phục được vấn đề vanishing gradient trong mô hình RNN mắc phải. Về kiến trúc chung thì mô hình LSTM vẫn dự vào kiến trúc của của mô hình RNN, nhưng điểm khác biệt giữa hai mô hình là trạng thái hidden của mô hình LSTM sẽ thêm 1 trạng thái là gọi là state c.

A picture containing diagram, line, plot, screenshot

Description automatically generated

Hình 1.13 Kiến trúc mô hình LSTM

Một mạng LSTM có ba cổng cập nhật và kiểm soát trạng thái cell, là forget gate, input gate và output gate. Các cổng sử dụng các hàm kích hoạt tiếp tuyến hyperbol và sigmoid

1.3.2 LSTM cell

Kiến trúc LSTM cell có phần khác với mô hình RNN là ta sẽ có 3 cổng để quyết định lượng dữ liệu sẽ được đi qua hoặc giữ lại thông qua 3 cổng: forget gate, input gate và output gate.

Trong mô hình LSTM, state c (hay còn gọi là cell state) là một trong ba đầu ra của lớp LSTM. Nó biểu thị trạng thái bên trong của mô hình và được cập nhật và truyền qua các lần tính toán.

Trong quá trình lan truyền thuận (forward pass) của LSTM, state c được cập nhật dựa trên đầu vào hiện tại và trạng thái trước đó của state c và các đầu ra khác của LSTM. Đầu ra state c được tính bằng cách áp dụng một số hàm kích hoạt và bộ lọc cho các giá trị đầu vào và đầu ra trước đó của LSTM. Điều này giúp cho state c có khả năng lưu trữ thông tin lâu dài và tránh tình trạng vanishing gradient (điều này xảy ra khi gradient trở nên quá nhỏ để được đưa trở lại các lớp trước đó trong mô hình và gây khó khăn trong việc huấn luyện mô hình).

Trong quá trình lan truyền ngược (backward pass) trong quá trình huấn luyện, gradient sẽ được lan truyền qua state c, để cập nhật các trọng số của mô hình.

A picture containing diagram, line, screenshot, plan

Description automatically generated

Hình 1.14 Hidden state của mô hình LSTM

1.3.3 Forget gate, input gate và output gate

Sự thành công của mô hình LSTM trong việc tránh vashing problem là có 3 cổng quyết định lượng tham số đi qua thông qua 3 cổng forget gate, input gate và output gate.

* Forget gate quyết định xem cần lấy bao nhiêu từ cell state trước
* Input gate sẽ quyết định lấy bao nhiêu từ input của state và hidden layer của layer trước
* Output gate quyết định xem cần lấy bao nhiêu từ cell state để trở thành output của hidden state. Ngoài ra ​ cũng được dùng để tính ra output cho state t.

Một cell của LSTM có dạng:

A picture containing text, font, handwriting, white

Description automatically generated

Hoặc:

A picture containing font, text, white, calligraphy

Description automatically generated

A picture containing text, font, diagram, white

Description automatically generated

Mô hình LSTM giúp tránh vanishing problem vì:

* Lưu ý rằng gradient chứa vectơ kích hoạt của forget gate, cho phép mạng kiểm soát tốt hơn các giá trị gradient, tại mỗi bước thời gian, bằng cách sử dụng các cập nhật tham số phù hợp của forget gate. Sự hiện diện của các kích hoạt của forget gate cho phép LSTM quyết định, tại mỗi bước, rằng không nên bỏ qua một số thông tin quan trọng và cập nhật các tham số của mô hình cho phù hợp.
* Chính sự hiện diện của vectơ kích hoạt của forget gate trong thuật ngữ gradient cùng với cấu trúc cộng cho phép LSTM tìm thấy bản cập nhật tham số như vậy tại bất kỳ thời điểm nào và điều này mang lại

A picture containing font, white, text, graphics

Description automatically generated

1.4 Ứng dụng mô hình LSTM cho bài toán Machine Translation

1.4.1 Machine Translation

Machine Translation hay còn gọi là dịch máy, đây là một bài toán khá phổ biến trong lĩnh vực xử lý ngôn ngữ tự nhiên với mục đích là dịch tự động văn bản hoặc lời nói từ ngôn ngữ tự nhiên này sang ngôn ngữ tự nhiên khác. Machine Translation sử dụng kết hợp nhiều ý tưởng và các kỹ thuật với nhau từ ngôn ngữ học, khoa học máy tính, xác suất thống kê. Mục tiêu của machine translation là tạo ra được một hệ thống cho phép tạo ra các bản dịch chính khác giữa các ngôn ngữ tự nhiên của con người.

A group of colorful speech bubbles with different languages

Description automatically generated with low confidence

Hình 1.15 Bài toán Machine Translation

Các hệ thống machine translation thành công có thể kể dến là: Google Translate, Bing Translator, … đã đạt được chất lượng bản dịch tốt. thích hợp trên mọi nền tảng khác nhau và có thể dịch trên hàng trăm ngôn ngữ khác nhau.

A screenshot of a computer

Description automatically generated with medium confidence

Hình 1.16 Google Translate

Từ đấy ta có thể thấy được **Input/Ouput** của bài toàn này là gì:

* **Input**: Văn bản đầu vào của ngôn ngữ nguồn.

VD: Câu đầu vào là tiếng Anh: “I am learning NLP”

* **Output**: Văn bản đầu ra của ngôn ngữ cần dịch

VD: Câu đầu ra là tiếng Việt: “Tôi đang học NLP”

1.4.2 Phương pháp giải quyết bài toán

Đối với bài toán Machine Translation, đây là bài toán thuộc dạng Many to Many (Sequence to Sequence), để thực hiện được bài toán này, ta cần phải thiết kế 2 khối Encoder và Decoder.

* Khối Encoder sẽ có chức năng mã hoá thông tin đầu vào thành vector để làm tham số đầu vào cho khối Decoder.
* Khối Decoder sẽ có chức năng nhận thông tin từ khối Encoder và dự đoán ra chuỗi văn bản dịch trên ngôn ngữ khác.



Hình 1.17 Kiến trúc chung của bài toán Machine Translation

Kiến trúc tổng quát của mô hình Machine Translation sử dụng LSTM:

A picture containing text, screenshot, font, plot

Description automatically generated

Hình 1.18 Kiến trúc tổng quát

Khối **Encoder** sẽ bao gồm:

* Input Layer: Các từ vựng của bộ dữ liệu nguồn sẽ đi vào khối Encoder
* Embedding Layer: Đây là lớp dùng để biểu diễn câu đầu vào thành word2vec, mỗi từ vựng trong câu sẽ mã hoá thành 1 vector đặc trưng.
* LSTM Layer: Đây là lớp sẽ tổng hợp toàn bộ thông tin của các tự vựng trong câu dữ liệu đầu vào và trả về vector đặc trưng của toàn bộ thông tin của câu dữ liệu đầu vào. Kết quả đầu ra của khối Encoder sẽ là đầu vào cho khối Decoder còn gọi là h0.

Khối **Decoder** sẽ tương tự như khối Encoder:

* Input Layer: Cell đầu tiên của khối Decoder sẽ là kí tự <START> nó đại diện cho bắt đầu chuỗi, sau khi cell đầu tiên đã predict ra từ vựng thì tự vựng này sẽ làm input cho cell phía sau.
* Embedding Layer: Đây là lớp dùng để biểu diễn câu đầu vào thành word2vec, mỗi từ vựng trong câu sẽ mã hoá thành 1 vector đặc trưng.
* LSTM Layer: Đây là lớp sẽ có chức năng dự đoán chuỗi output đầu ra, với tham số đầu vào sẽ là đặc trưng cảu câu nguồn.
* Dense Layer: Đây là một lớp neural network đơn giản dùng để dự đoán tự vừng cần dịch.

Các bước thực hiện:

* **Bước 1**: Chuẩn bị dữ liệu, xây dựng bộ tokenizer cho lần lượt ngôn ngữ tiếng anh và tiếng việt.
* **Bước 2**: Chuyển văn bản thành các vector biểu diễn dược vào bộ tokenizer đã tạo tưng ứng với từng ngôn ngữ.
* **Bước 3**: Xây dựng mô hình với 2 khối Encoder và Decoder bằng LSTM
* **Bước 4**: Huấn luyện mô hình
* **Bước 5**: Tạo ra 2 mô hình dự đoán Encoder và Decoder để dịch các câu trên bộ dữ liệu test.
* **Bước 6**: Đánh giá kết quả bằng BLEU score

CHƯƠNG 2 – THỰC NGHIỆM

2.1 Mô tả bài toán

Như đã trình bài ở phần 1.4, bài toán Machine Translation là mục tiêu chính là dịch một đoạn văn bản từ ngôn ngữ nguồn sang ngôn ngữ đích.

A screenshot of a computer

Description automatically generated with medium confidence

Hình 2.1 Machine Translation Problem

Về các bước thực hiện:

* Xử lý input đầu vào
* Chuẩn hoá dữ liệu để đưa vào model
* Xây dựng model
* Cải thiện và tối ưu model

2.2 Xử lý dữ liệu đầu vào

Trong bài toán này, bộ dataset mà chúng ta sẽ dùng là bộ data **machine translation english to vietnamese** trên Kaggle với 146148 cặp câu english-vietnamese. Trong bộ dữ liệu, ngôn ngữ tiếng Việt đã được word segmentation, điều này giúp ta giảm bớt đi công việc word segmentation, nếu dữ liệu chưa được word segmentation, ta có thể dùng thư viện underthesea – đây là thư viện hỗ trợ xử lý tiếng Việt nổi tiếng.



A screenshot of a computer

Description automatically generated with low confidence

Hình 2.2 Dữ liệu cho bài toán

A screenshot of a computer

Description automatically generated with low confidence

Hình 2.3 Một ví dụ trong bộ dataset

Đối với các câu ngôn ngữ đích, ta sẽ thêm 2 kí hiệu <START> và <END> vào đầu vào cuối câu, điều này sẽ giúp cho khối Decoder của mô hình sẽ dự đoán được các chuỗi đích cần dịch và kí tự <END> dùng để biết là kết thúc chuỗi.

A screen shot of a computer code

Description automatically generated with low confidence

Sau khi thực hiện các bước tiền xử lý dữ liệu, ta sẽ tách bộ dữ liệu thành 3 cặp training, testing và validation với tỉ lệ là 8:2 training và testing, tập validation sẽ là 2:8 đối với bộ dữ liệu training.

A screen shot of a computer code

Description automatically generated with low confidence

Hình 2.4 Kích thước bộ train, test và validation

Để dễ dàng quan sát và tránh nhầm lẫn trong quá trình code, chúng ta sẽ tạo ra một lớp Language để biểu diễn những đặc trưng cho từng bộ ngôn ngữ: từ vựng, kích thước từ vựng, câu dài nhất, word2idx, idx2word, …

A screen shot of a computer program

Description automatically generated with low confidence

Hình 2.5 Language Class

A screen shot of a computer code

Description automatically generated with low confidence

Hình 2.6 English language

2.3 Chuẩn hoá dữ liệu

Một bước quan trọng đó là quá trình chuẩn hoá dữ liệu để đưa vào mô hình để huán luyện, ta sẽ chuyển các chuỗi đầu vào thành các các vector bằng lớp Tokenizer của thư viện Keras, vì trong class Language chúng ta đã tạo tokenizer cho từng bộ ngôn ngữ, sau đó sử dụng phương thức texts\_to\_sequences để chuyển chuỗi đầu vào về thành vector.

A screen shot of a computer code

Description automatically generated with low confidence

A screenshot of a computer code

Description automatically generated with low confidence

Hình 2.7 Một ví dụ chuyển câu đầu vào thành vector

Sau khi thực hiện chuyển các chuỗi đầu vào thành vector, bước tiếp theo ta cần chèn các <PADDING> tương ứng với giá trị 0 để biến các vector đầu vào cho chúng có cùng kích thước. Kích thước chung mà ta quy ước ở đây sẽ là câu có chiều dài dài nhất.

A close-up of a computer code

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with medium confidence

Hình 2.8 Chuyển dữ liệu thành vector

Sau khi tiền xử lý bộ dữ liệu, ta sẽ có 3 cặp dữ liệu chính để huấn luyện mô hinh là:

* train\_context\_inputs: input sẽ đi vào khối Encoder
* train\_target\_inputs: input sẽ đi vào khối Decoder
* train\_target\_outputs: label của các từ vựng để huấn luyện mô hình. Tập train\_target\_outputs thực chất là tập train\_target\_inputs nhưng ta chỉ lấy từ [1:} nghĩa là ta sẽ bỏ đi kí tự <START> để giúp cho khối Decoder dự đoán nhiễu.

2.4 Xây dựng mô hình và cải tiến mô hình

Mô hình sẽ bao gồm 2 khối chính là Encoder và Decoder, ngoài ra để tối ưu hoá được kết quả học của mô hình, chúng ta sẽ sử dụng thêm cơ chế Attention vào mô hình. Quá trình tính toán Attention sẽ ở giai đoạn Encoder, kết quả trả về các hidden state của khối Encoder sẽ làm Key cho kết quả trả về của khối Decoder.

A picture containing text, line, diagram, parallel

Description automatically generated

Hình 2.9 Kiến trúc mô hình cho bài toán dịch máy

Cơ chế Attention là cơ chế giúp mô hình predict tốt hơn nhờ vào phân phối xác suất của Attention giúp cho 1 cell của state Decoder biết là sẽ tập trung vào từ nào là chủ yếu. Trong bài toán trên, query sẽ là output của lớp LSTM khối Decoder và key sẽ là output của lớp LSTM khối encoder. Sau khi thực hiện tính trọng số Attention, kết quả attention score sẽ được nối với output của lớp LSTM khối Decoder.

A screenshot of a computer program

Description automatically generated with low confidence

Hình 2.10 Tạo hai khối Encoder và Decoder

Mô hình gồm 2 khối Encoder và Decoder:

* Khối Encoder: 1 lớp Embedding và 1 lớp LSTM.
* Embedding layer sẽ có chuyển 1 token đầu vào thành vector 128 chiều.
* LSTM layer: 256 hiddent units
* Khối Decoder:
* Embedding layer sẽ có chuyển 1 token đầu vào thành vector 128 chiều.
* LSTM layer: 256 hiddent units
* Dense layer: 40934 label đầu ra, tương ứng với số lượng từ vựng.

Mô hình có khoảng 34,629,734 tham số cần học

A screenshot of a computer

Description automatically generated

Hình 2.11 Model Summary

CHƯƠNG 3 – HUẤN LUYỆN MÔ HÌNH VÀ KẾT QUẢ

3.1 Huán luyện mô hình

Để huấn luyện mô hình tốt hơn, ta sẽ sử dụng 2 phương pháp EarlyStopping và ModelCheckPoint của keras hỗ trợ việc model bị overfitting và sập kernel khi train mô hình quá lớn.

A picture containing text, font, line, screenshot

Description automatically generated

Hình 3.1 EarlyStopping và ModelCheckPoint

Mô hình được train 150 epochs, batch\_size = 128, hàm opimizer là hàm Adam, độ loss được tính bằng Sparse Categorical Crossentropy. Mô hình train trong khoản 9 tiếng, đây là kết quả train của 5 epochs cuối:

A picture containing text, screenshot, font, document

Description automatically generated

Hình 3.2 Kết quả train mô hình

3.2 BLEU score

BLEU (Bilingual Evaluation Understanding) là một thuật toán để đánh giá chất lượng văn bản đã được dịch từ ngôn ngữ tự nhiên này sang ngôn ngữ tự nhiên khác. Chất lượng được coi là sự tương ứng giữa đầu ra của mô hình và đầu ra của con người: "bản dịch càng gần với bản dịch của con người thì càng tốt". BLEU là một trong những cách tính mối tương quan giữa dự đoán của mô hình và human target sentence là cao hay thấp và đây cũng là một thuật toán một trong những cách tính dễ hiểu, đơn giản và phổ biến nhất trong đánh giá bài toán Seq2Seq.

BLUE score sẽ có giá trị trong khoảng từ 0 đến 1. Thông thường các mô hình Seq2Seq sẽ có giá trị BLUE score trong khoảng 0.6 đến 0.7 đều được xem là tốt, vì việc dịch 1 đoạn văn bản sẽ có nhiều cách dịch khác nhau, nếu mô hình tính giá trị blue score bằng 1 thì mô hình rất có thể đã bị overfitting.

Hai phương pháp tạo nên thuật toán BLUE score là N-grams và Precistion.

1. ***N-grams***

N-gram là tần suất xuất hiện của n kí tự (hoặc từ) liên tiếp nhau có trong tập corpus. Một số mô hình n-grams phổ biến, ví dụ với câu đầu vào "The ball is blue":

* 1-gram (unigram): "The", "ball", "is", "blue".
* 2-gram (bigram): "The ball", "ball is", "is blue".
* 3-gram (trigram): "The ball is", "ball is blue".
* 4-gram: "The ball is blue".

1. ***Precision***

Metric đo bằng số lượng từ ở trong predicted sentence mà cũng xuất hiện ở trong target sentence.

Ví dụ, ta có câu đầu vào:

* Target Sentence: He eats an apple.
* Predicted Sentence: He ate an apple.

Khi đó Precision sẽ được tính như sau:

Precision = 3 / 4

1. ***BLEU score***

Đây là một chỉ số đo lường số lượng các từ liên tiếp khớp giữa chú thích được dự đoán và chú thích đúng. Nó so sánh n-gam độ dài khác nhau từ 1 đến 4 để thực hiện điều này.

Giả sử chúng ta có 1 mô hình NLP mà sinh predicted sentence như dưới đây. Để đơn giản, chúng ta chỉ lấy 1 Target Sentence, dù trên thực tế có rất nhiều Target Sentence mang nghĩa như nhau.

* Target Sentence: The guard arrived late because it was raining.
* Predicted Sentence: The guard arrived late because of the rain.

Bước đầu tiên ta sẽ tính Precision Score cho 1-gram đến 4-gram

**Precision 1-gram**

A picture containing text, font, screenshot, line

Description automatically generated

Hình 3.3 Precision 1-gram

Do đó, Precision 1-gram (p1​) = 5 / 8

**Precision 2-gram**

A picture containing text, font, screenshot, line

Description automatically generated

Hình 3.4 Precision 2-gram

Do đó, Precision 2-gram (p2​) = 4 / 7

Tương tự với 3-gram và 4-gram

**Precision 3-gram**

Tương tự, Precision 3-gram (p3​) = 3 / 6

**Precision 4-gram**

Và cuối cùng, Precision 4-gram (p4​) = 2 / 5

**Geometric Average Precision Scores**

Tiếp theo, chúng ta kết hợp các Precision Score sử dụng công thức dưới đây. Cái này có thể tính toán cho các N khác nhau và các giá trị weight khác nhau. Cụ thể, ta cho N = 4 và uniform weights wn​ = N / 4.

**Brevity Penalty**

Bước thứ 3 là tính "Brevity Penalty".

Nếu để ý cách Precision được tính toán, có thể thấy rằng output có predicted sentence chứ các từ đơn như "The" hay "late". Khi đó, 1-gram Precision sẽ là 1/1 = 1, perfect score. Điều này là sai vì nó khuyến khích model sinh đầu ra ngắn hơn và điểm cao hơn.

Do đó, Brevity Penalty penalize những câu trả lời quá ngắn.

Trong đó:

* c là predicted length = số lượng từ có trong predicted sentence
* r là target length = số lượng từ có trong target sentence

Công thức trên đảm bảo rằng Brevity Penalty không thể lớn hơn 1, cho dù predicted sentence ngắn hơn nhiều so với target sentence. Và nếu model predict rất ít từ, giá trị này sẽ càng nhỏ.

**BLEU Score**

Cuối cùng, để tính Bleu Score, ta nhân Brevity Penalty với Geometric Average of Precision Scores.

Bleu Score có thể tính toán cho nhiều giá trị N khác nhau. Cụ thể trong trường hợp này, ta xét N = 4.

* BLEU-1 sử dụng unigram Precision Score.
* BLEU-2 sử dụng geometric average of unigram and bigram Precision Score.
* BLEU-3 sử dụng geometric average of unigram, bigram, and trigram Precision Score.
* Và cứ tiếp tục như thế cho đến BLEU-N.

Với 1 số nguồn khác, hàm Bleu Score có thể được viết lại như sau:

3.3 Kết quả

Kết quả của mô hình:

A screenshot of a computer

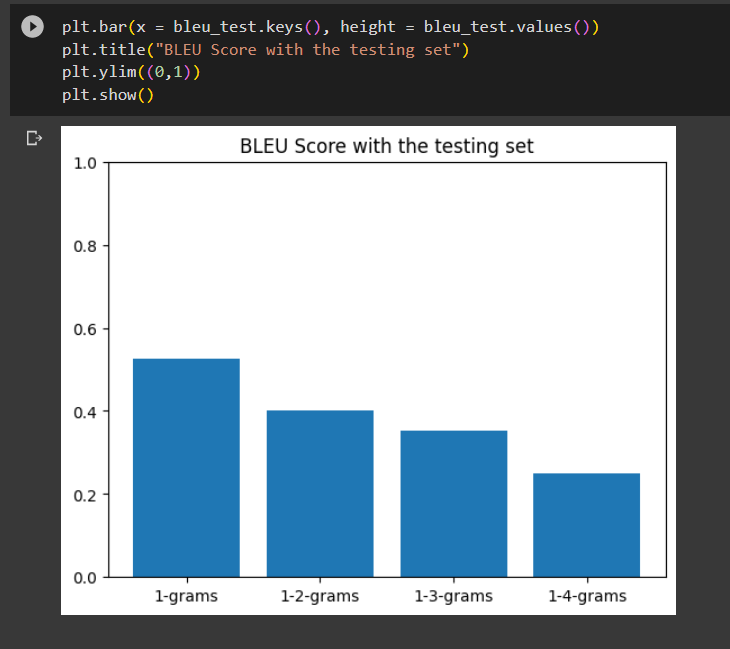
Description automatically generated with medium confidence

Hình 3.6 Kết quả dự đoán trên bộ test

Kết quả BLEU score trên bên training dataset



Hình 3.7 BLEU score trên bộ train dataset



Hình 3.8 BLEU score trên bộ test dataset

CHƯƠNG 4 – CẢI TIẾN MODEL

4.1 Bidirectional LSTM

4.1.1 Bidirectional

Trong khối Encoder của mô hình phía trên, chúng ta chỉ sự dụng 1 lớp LSTM đơn thuần để mã hoá thông tin đầu vào theo 1 chiều từ trái qua phải.

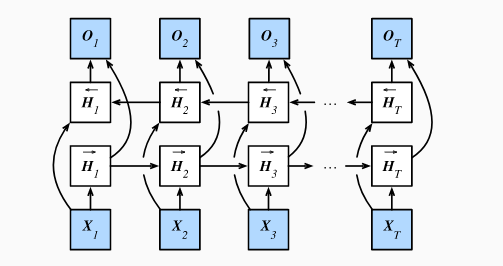
A diagram of a flowchart

Description automatically generated with low confidence

Hình 4.1 Kiến trúc LSTM

Khi đó, mô hình chỉ học được thông tin từ trái qua phải điều này sẽ làm cho mô hình học không được tốt, như kết quả phía trên. Nếu chúng ta mã hoá thông tin trong khối Encoder theo 2 chiều thì kết quả sẽ như thế nào?

Thay vì chỉ vận hành một LSTM chạy từ kí tự đầu đến cuối, ta sẽ khởi tạo một LSTM nữa chạy từ kí tự cuối lên đầu. Bidirectional Long Short-Term Memory sẽ thêm một tầng ẩn cho phép xử lý dữ liệu theo chiều ngược lại một cách linh hoạt hơn so với LSTM truyền thống. Mô tả cấu trúc của mạng nơ-ron hồi tiếp hai chiều với một tầng ẩn.



Hình 4.2 Kiến trúc Bidirectional LSTM

Kết quả trả về của khối Encoder sẽ là kết quả mã hoá câu đầu vào theo chiều từ trái qua phải và ngược lại. Điều này sẽ làm cho mô hình có nhiều tham số để học, nhưng kết quả mang lại sẽ tốt hơn mô hình cũ.

4.1.2 Cài đặt

A screenshot of a computer program

Description automatically generated with low confidence

Hình 4.3 Khối Encoder và Decoder trong BiLSTM

A screenshot of a computer program

Description automatically generated with medium confidence

Hình 4.4 Model Summary

Số lượng epochs và batch\_size của mô hình sẽ tương tự như mô hình trước.

4.2 Kết quả

Kết quả sau khi cải tiến:

A screenshot of a computer

Description automatically generated with medium confidence

Hình 4.5 Kết quả dự đoán trên bộ test

A screen shot of a graph

Description automatically generated with low confidence

Hình 4.6 BLEU score trên bộ train dataset

A screen shot of a graph

Description automatically generated with medium confidence

Hình 4.7 BLEU score trên bộ train dataset

TÀI LIỆU THAM KHẢO

**Tiếng Việt**

[1] [9.4. Mạng Nơ-ron Hồi tiếp Hai chiều — Đắm mình vào Học Sâu 0.14.4 documentation (aivivn.com)](https://d2l.aivivn.com/chapter_recurrent-modern/bi-rnn_vn.html)

[2] [Tìm hiểu về BLEU và WER - Metric cho 1 số tác vụ trong NLP (viblo.asia)](https://viblo.asia/p/tim-hieu-ve-bleu-va-wer-metric-cho-1-so-tac-vu-trong-nlp-Eb85oA16Z2G)

[3] [Machine Translate với Attention trong Deep Learning (viblo.asia)](https://viblo.asia/p/machine-translate-voi-attention-trong-deep-learning-Az45bY7zlxY)

**Tiếng Anh**

[1] [Neural machine translation with attention  |  Text  |  TensorFlow](https://www.tensorflow.org/text/tutorials/nmt_with_attention)

[2] [Foundations of NLP Explained — Bleu Score and WER Metrics | by Ketan Doshi | Towards Data Science](https://towardsdatascience.com/foundations-of-nlp-explained-bleu-score-and-wer-metrics-1a5ba06d812b)

[3] [[1706.03762] Attention Is All You Need (arxiv.org)](https://arxiv.org/abs/1706.03762)

[4] [How LSTMs solve the problem of Vanishing Gradients? | by Parveen Khurana | Medium](https://prvnk10.medium.com/how-lstms-solve-the-problem-of-vanishing-gradients-ea88f08c78ca)