

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN  
KHOA CÔNG NGHỆ THÔNG TIN  
BỘ MÔN CÔNG NGHỆ PHẦN MỀM**

**TRƯỜNG PHẠM NHẬT TIẾN – NGUYỄN MINH TRÍ**

**XÂY DỰNG MÔ HÌNH DỊCH MÁY TỪ  
TIẾNG ANH SANG TIẾNG VIỆT**

**KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT**

**TP. HCM, NĂM 2020**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN KHOA CÔNG NGHỆ  
THÔNG TIN BỘ MÔN CÔNG NGHỆ PHẦN MỀM**

**TRƯỜNG PHẠM NHẬT TIẾN – 1612689  
NGUYỄN MINH TRÍ – 1612726**

# **XÂY DỰNG MÔ HÌNH DỊCH MÁY TỪ TIẾNG ANH SANG TIẾNG VIỆT**

**KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT**

**GIÁO VIÊN HƯỚNG DẪN**

**TS. NGÔ HUY BIÊN**

**KHÓA 2016 - 2020**

## This image shows a full page of a document template designed for handwritten notes or essays. It features approximately 28 evenly spaced, thin grey horizontal lines extending across the entire width of the page. The margins are consistent on all sides, providing ample space for writing. There are no pre-printed questions, headings, or other markings on the page.

Giáo viên hướng dẫn

[Kí tên và ghi rõ họ tên]

**NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN**

Khóa luận đáp ứng yêu cầu của Khóa luận cử nhân CNTT.

TpHCM, ngày ..... tháng ..... năm .....

Giáo viên phản biện

[Ký tên và ghi rõ họ tên]

## LỜI CẢM ƠN

Tri ân thầy - Tiến sĩ Ngô Huy Biên, người đã luôn trực tiếp hướng dẫn, định hướng cho hướng đi của luận văn, góp ý giúp đỡ nhiệt tình chúng em trong các vấn đề về kiến thức, nội dung, cách thức trình bày đồng thời luôn tạo điều kiện thoải mái nhất để chúng em có thể hoàn thành khóa luận, chúng em xin gửi đến thầy lời cảm ơn chân thành và sâu sắc nhất.

Chúng em xin gửi lời cảm ơn đến quý Thầy Cô trong khoa công nghệ thông tin của trường đại học Khoa Học Tự Nhiên đã tận tình giảng dạy nâng bước chúng em trong suốt gần 4 năm học vừa qua. Em xin chân thành cảm ơn Khoa Công Nghệ Thông Tin, trường Đại học Khoa Học Tự Nhiên, Đại học Quốc gia Tp. Hồ Chí Minh đã tạo điều kiện thuận lợi cho chúng em trong quá trình học tập và thực hiện đề tài tốt nghiệp. Đồng thời chúng em cũng không quên gửi những lời cảm ơn chân thành đến những người thân trong gia đình và bạn bè đã giúp đỡ chúng em trong quá trình thực hiện luận văn.

Do trình độ nghiên cứu và thời gian có hạn, chúng em đã cố gắng hết sức nhưng chắc chắn không tránh khỏi những thiếu sót và hạn chế. Rất mong nhận được sự góp ý và chỉ dẫn của quý Thầy Cô.

Cuối cùng, chúng em xin trân trọng cảm ơn và chúc sức khỏe quý Thầy Cô!

TpHCM, ngày . . . tháng . . . năm 2020

Sinh viên

## ĐỀ CƯƠNG CHI TIẾT

<b>Tên Đề Tài:</b> Xây dựng mô hình dịch máy từ tiếng Anh sang tiếng Việt
<b>Giáo viên hướng dẫn:</b> TS. Ngô Huy Biên
<b>Thời gian thực hiện:</b> 20/11/2019 – 06/2020
<b>Sinh viên thực hiện:</b> <ul style="list-style-type: none"><li>• Trương Phạm Nhật Tiến -1612689</li><li>• Nguyễn Minh Trí -1612726</li></ul>
<b>Loại đề tài:</b> Nghiên cứu và ứng dụng

### Nội Dung Đề Tài:

1. Trình bày lý thuyết nền tảng và giải pháp đề xử lý việc dịch một văn bản từ tiếng Anh sang tiếng Việt.
2. Xây dựng, thu thập dữ liệu và đào tạo mô hình để dịch một văn bản từ tiếng Anh sang tiếng Việt.
3. Xây dựng một trang web demo việc sử dụng mô hình để dịch một văn bản từ tiếng Anh sang tiếng Việt.

### Kế Hoạch Thực Hiện:

Thời gian thực hiện	Công việc thực hiện	Người thực hiện
20/11/2019 – 25/11/2019	<ul style="list-style-type: none"><li>• Nhận đề tài</li><li>• Xây dựng bản kế hoạch sơ bộ cho các công việc cần thực hiện</li></ul>	Tiến, Trí

26/11/2019 – 02/12/2019	<ul style="list-style-type: none"> <li>• Tìm hiểu và phân tích các yêu cầu về kiến thức cho đề tài</li> <li>• Khảo sát và dùng thử các hệ thống cung cấp dịch vụ mẫu có sẵn trên thị trường</li> </ul>	Tiến , Trí
03/12/2019 – 05/12/2019	<ul style="list-style-type: none"> <li>• Thống nhất nội dung chính của ứng dụng demo việc sử dụng API</li> </ul>	Tiến, Trí
06/12/2019 – 15/02/2019	<ul style="list-style-type: none"> <li>• Tìm hiểu lý thuyết nền tảng trong máy học</li> <li>• Tìm hiểu lý thuyết nền tảng trong dịch máy</li> </ul>	Tiến, Trí
20/12/2019 – 26/12/2019	<ul style="list-style-type: none"> <li>• Executive summary</li> <li>• Project vision</li> </ul>	Tiến, Trí
27/12/2019 – 02/01/2020	<ul style="list-style-type: none"> <li>• Tạo Ec2</li> <li>• Tạo trello</li> </ul>	Tiến, Trí
03/01/2020 – 10/01/2020	<ul style="list-style-type: none"> <li>• Viết release plan</li> <li>• Product backlog</li> <li>• Rise management</li> </ul>	Tiến, Trí

11/01/2020 – 01/02/2020	<ul style="list-style-type: none"> <li>• Tìm hiểu về các thư việc Scikit-Learn, Tensorflow, Keras</li> </ul>	Tiến, Trí
02/02/2020 – 15/02/2020	<ul style="list-style-type: none"> <li>• Tìm hiểu các model và kiến trúc, chạy thử các ví dụ để đánh giá</li> </ul>	Tiến, Trí
16/02/2020 - 22/02/2020	<ul style="list-style-type: none"> <li>• Chạy thử mô hình dịch máy từ tiếng Anh sang các ngôn ngữ khác</li> </ul>	Tiến, Trí
23/02/2020 - 29/02/2020	<ul style="list-style-type: none"> <li>• Thu thập dữ liệu ngôn ngữ</li> <li>• Viết chương 1 luận văn</li> </ul>	Tiến, Trí
01/03/2020 - 06/03/2020	<ul style="list-style-type: none"> <li>• Chỉnh sửa dữ liệu âm thanh</li> <li>• Tìm hiểu và xây dựng mô hình dịch máy từ tiếng Anh sang tiếng Việt</li> <li>• Chỉnh sửa chương 1 luận văn</li> </ul>	Tiến, Trí
07/03/2020 - 15/03/2020	<ul style="list-style-type: none"> <li>• Huấn luyện mô hình</li> <li>• Viết chương 2 luận văn</li> </ul>	Tiến, Trí
16/03/2020 - 21/03/2020	<ul style="list-style-type: none"> <li>• Cải tiến mô hình</li> </ul>	Tiến, Trí



	<ul style="list-style-type: none"> <li>• Chỉnh sửa chương 2 luận văn</li> </ul>	
22/03/2020 - 30/03/2020	<ul style="list-style-type: none"> <li>• Viết chương 3 luận văn</li> <li>• Chỉnh sửa chương 3 luận văn</li> </ul>	Tiến, Trí
01/04/2020 - 07/04/2020	<ul style="list-style-type: none"> <li>• Xây dựng và triển khai hệ thống cung cấp dịch vụ web (API)</li> <li>• Viết chương 4 luận văn</li> </ul>	Tiến, Trí
08/04/2020 - 15/04/2020	<ul style="list-style-type: none"> <li>• Xây dựng ứng dụng demo việc sử dụng API trên nền tảng web</li> <li>• Chỉnh sửa chương 4 luận văn</li> </ul>	Tiến, Trí
16/04/2020 - 21/04/2020	<ul style="list-style-type: none"> <li>• Viết chương 5 luận văn</li> <li>• Chỉnh sửa chương 5 luận văn</li> </ul>	Tiến, Trí
21/04/2020 - 30/04/2020	<ul style="list-style-type: none"> <li>• Hoàn thành luận văn</li> <li>• Chỉnh sửa và cải thiện hiệu năng ứng dụng demo</li> </ul>	Tiến, Trí

01/05/2020 - 30/05/2020	<ul style="list-style-type: none"> <li>• Nâng cấp mô hình hoàn thiện hơn</li> <li>• Cải thiện hiệu năng hệ thống cung cấp dịch vụ web(API)</li> </ul>	Tiến, Trí
03/06/2019 - 19/06/2019	<ul style="list-style-type: none"> <li>• Hoàn chỉnh cuốn luận văn</li> </ul>	Tiến, Trí
20/06/2019 - 30/06/2019	<ul style="list-style-type: none"> <li>• Hoàn chỉnh slide trình bày</li> <li>• Hoàn chỉnh sản phẩm khoá luận</li> </ul>	Tiến, Trí
<b>Xác nhận của GVHD</b>		<b>Ngày.....tháng.....năm.....</b>  <b>SV Thực hiện</b>

## MỤC LỤC

<b>CHƯƠNG 1: GIỚI THIỆU LUẬN VĂN .....</b>	<b>16</b>
<b>1.1 GIỚI THIỆU ĐỀ TÀI .....</b>	<b>16</b>
<b>1.2 LÝ DO LỰA CHỌN ĐỀ TÀI .....</b>	<b>17</b>
<b>1.3 HƯỚNG PHÁT TRIỂN CỦA LUẬN VĂN .....</b>	<b>20</b>
<b>1.4 MỤC TIÊU CỦA LUẬN VĂN .....</b>	<b>21</b>
<b>CHƯƠNG 2: LÝ THUYẾT NỀN TẢNG.....</b>	<b>23</b>
<b>2.1 LÝ THUYẾT NỀN TẢNG CỦA DỊCH MÁY: .....</b>	<b>23</b>
<b>2.1.1 Định nghĩa: .....</b>	<b>23</b>
<b>2.1.1.1 Định nghĩa dịch máy: .....</b>	<b>23</b>
<b>2.1.1.2 Word embeddings: .....</b>	<b>23</b>
<b>2.1.1.2.1 Word2vec: .....</b>	<b>25</b>
<b>2.1.1.2.2 Skip-gram:.....</b>	<b>25</b>
<b>2.1.1.3 Continuous bag of words (CBOW) .....</b>	<b>26</b>
<b>2.1.1.4 Thuật toán tìm kiếm tham lam và thuật toán tìm kiếm</b> <b>chùm tia (Greedy Search và Beam Search) .....</b>	<b>27</b>
<b>2.1.1.5 Bleu Score .....</b>	<b>29</b>
<b>2.1.2 Lý thuyết nền tảng mạng nơ-ron (Neural Network) .....</b>	<b>29</b>
<b>2.1.2.1 Mô tả mạng nơ-ron: .....</b>	<b>29</b>
<b>2.1.2.2 Hàm kích hoạt (Activation function) .....</b>	<b>31</b>
<b>2.1.2.3 Lan truyền ngược (Back propagation).....</b>	<b>34</b>
<b>2.1.2.4 Lan truyền ngược (Back propagation).....</b>	<b>35</b>

2.1.2.5 Phương pháp giảm độ dốc với Gradient Descent và các biến thể .....	37
2.1.2.5.1 Giảm độ dốc theo lô nhỏ (Mini-batch Gradient Descent) .....	37
2.1.2.5.2 Phương pháp giảm độ dốc với động lượng (Momentum) .....	38
2.1.3 Các phương pháp huấn luyện mạng nơ-ron hiện đại .....	40
2.1.3.1 Hàm kích hoạt đơn vị tuyến tính chỉnh lưu (Rectified Linear Unit) .....	40
2.1.3.2 Phương pháp chuẩn hoá hàng loạt (Batch Normalization) .....	42
2.1.3.3 Phương pháp cắt giảm (Dropout) .....	43
2.1.4 Các kiến trúc mạng nơ-ron hồi quy .....	44
2.1.4.1 Mạng nơ-ron hồi quy (RNN – Recurrent Neural Network) ..	44
2.1.4.2 Mạng bộ nhớ dài ngắn (Long Short Term Memory - LSTM) .....	47
2.1.4.3 Mạng nơ-ron hồi quy hai chiều (Bidirectional Recurrent Neural Network – BiRNN) .....	50
2.1.4.4 Mạng nơ-ron hồi quy sâu (Deep Recurrent Neural Network – Deep RNN) .....	51
2.2 MÔ HÌNH DỊCH MÁY: .....	52
2.2.1 Giới thiệu và đặt vấn đề .....	52
2.2.2 Mô hình dịch máy Sequence to Sequence với cơ chế chú ý (Attention Mechanism) .....	55

<b>CHƯƠNG 3: GIẢI PHÁP ĐỀ TÀI.....</b>	<b>59</b>
<b>3.1 TỔNG QUAN GIẢI PHÁP KIẾN TRÚC MÔ HÌNH .....</b>	<b>59</b>
<b>3.2 GIẢI PHÁP BIỂU DIỄN TỪ.....</b>	<b>60</b>
3.2.1 Tổng quan về giải pháp.....	60
3.2.2 Chi tiết giải pháp .....	61
<b>3.3 GIẢI PHÁP XÂY DỰNG MÔ HÌNH DỊCH MÁY .....</b>	<b>64</b>
3.3.1 Tổng quan về giải pháp.....	64
3.3.2 Mô hình mạng nơ-ron hồi quy và khung huấn luyện .....	65
<b>3.4 GIẢI PHÁP XÂY DỰNG MÁY CHỦ .....</b>	<b>66</b>
<b>3.5 GIẢI PHÁP XÂY DỰNG ỨNG DỤNG .....</b>	<b>66</b>
3.5.1 Thiết kế giao diện ứng dụng .....	67
3.5.2 Thiết kế kiến trúc ứng dụng .....	67
<b>3.6 TỔNG KẾT .....</b>	<b>68</b>
<b>CHƯƠNG 4: CÀI ĐẶT VÀ TRIỂN KHAI.....</b>	<b>70</b>
<b>4.1 GIỚI THIỆU VỀ PYTHON VÀ THU VIỆN TENSORFLOW.....</b>	<b>70</b>
4.1.1 Python.....	70
4.1.2 Tensorflow .....	70
<b>4.2 DỮ LIỆU HUẤN LUYỆN MÔ HÌNH .....</b>	<b>72</b>
<b>4.3 CÀI ĐẶT.....</b>	<b>73</b>
4.3.1 Giới thiệu .....	73
4.3.2 Cài đặt .....	73
<b>4.4 HUẤN LUYỆN MÔ HÌNH.....</b>	<b>74</b>
4.4.1 Điều chỉnh num_layer .....	75

4.4.2 Điều chỉnh num_hidden .....	75
4.4.3 Điều chỉnh batch_size .....	76
4.5 ĐÓNG GÓI MÔ HÌNH.....	76
4.6 XÂY DỰNG MÁY CHỦ (SERVER) .....	77
4.7 MỘT SỐ VẤN ĐỀ PHÁT SINH VÀ GIẢI PHÁP .....	77
4.8 TỔNG KẾT .....	78
CHƯƠNG 5: TỔNG KẾT VÀ ĐÁNH GIÁ.....	79
5.1 KIẾN THỨC ĐẠT ĐƯỢC.....	79
5.2 KẾT QUẢ MÔ HÌNH HUẤN LUYỆN.....	79
5.3 KẾT QUẢ HỆ THỐNG .....	80
5.3.1 Môi trường phát triển.....	80
5.3.2 Môi trường triển khai .....	83
5.3.3 Chức năng đã cài đặt .....	83
5.4 KẾT QUẢ ỨNG DỤNG WEB.....	83
5.4.1 Môi trường phát triển.....	83
5.4.2 Môi trường triển khai .....	83
5.4.3 Chức năng đã cài đặt .....	83
5.5 SO SÁNH KẾT QUẢ VỚI CÁC MỤC TIÊU ĐẶT RA .....	85
5.6 ĐỊNH HƯỚNG PHÁT TRIỂN VÀ NGHIÊN CỨU TRONG TƯƠNG LAI .....	86
LỜI KẾT .....	87
TÀI LIỆU THAM KHẢO .....	88



# CHƯƠNG 1: GIỚI THIỆU LUẬN VĂN

## 1.1 GIỚI THIỆU ĐỀ TÀI

Trong cuộc sống hiện nay có rất nhiều thiết bị công nghệ được áp dụng rộng rãi và phục vụ cho nhiều mục đích khác nhau trong cuộc sống và hiện nay đang càng được quan tâm để mở rộng. Một trong những vấn đề đã và đang được phát triển đó chính là dịch máy. Trên thế giới có rất nhiều ngôn ngữ nói, viết khác nhau trên thế giới và sự khác biệt về ngôn ngữ là một trở ngại lớn trong hầu hết các mặt của đời sống. Sự khác biệt đó làm cho con người khó tiếp thu được các kiến thức đến từ các ngôn ngữ khác và nó cũng làm cho sự toàn cầu hoá bị tác động chậm lại.

Trên thế giới có khoảng bảy ngàn ngôn ngữ khác nhau, mỗi ngôn ngữ lại bị phân hoá theo từng vùng miền, quốc gia mà cách viết cũng khác nhau dẫn đến sự hình thành phương ngữ lẫn các biến thể ngôn ngữ dẫn đến các dịch vụ dịch máy vì thế cũng bị hạn chế đi rất nhiều. Ở các nước phát triển trên thế giới đã có nhiều dịch vụ dịch máy được phát triển. Tuy nhiên, các dịch vụ này phục vụ chủ yếu cho thị trường của họ nên những ngôn ngữ khác thì hạn chế hơn.

Vấn đề đặt ra khi sử dụng dịch vụ này, nhà phát triển ứng dụng chỉ quan tâm về độ hiệu quả của dịch vụ cho cặp ngôn ngữ đang sử dụng mang lại lợi ích trực tiếp cho họ.

Vì thế, con người mong muốn có thêm những công cụ hỗ trợ họ một phần giúp họ có thể tạm thời bỏ qua sự khác biệt ngôn ngữ khác nhau để hoàn thành một mục đích của họ. Từ đó các dịch vụ về ngôn ngữ được ra đời và phát triển một cách nhanh chóng.

Trong những năm gần đây, dịch máy đóng vai trò như một công cụ hỗ trợ con người có thể bỏ qua rào cản ngôn ngữ để cập nhật thông tin từ nhiều nguồn khác nhau một cách dễ dàng. Nó đã giúp con người rất nhiều trong các công việc học tập và làm việc một cách hiệu quả hơn.



Nếu như ngày xưa để hiểu được một câu tiếng Anh bạn có thể cần phải tra trong từ điển dày cộm tốn thời gian và không hiệu quả. Thế nhưng ngày nay với các phần mềm dịch tiếng Anh sang tiếng Việt online, bạn sẽ dễ dàng tra cứu ý nghĩa của các từ, câu, đoạn văn bản mình cần trong thời gian ngắn gọn như là ngay lập tức.

Với sự phát triển kinh tế và toàn cầu hoá, các dịch vụ dịch cũng như các ứng dụng sử dụng dịch vụ này không chỉ hấp dẫn người sử dụng mà còn thu hút giới công nghệ trên thế giới. Một dịch vụ dịch máy thường sở hữu những đặc điểm nổi bật như:

- Sử dụng hoàn toàn miễn phí, kho từ phong phú.
- Phần mềm hỗ trợ dịch với độ chính xác cao.
- Sử dụng phần mềm một cách dễ dàng.
- Đôi khi không cần kết nối mạng, chúng ta vẫn có thể sử dụng các chức năng phiên dịch từ của những ứng dụng này.
- Trợ giúp tự nhiên nhất cho người sử dụng.
- Tiềm năng kinh doanh lợi nhuận cũng rất to lớn.

Từ đó, việc dịch máy và các ứng dụng sử dụng dịch máy đã đem lại một lợi ích khổng lồ trong cuộc sống của con người.

## **1.2 LÝ DO LỰA CHỌN ĐỀ TÀI**

Nhằm mục đích phát triển kiến thức của bản thân, chúng em quyết định lựa chọn đề tài “Xây dựng mô hình dịch máy từ tiếng Anh sang tiếng Việt” để tìm hiểu thêm các kiến thức, có sản phẩm mang ý nghĩa thực tế và có tiềm năng trong tương lai.

Ngoài ra việc lựa chọn đề tài này giúp chúng em có thêm các kiến thức về các thư viện, các mã nguồn mở là những nguồn tài liệu quý giá đối với chúng em. Với các lợi ích kể trên và sau khi thực hiện luận, chúng em sẽ có thêm hiểu biết về các

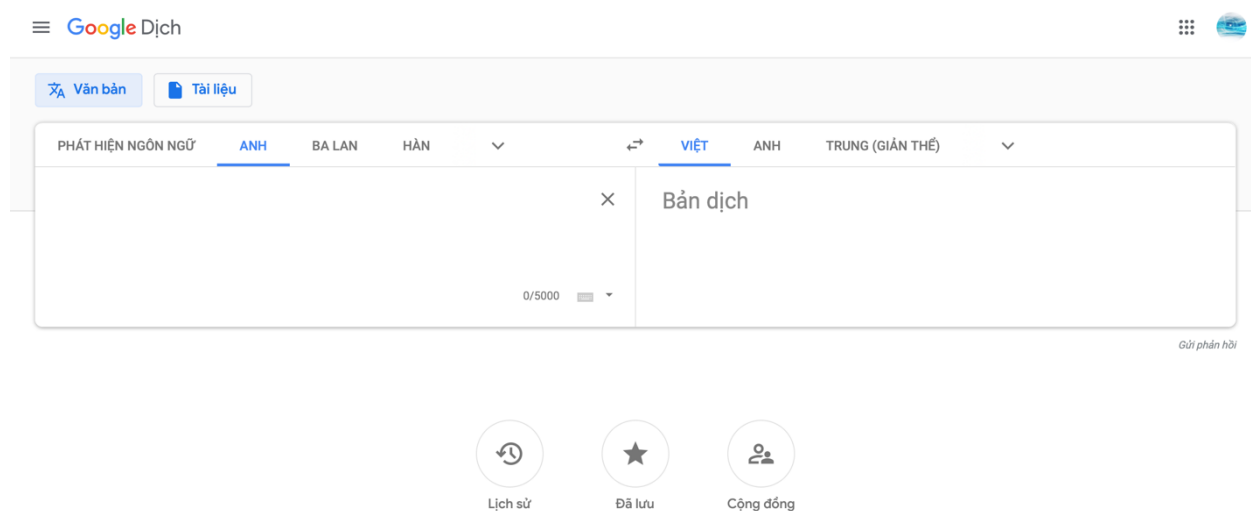
quy trình phát triển dự án thực , không những thế chúng em còn được học hỏi các kiến thức nền tảng và chuyên sâu liên quan đến dịch máy. Chính những điều đó sẽ là nền tảng giúp chúng em phát triển hơn trong con đường học tập cũng như trong công việc của chúng em.

Như đã trình bày ở trên, các ứng dụng sử dụng dịch vụ dịch máy ngày càng thu hút sự đầu tư và các nhà phát triển phần mềm lẫn người sử dụng phần mềm. Việt Nam đang đi trên con đường toàn cầu hoá nên thị trường Việt Nam là một thị trường đầy tiềm năng. Nhu cầu học tiếng Anh của các học sinh, sinh viên cũng như người dân là rất lớn. Trên thị trường hiện nay có khá nhiều các công cụ miễn phí cũng như thu phí nhưng những ứng dụng này cũng có một số ưu điểm cũng như khuyết điểm của nó.

Với các lý do trên, nhóm sinh viên quyết định chọn đề tài “Xây dựng mô hình dịch máy từ tiếng Anh sang tiếng Việt” để tạo ra một dịch vụ miễn phí và độ chính xác tạm chấp nhận được để phục vụ cho cộng đồng.

## Google Translate

Đây là một sản phẩm rất hữu ích của Google và là phần mềm dịch tiếng Anh sang tiếng Việt đầu tiên được rất nhiều người sử dụng cho các nhu cầu như là học tập.



### Ưu điểm:

- Google Translate có giao diện dễ nhìn, dễ sử dụng, đặc biệt có độ chính xác khá cao.
- Có thể sử dụng phần mềm này trên máy tính, điện thoại hay cả máy tính bảng.
- Miễn phí và dịch được nhiều ngôn ngữ khác nhau trên thế giới.

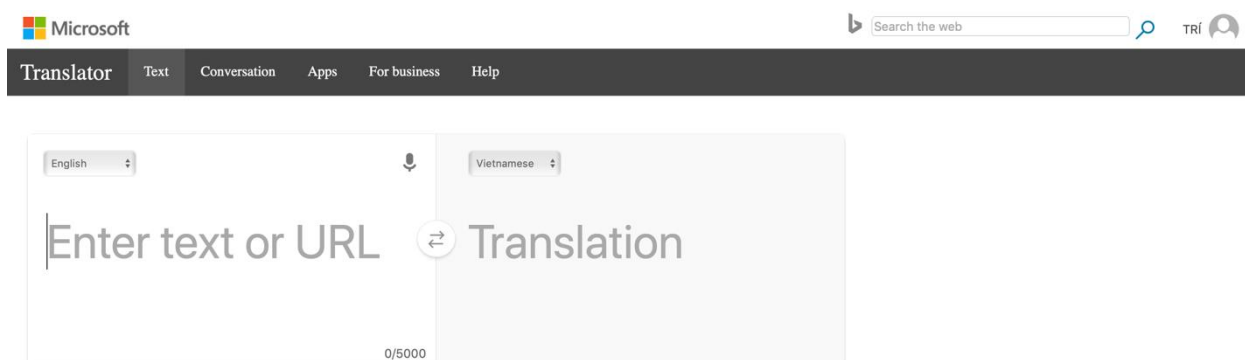
### Nhược điểm:

- Để sử dụng phần mềm này cần phải có kết nối mạng cho thiết bị hoặc phải tải bộ dữ liệu để dùng khi không có mạng.
- Chỉ hiển thị giới hạn 5000 từ mỗi lần và cắt đoạn chưa hợp lí.

**Link tham khảo:** <https://translate.google.com>

### Microsoft Translator

Nói tiếp Google, Microsoft cũng như cho ra đời phần mềm dịch trực tuyến của riêng mình.



### Ưu điểm:

- Giao diện dễ nhìn, độ chính xác cao.
- Có thể sử dụng trên các thiết bị khác nhau như: máy tính, điện thoại.
- Sử dụng miễn phí và dịch được nhiều ngôn ngữ..

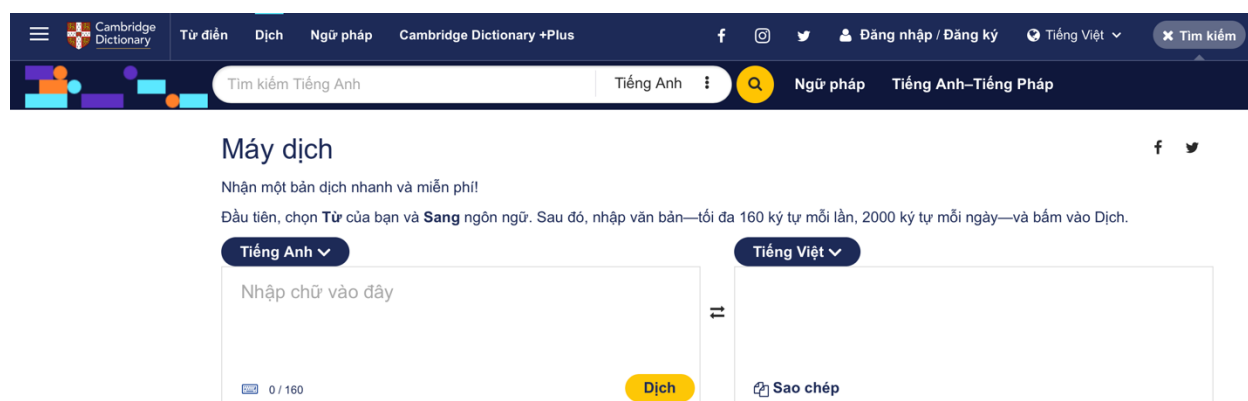
### Nhược điểm:

- Cần có kết nối mạng cho thiết bị.
- Chỉ giới hạn 5000 từ và không tự cắt đoạn văn bản để dịch tiếp.
- Giao diện không cân đối.

Link tham khảo: <https://www.bing.com/translator>

## Cambridge Dictionary Translate

Phần mềm này được xuất bản bởi Đại học Cambridge.



### Ưu điểm:

- Dịch được 23 ngôn ngữ.
- Có thể tự động phát hiện ngôn ngữ.
- Kết quả có độ chính xác cao.

### Nhược điểm:

- Miễn phí tối đa 160 từ trong một lần dịch và 2000 từ trên một ngày.
- Cần phải qua một bước trung gian là ấn nút “Dịch” để thực hiện dịch.
- Cần có kết nối mạng có thiết bị.

Link tham khảo: <https://dictionary.cambridge.org/vi/translate>

## 1.3 HƯỚNG PHÁT TRIỂN CỦA LUẬN VĂN

Trong lịch sử phát triển dịch máy, có rất nhiều phương pháp đã được nghiên cứu và áp dụng đem lại thành công nhất định. Trong sự phát triển của dịch máy, có

cách tiếp cận chủ yếu là dịch chuyển đổi, lịch liên ngữ và dịch dựa trên dữ liệu. Trong đó, dịch máy thống kê, một trong những phương pháp theo cách tiếp cận dựa trên dữ liệu, hiện đang là một hướng pháp triển đầy tiềm năng, thu hút được nhiều sự quan tâm của các nhà nghiên cứu.

Ưu điểm vượt trội của phương pháp dịch máy thống kê là thay vì xây dựng các quy luật, từ điển được chuyển đổi bằng tay, nó tự động thiết lập các quy luật, từ điển dựa trên kết quả thống kê có được từ kho ngữ liệu. Chính vì thế nên dịch máy thống kê có tính linh hoạt cao có thể áp dụng được cho bất kì cho một cặp ngôn ngữ ngẫu nhiên. Dịch máy thống kê hiện nay có 3 hướng tiếp cận chính đó là: dịch máy thống kê theo đơn vị từ, dựa trên đơn vị cụm từ và dựa trên cú pháp. Tuy nhiên các phương pháp này vẫn có những hạn chế do sự thiếu hụt về thông tin ngôn ngữ. Mô hình dịch thống kê vẫn chưa giải quyết được một số vấn đề còn sai sót như trật tự từ, khả năng chọn cụm từ phù hợp.

Luận văn sử dụng phương pháp dịch máy dựa thống kê để tiếp cận hoàn toàn dựa trên ngữ liệu nên nó hoàn toàn độc lập với ngôn ngữ. Những tham số thống kê thu được từ việc huấn luyện trên ngữ liệu song ngữ sẽ được sử dụng cho các lần dịch sau.

Để hoàn thành luận văn, nhóm sinh viên tiến hành xây dựng một mô hình dịch máy từ tiếng Anh sang tiếng Việt và có ứng dụng thử nghiệm để đánh giá mô hình.

#### **1.4 MỤC TIÊU CỦA LUẬN VĂN**

Để hoàn thành tốt đề tài luận văn, bản luận văn và sản phẩm cuối cùng của nhóm sinh viên sẽ đảm bảo các mục tiêu sau đây:

- Trình bày lý thuyết nền tảng và giải pháp để xử lý việc dịch một văn bản từ tiếng Anh sang tiếng Việt.

- Xây dựng, thu thập dữ liệu và đào tạo mô hình để dịch một văn bản từ tiếng Anh sang tiếng Việt.
- Xây dựng trang web demo việc sử dụng mô hình để dịch một văn bản từ tiếng Anh sang tiếng Việt
- Viết 120 trang luận văn trình bày các nội dung liên quan theo đúng chuẩn nhà trường yêu cầu và có trích dẫn tài liệu tham khảo một cách chi tiết và đầy đủ.

## **1.5 PHẠM VI ĐỀ TÀI**

Sản phẩm được tạo ra hướng đến những người có mong muốn sử dụng các công cụ dịch.

- Sản phẩm chỉ cung cấp dịch từ tiếng Anh sang tiếng Việt.
- Sản phẩm yêu cầu kết nối internet để sử dụng.
- Sản phẩm luận văn sẽ được áp dụng và mở rộng các thư viện có sẵn cũng như các cắt thô để đáp ứng yêu cầu nhằm đạt được các mục tiêu đã đặt ra.

## **Chương 2: LÝ THUYẾT NỀN TẢNG**

### **2.1 LÝ THUYẾT NỀN TẢNG CỦA DỊCH MÁY:**

Ở chương 1, nhóm sinh viên đã đề xuất sử dụng phương pháp dịch máy dựa trên thống kê để tiếp cận bằng ngữ liệu nên nó độc lập với ngôn ngữ. Trong luận văn, nhóm sinh viên thực hiện xây dựng mô hình dịch máy Sequence to Sequence kết hợp với Attention Mechanism (cơ chế chú ý) để phát triển mô hình. Sau đây nhóm sinh viên sẽ trình bày các kiến thức liên quan được ghi lại trong quá trình thực hiện luận văn để xây dựng một mô hình dịch máy.

#### **2.1.1 Định nghĩa:**

Phần này trình bày một số khái niệm cốt lõi về dịch máy, các mô hình ngôn ngữ và phương pháp học theo đặc trưng trong xử lý ngôn ngữ tự nhiên (Neural Language Processing).

##### **2.1.1.1 Định nghĩa dịch máy:**

Dịch máy (machine translation) là một quá trình thay đổi văn bản từ ngôn ngữ này sang ngôn ngữ khác (gọi là ngôn ngữ đích) một cách tự động, không có sự can thiệp của con người trong quá trình dịch.

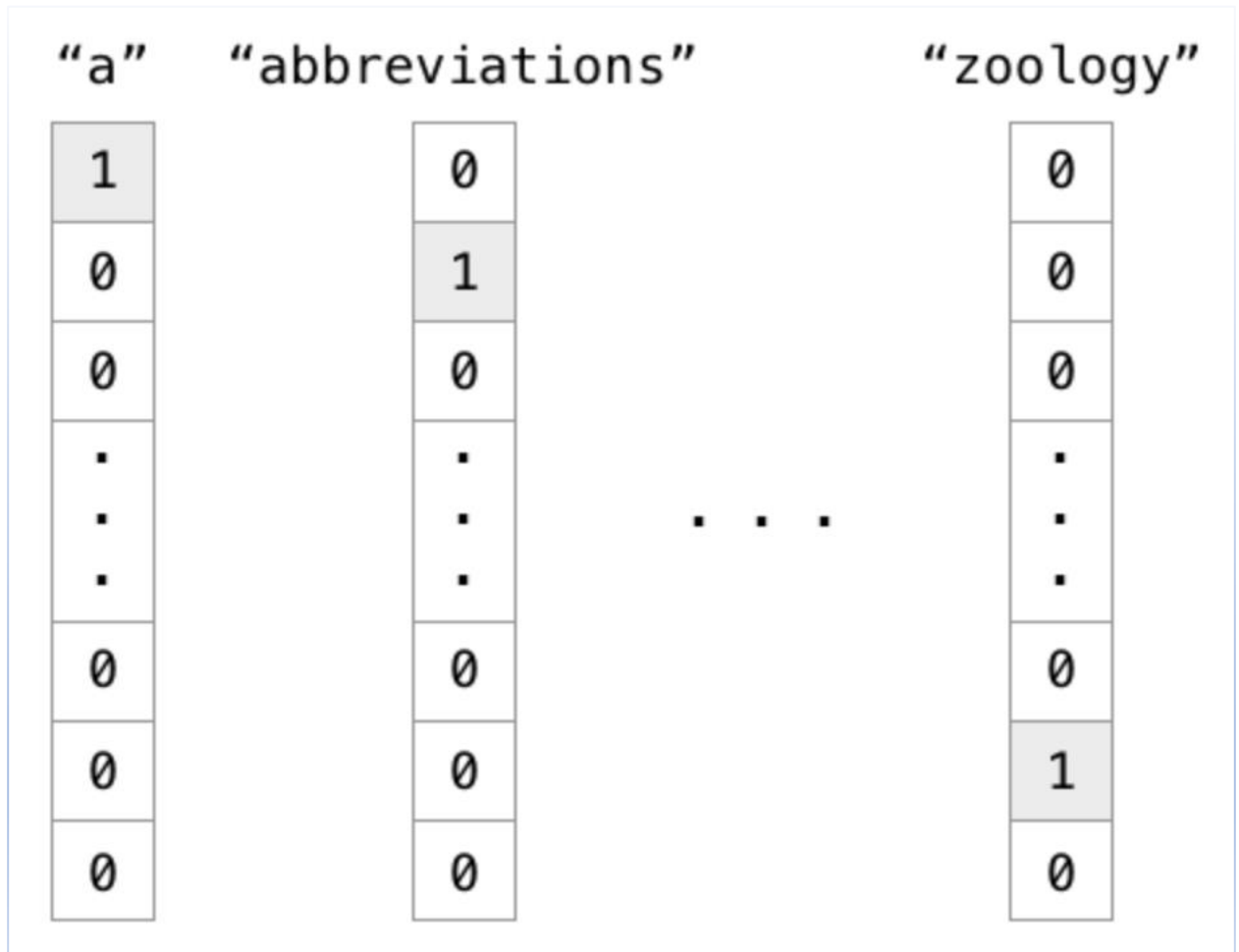
##### **2.1.1.2 Word embeddings:**

Xử lý đầu vào cho bài toán dịch máy là một bước rất quan trọng, các thuật toán, vì các kiến trúc Machine learning, Deep learning chúng chỉ có thể hiểu được đầu vào ở dạng là số nên cần chuyển đầu vào ở dạng text sang dạng số để chúng có thể hiểu được.

Nhưng nếu chỉ đơn giản biểu diễn từ bằng một con số có thể dẫn đến sai lệch mối quan hệ ngữ nghĩa giữa các từ. Ví dụ như nếu đánh dấu “mèo” là số 1 và “chó” là số 2, như vậy “mèo” + “mèo” = “chó”.

Một kỹ thuật đơn giản được sử dụng để khắc phục là One-hot vector, chúng chuyển các từ thành vector có số chiều bằng số từ của bộ từ vựng đầu vào,

trong đó chỉ có duy nhất một phần tử bằng 1 (các phần tử khác bằng 0) tương ứng với vị trí từ đó trong bộ từ vựng. Tuy nhiên cách biểu diễn này là số chiều của vector lại rất lớn, ảnh hưởng đến quá trình xử lý và lưu trữ.



Hình 2.1: Hình mô tả cách mã hóa one-hot-vector

(Nguồn: Leonardo Barazza)

Một cách khác là sử dụng vector ngẫu nhiên, mỗi từ được biểu thị bằng một vector có giá trị các chiều là ngẫu nhiên, mỗi từ là một điểm trong không gian 3D, do đó làm giảm số chiều vector, tuy nhiên nó lại không biểu diễn quan hệ tương đồng giữa các từ.



Sử dụng Word embeddings được coi là cách tốt nhất để thể hiện các từ trong văn bản nó cũng gán mỗi từ với một vector nhưng các vector được tính toán để biểu diễn quan hệ tương đồng giữa các từ.

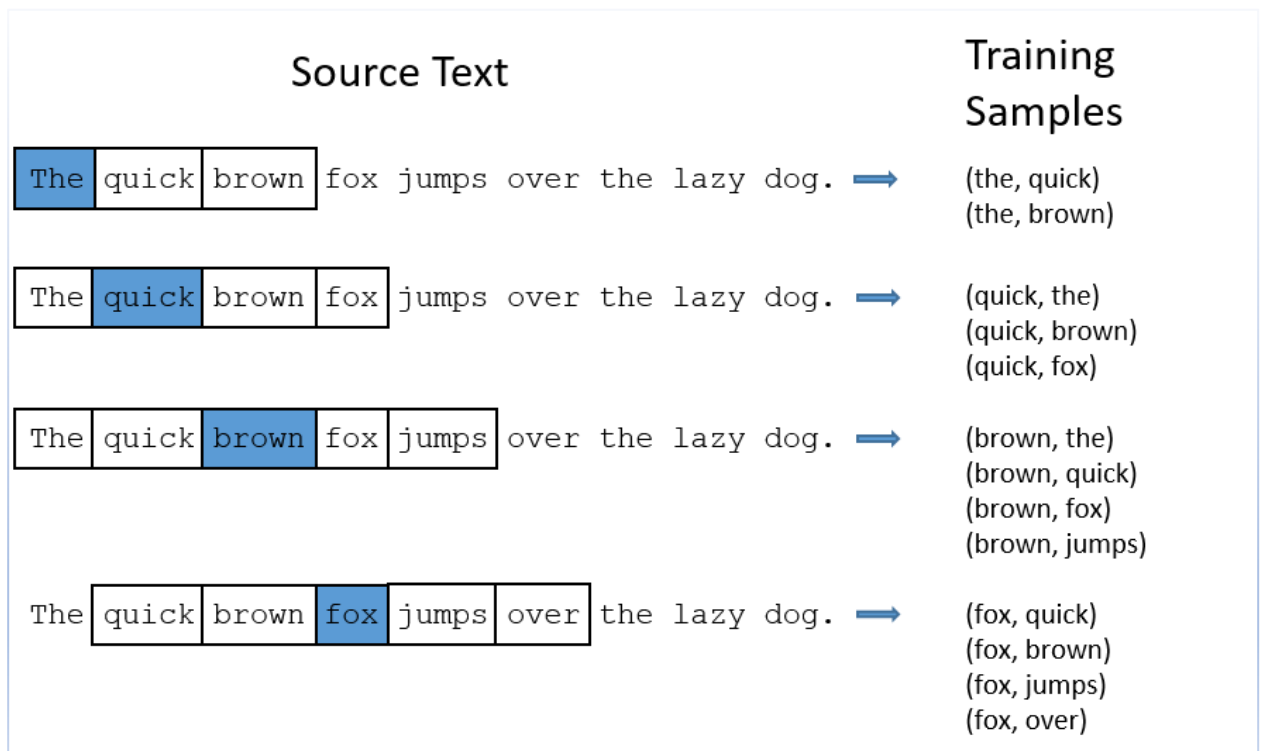
Word embeddings có 2 model nổi tiếng là Word2vec và Glove.

#### 2.1.1.2.1 Word2vec:

Word2vec là một model unsupervised learning nó dùng để thể hiện mối quan hệ giữa các từ, nó được kết hợp từ hai thuật toán Skip-gram và Continuous bag of words (CBOW).

#### 2.1.1.2.2 Skip-gram:

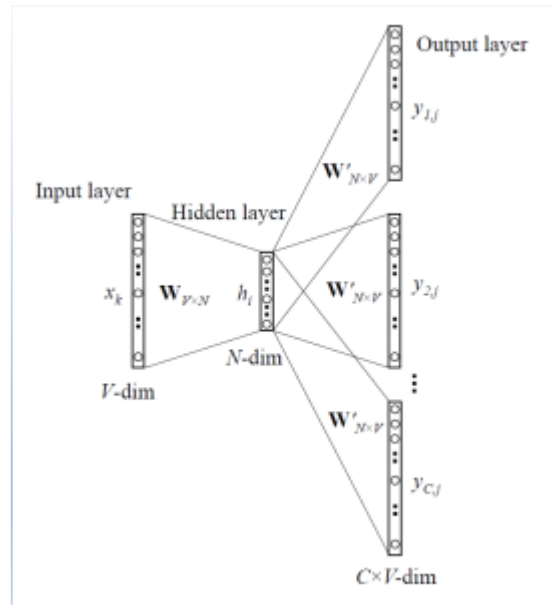
Ý tưởng chính của mô hình này là xác định các từ xung quanh từ mục tiêu trong một khoảng nhất định gọi là ‘window’.



Hình 1.2: Hình Mô Tả Trainning Với Window Bằng 2.

(Nguồn: Leonardo Barazza)

Đối với Skip-gram, đầu vào là từ đích, trong khi đầu ra là các từ xung quanh từ đích. Tất cả dữ liệu đầu vào và đầu ra có cùng kích thước được mã hóa bằng one-hot. Mạng chứa một lớp ẩn có kích thước bằng kích thước nhúng, nhỏ hơn vector đầu vào và đầu ra. Ở cuối lớp đầu ra, một hàm kích hoạt softmax được áp dụng sao cho mỗi phần tử của vector đầu ra mô tả khả năng một từ cụ thể sẽ xuất hiện trong ngữ cảnh.



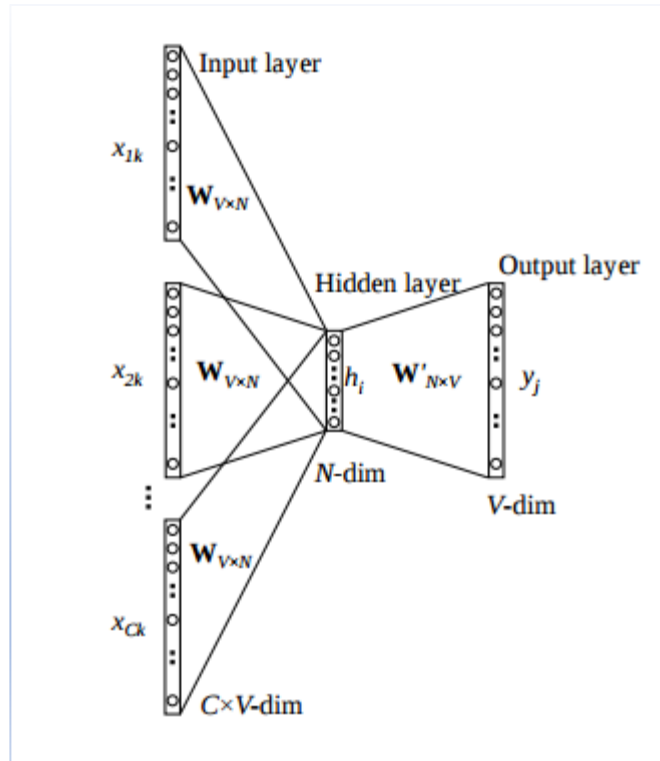
Hình 2.3: Hình mô tả cấu trúc mạng của Skip-gram

(Nguồn <https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/>)

Với skip-gram, kích thước biểu diễn từ giảm từ kích thước bằng số từ trong bộ từ vựng xuống bằng chiều dài lớp ẩn. Hơn nữa các vector có ý nghĩa nhiều hơn về mặt mô tả mối quan hệ giữa các từ.

### 2.1.1.3 Continuous bag of words (CBOW)

Ngược lại với Skip-gram nó hoán đổi đầu vào và đầu ra, ý tưởng của thuật toán CBOW là đưa ra một bối cảnh và cho biết từ nào có khả năng xuất hiện nhiều nhất trong đó.



Hình 2.4: Mô Tả Mạng CBOW

(Nguồn <https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/>)

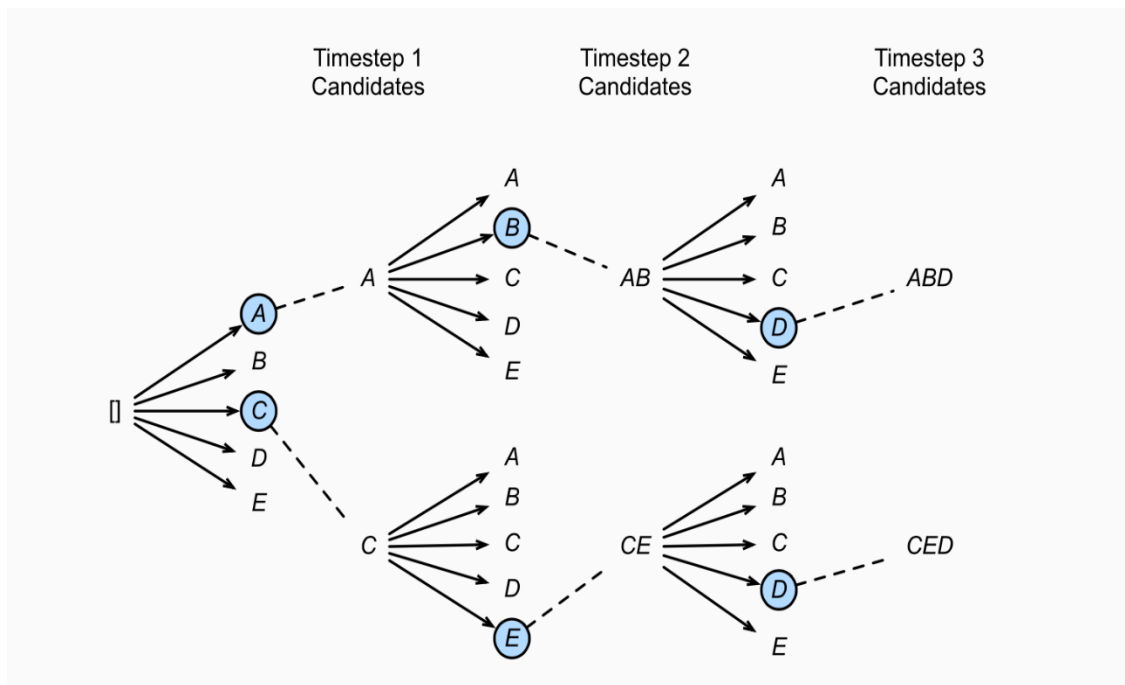
#### 2.1.1.4 Thuật toán tìm kiếm tham lam và thuật toán tìm kiếm chùm tia (Greedy Search và Beam Search)

Thuật toán Greedy Search chọn một ứng cử viên tốt nhất làm chuỗi đầu vào cho mỗi bước thời gian (ứng cử viên có xác suất cao nhất). Chọn chỉ một ứng cử viên tốt nhất có thể phù hợp với bước thời gian hiện tại, nhưng khi xây dựng câu đầy đủ, nó có thể là một lựa chọn tối ưu. Tuy nhiên thì xác suất cao nhất ở bước hiện tại chưa chắc sẽ cho ra xác suất cao nhất ở bước tiếp theo, vậy nên thay vì chỉ giữ 1 kết quả có xác suất cao nhất chúng ta sẽ giữ lại k kết quả có xác suất cao nhất và đó chính là Beam Search.

Thuật toán Beam Search chọn nhiều lựa chọn thay thế cho chuỗi đầu vào tại mỗi dấu thời gian dựa trên xác suất có điều kiện. Số lượng nhiều lựa chọn

thay thế phụ thuộc vào một tham số gọi là Beam Width B. Ở mỗi bước thời gian, tìm kiếm chùm tia chọn B số lựa chọn thay thế tốt nhất với xác suất cao nhất là lựa chọn khả dĩ nhất cho bước thời gian (ví dụ với Beam Width  $B = 2$  thì tại mỗi bước thời gian sẽ chọn ra 2 ứng cử viên có xác suất cao nhất làm đầu vào cho bước thời gian tiếp theo (t). Sau đó lại tiếp tục chọn 2 ứng cử viên ở bước thời gian tiếp theo (t+1) làm đầu vào ở bước thời gian t+2. Cứ như vậy cho đến cuối cùng ta sẽ thu được 3 kết quả và chọn ra kết quả từ đó). Thuật toán Beam Search nếu có Beam Width  $B = 1$  thì nó trở thành thuật toán Greedy Search. Beam Width  $B = 10$  thường được sử dụng và mang lại hiệu quả đủ tốt.

Thuật toán tìm kiếm chùm tia được minh họa như hình 2.5.



Hình 2.5: Mô tả thuật toán tìm kiếm chùm tia (Beam search) hoạt động với beam-width = 2.

(Nguồn: [https://d2l.ai/chapter\\_recurrent-modern/beam-search.html](https://d2l.ai/chapter_recurrent-modern/beam-search.html))

#### **2.1.1.5 Bleu Score**

BLEU là một thuật toán để đánh giá chất lượng văn bản đã được dịch bằng máy từ ngôn ngữ tự nhiên này sang ngôn ngữ tự nhiên khác. Chất lượng được coi là sự tương ứng giữa đầu ra của máy và của con người: "bản dịch máy càng gần với bản dịch chuyên nghiệp của con người thì càng tốt" - đây là ý tưởng trung tâm của BLEU.

Bilingual Evaluation Understudy Score hay ngắn gọn là BLEU score là một thang điểm được dùng phổ biến trong đánh giá Machine Translation. BLEU được Kishore Papineni và cộng sự đề xuất lần đầu vào năm 2002 qua bài nghiên cứu "a Method for Automatic Evaluation of Machine Translation".

BLEU được tính dựa trên số lượng n-grams[1] giống nhau giữa câu dịch của mô hình (output) với các câu tham chiếu tương ứng (reference) có xét tới yếu tố độ dài của câu.

Số n-grams tối đa của BLEU là không giới hạn, nhưng vì xét về ý nghĩa, cụm từ quá dài thường không có nhiều ý nghĩa, và nghiên cứu cũng đã cho thấy là với 4-gram, điểm số BLEU trung bình cho khả năng dịch thuật của con người cũng đã giảm khá nhiều nên n-grams tối đa thường được sử dụng là 4-gram.

### **2.1.2 Lý thuyết nền tảng mạng nơ-ron (Neural Network)**

#### **2.1.2.1 Mô tả mạng nơ-ron:**

Mạng nơ-ron là một tập hợp các mô hình toán học được xây dựng dựa trên tập hợp các nút, được kết nối với các hàm kích hoạt phi tuyến tính cùng

các tham số có khả năng học. Mạng nơ-ron hiện là mô hình phổ biến nhất được sử dụng cho các ứng dụng máy học trong một loạt các lĩnh vực như thị giác máy tính, nhận dạng giọng nói, xử lý ngôn ngữ tự nhiên,...

Một tế bào (nút) của mạng nơ-ron là một hàm của tập các trọng số tương ứng với các giá trị đầu vào (inputs)  $\{x_0, \dots, x_N\}$ .

Trong đó:

- $w_i$  : trọng số của đầu vào  $x_i$
- $a$ : hàm kích hoạt (activation function)
- $b$ : độ sai lệch (bias)

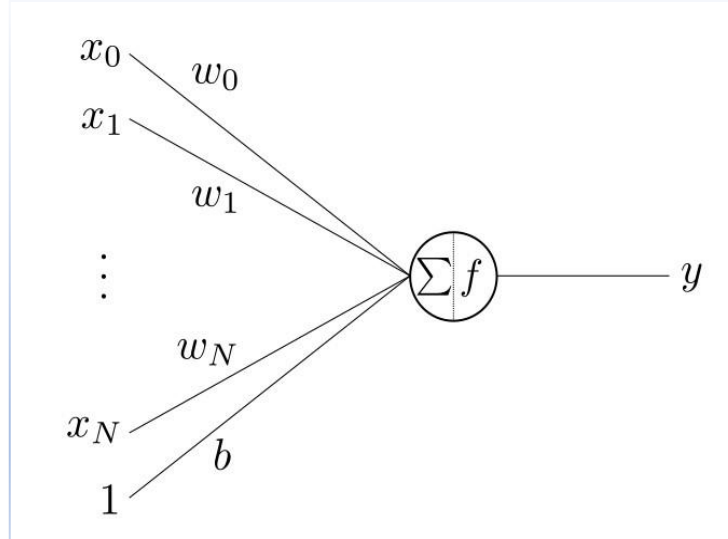
Ta sẽ sử dụng kí hiệu ma trận để làm đơn giản cách thể hiện, trong đó mỗi tế bào nơ-ron bao gồm một vector đầu vào  $x = \{x_0, \dots, x_N\}$ , một vector trọng số  $w = \{w_0, \dots, w_N\}$  và một vector sai lệch  $b$ , khi đó đầu ra là:

$$y = a(w^T x + b)$$

Nếu hàm kích hoạt  $a$  là một biến thể của hàm Heaviside,

$$a(x) = \begin{cases} 1, & x \geq 0 \\ 0 \text{ hoặc } -1, & x < 0 \end{cases}$$

thì tế bào nơ-ron này được gọi là một perceptron, một bộ phân loại nhị phân đơn giản, là một trong những phương pháp học kết nối sớm nhất được phát minh bởi Rosenblatt.



Hình 2.6: Minh họa kiến trúc điển hình của một tế bào mạng nơ-ron

(Nguồn: [1])

#### 2.1.2.2 Hàm kích hoạt (Activation function)

Hàm kích hoạt là phần rất quan trọng trong mạng nơ-ron, đặc biệt là mạng nơ-ron nhiều lớp ẩn. Nếu không có hàm kích hoạt phi tuyến tính, cho dù mạng nơ-ron có nhiều lớp ẩn đến cỡ nào thì cũng chỉ có sức mạnh đại diện cho phân loại tuyến tính, điều này tương đương với một mạng mà không có lớp ẩn nào. Vì bản chất tổng hợp các hàm tuyến tính là một hàm tuyến tính. Do đó, hàm kích hoạt  $a$  là một hàm phi tuyến tính được áp dụng cho đầu ra tại mỗi nút và input data cho tầng tiếp theo, cho phép mạng nơ-ron nhiều lớp ẩn học các hàm phi tuyến phức tạp.

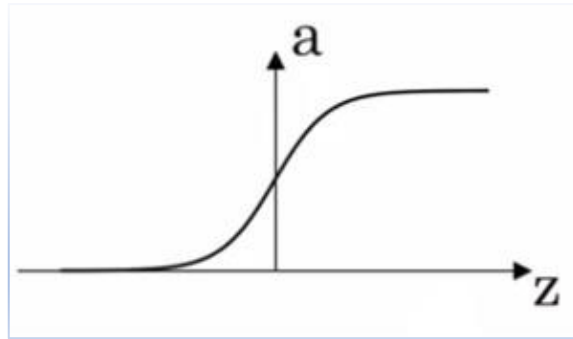
Hàm kích hoạt phổ biến là hàm sigmon, nó là một hàm phi tuyến với đầu vào là các số thực cho kết quả nằm trong khoảng từ 0 đến 1, phù hợp cho các mạng phân loại nhị phân (nổi tiếng như là thuật toán Logistic Regression), nó có công thức theo phương trình:

$$z^{[i]} = W^{[i]}x + b^{[i]}$$

$$a = \frac{1}{1 + e^{-z}}$$

Trong đó:

- $W^{[l]}$ : trọng số tại lớp thứ  $l$
- $b^{[l]}$ : tham số sai lệch tại lớp thứ  $l$
- $a$ : hàm kích hoạt



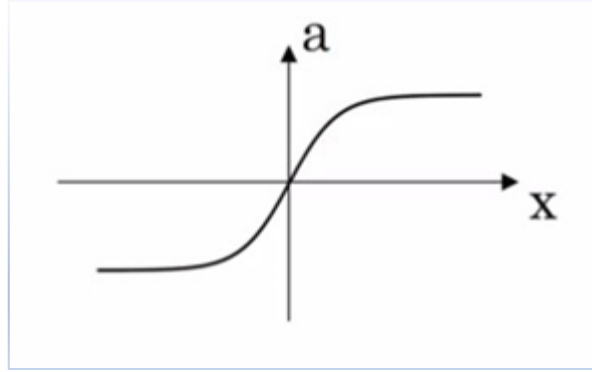
Hình 2.7: Minh họa hàm kích hoạt sigmoid.

(Nguồn: Coursera Sequence Models)

Ngoài ra, có một hàm kích hoạt luôn hoạt động tốt hơn hàm sigmoid là hàm tiếp tuyến hyperbolic (hyperbolic tangent function), ánh xạ các giá trị đầu vào vào khoảng biên từ -1 đến 1. Có công thức là:

$$a = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$





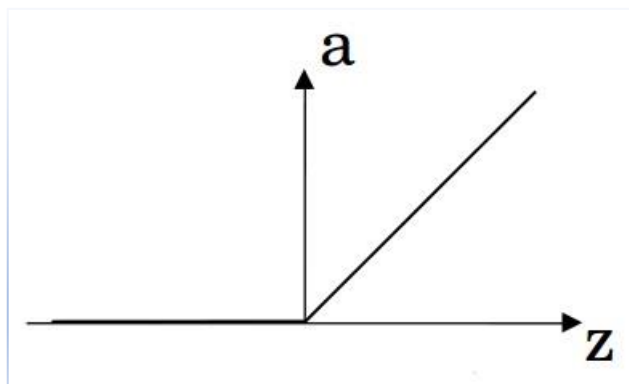
Hình 2.8: Minh họa hàm kích hoạt tanh.

(Nguồn: Coursera Sequence Models)

Tuy nhiên, một vấn đề với hàm kích hoạt sigmoid và tanh nếu  $z$  quá lớn hoặc quá nhỏ thì độ dốc của hàm sẽ rất nhỏ, điều này làm chậm quá trình tìm điểm cực tiểu của hàm chi phí, dẫn đến làm chậm quá trình học. Vì lý do này, dựa vào các kết quả thực nghiệm được cải thiện, mạng nơ-ron hiện đại có xu hướng sử dụng hàm kích hoạt đơn vị tuyến tính chỉnh lưu (ReLU - Rectified Linear Unit). Có công thức như sau:

$$z[i] = W[i]_x + b[i]$$

$$a = \max(0, z)$$



Hình 2.9: Minh họa hàm kích hoạt ReLu.

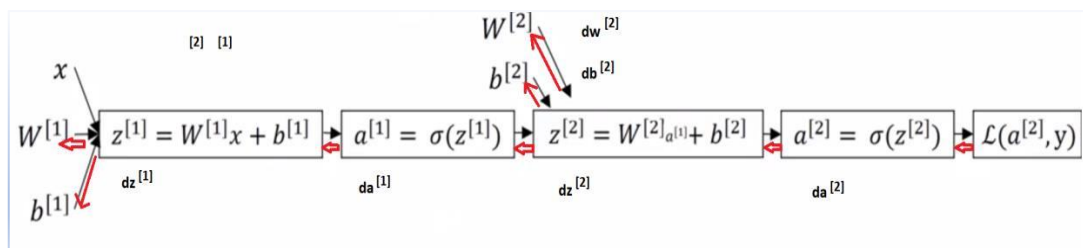
(Nguồn: Coursera Sequence Models)

Vì vậy, đạo hàm luôn bằng 1 nếu  $z$  dương, và bằng 0 nếu  $z$  âm. Dựa trên thực nghiệm, sử dụng hàm kích hoạt ReLu, mạng nơ-ron sẽ học nhanh hơn so với khi dùng với hàm sigmoid hoặc hàm tanh. Lý do chính là có ít hơn sự ảnh hưởng của độ dốc hàm bằng 0 làm chậm việc học. Vì mặc dù, có một nửa phạm vi của  $z$  làm độ dốc hàm ReLu bằng 0, nhưng trong thực tế, đủ các đơn vị ẩn thì ta sẽ có  $z$  lớn hơn 0, vì vậy việc học vẫn khá nhanh với hầu hết các ví dụ đào tạo.

### 2.1.2.3 Lan truyền ngược (Back propagation)

Các thuật toán học sâu tương phản với các thuật toán học nông bởi số biến đổi được tham số hóa một tín hiệu gặp phải khi nó lan truyền từ các lớp đầu vào đến các lớp đầu ra. Mỗi chuỗi các biến đổi từ đầu vào đến đầu ra gọi là một đường gán kế thừa (CAP - Credit Assignment Path). Vấn đề gán kế thừa (Credit Assignment Problem) được giải quyết với khám phá lan truyền ngược (backpropagation), cho phép học với mạng nơ-ron nhiều lớp. Sau đây, nhóm sinh viên sẽ trình bày ý tưởng của quá trình lan truyền ngược.

Hình 2.10 minh họa một mạng nơ-ron 2 lớp, gồm một lớp đầu vào, một lớp ẩn và một lớp đầu ra.



Hình 2.10: Minh họa một mạng nơ-ron 2 lớp

(Nguồn: Coursera Sequence Model)

Lưu ý ở lan truyền tiến, các bước tính toán là như sau: đầu tiên ta tính toán  $z^{[1]}$ , sau đó tính toán  $a^{[1]}$ , rồi tính toán  $z^{[2]}$  ghi chú là  $z^{[2]}$  cũng phụ thuộc vào

các tham số  $W^{[2]}$  và  $b^{[2]}$ , sau đó dựa vào  $z^{[2]}$  tính toán  $a^{[2]}$  và cuối cùng là tính toán chi phí. Đối với lan truyền ngược, ta sẽ đi tính toán theo chiều ngược lại, cụ thể là tính toán  $da^{[2]}$ , sau đó tính  $dz^{[2]}$ , quay ngược lên tính  $dW^{[2]}$  và  $db^{[2]}$ , tương tự ta tính toán tiếp cho các biến  $da^{[1]}$ ,  $dz^{[1]}$ ,  $dW^{[1]}$ ,  $db^{[1]}$ . Thông thường, ta sẽ bỏ qua tính đạo hàm của  $da^{[2]}$ , thay vào đó nhập lại thành một bước là tính trên  $dz^{[2]}$ . Sau cùng ta rút ra các công thức sau:

$$da^{[2]} = a^{[2]} - y$$

$$dW^{[2]} = dz^{[2]}a^{[1]T}$$

$$db^{[2]} = dz^{[2]}$$

$$dz^{[1]} = W^{[2]T} dz^{[2]} * g^{[1]'}(z^{[1]}) dW^{[1]} = dz^{[1]}x^T$$

$$db^{[1]} = dz^{[1]}$$

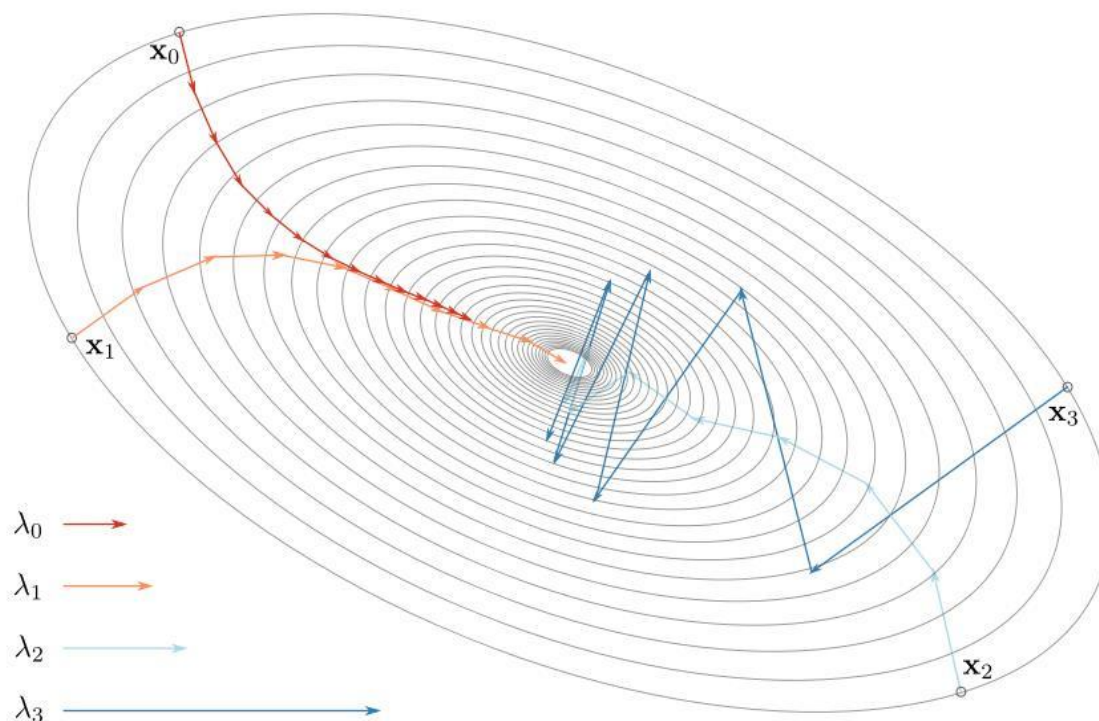
#### 2.1.2.4 Lan truyền ngược (Back propagation)

Học với lan truyền ngược giống như quy tắc delta, các độ nhạy được dùng để điều chỉnh trọng số tỷ lệ với một hằng số tỉ lệ học (learning rate)  $\alpha$ . Việc cập nhật trọng số thực hiện theo công thức sau:

$$\begin{aligned}\Delta w_{ij}^n &= -\alpha \frac{\partial E^n}{\partial w_{ij}} \\ &= -\alpha \delta_j^n y_i\end{aligned}$$

Trong đó:

- $\delta_j^n$ : đã được định nghĩa ở công thức
- $y_i$ : đầu ra ở nơ-ron i



Hình 2.11: Minh họa sự ảnh hưởng của tỉ lệ học và chính sách học lên độ hội tụ với lan truyền ngược. (Nguồn: [1])

Lan truyền ngược là một phương pháp gốc dốc nhất (steepest descent).

Hình 2.11 minh họa quy tắc học với lan truyền ngược, đặc trưng cho kích thước mỗi bước là tham số tỉ lệ học. Tham số tỉ lệ học thay đổi kích thước bước hay độ lớn của vector thay đổi trọng số. Hình 2.11 cũng minh họa độ ảnh hưởng của tỉ lệ học trên độ giảm dốc. Tỉ lệ học quá nhỏ làm cho kết quả học rất chậm như  $a_0$ , tuy nhiên tỉ lệ học quá lớn lại làm cho bước nhảy xung quanh khu vực điểm cực tiểu và làm mất thời gian để tiếp cận điểm cực tiểu này như  $a_2$  và  $a_3$ .

Để đạt được điểm cực tiểu cục bộ, tỉ lệ học cũng nên giảm dần trong quá trình huấn luyện. Tuy nhiên nếu giảm quá nhanh, nó có thể không đạt được lưu vực gần với điểm cực tiểu như  $a_0$ , ngược lại nếu giảm quá chậm, nó có thể mất một khoảng thời gian dài để tiến vào lưu vực này như  $a_2$  và  $a_3$ .

Cân bằng việc cố gắng tìm một tỉ lệ học và chính sách học phù hợp không may là một phần "ma thuật đen" đằng sau việc huấn luyện DNN đến từ kinh nghiệm. Tuy nhiên Bottou (2012) và I. Goodfellow, Y. Bengio, và Courville (2016) là tài liệu tham khảo tuyệt vời về một số các tiếp cận phổ biến để làm công việc này dễ dàng hơn.

### **2.1.2.5 Phương pháp giảm độ dốc với Gradient Descent và các biến thể**

#### **2.1.2.5.1 Giảm độ dốc theo lô nhỏ (Mini-batch Gradient Descent)**

Quá trình áp dụng máy học là một quá trình thực nghiệm và có sự lặp lại cao, nghĩa là ta phải huấn luyện rất nhiều mô hình để tìm ra mô hình tốt nhất cho vấn đề giải quyết. Một điều khó khăn hơn là việc học sâu không hoạt động tốt cho chế độ dữ liệu lớn, việc huấn luyện để có mô hình trên tập dữ liệu lớn là rất chậm và tốn kém. Một trong những thuật toán tối ưu hóa tốc độ huấn luyện mô hình là giảm độ dốc theo lô nhỏ (mini-batch gradient descent).

#### **❖ Giảm độ dốc theo lô nhỏ**

Thay vì tính toán độ dốc trên toàn bộ tập huấn luyện, ta có thể thay thế bằng một tập con đặc trưng của tập huấn luyện – một lô nhỏ (mini-batch). Lấy mẫu ngẫu nhiên (không thay thế) một tập con của tập huấn luyện  $X_{mb} \in X$ , sao cho:

$$\Delta w_{ij} = -\alpha \frac{1}{|X_{mb}|} \sum_{\{n | x^n \in X_{mb}\}} \frac{\partial E^n}{\partial w_{ij}}$$

Trong đó kích thước lô nhỏ  $X_{mb}$  phải đủ ý nghĩa để thể hiện số liệu thống kê của phân phối tập huấn luyện. Với kích thước lô nhỏ bằng 1, ta có thuật toán Stochastic Gradient Descent, nhưng thường thuật toán này khá nhiễu và không đại diện được cho tập dữ liệu chung.

#### ❖ Giảm độ dốc theo theo lô (Batch Gradient Descent)

Trong thuật toán độ giảm dốc được thực hiện trên toàn bộ tập huấn luyện cùng một thời điểm:

$$\Delta w_{ij} = -\alpha \frac{1}{N} \sum_{n=0}^N \frac{\partial E^n}{\partial w_{ij}}$$

Trong đó:

- $N$ : số lượng mẫu huấn luyện trong  $X$ .
- $\alpha$ : tỉ lệ học (learning rate).
- $\Delta w_{ij}$ : giá trị delta cho mỗi lần cập nhật trọng số.

Ở đây, chúng ta huấn luyện với kích thước là toàn bộ tập huấn luyện là rất tốn kém và chậm vì nó yêu cầu thực hiện chuyển tiếp qua tất cả các mẫu đào tạo cho mỗi lần cập nhật trọng số.

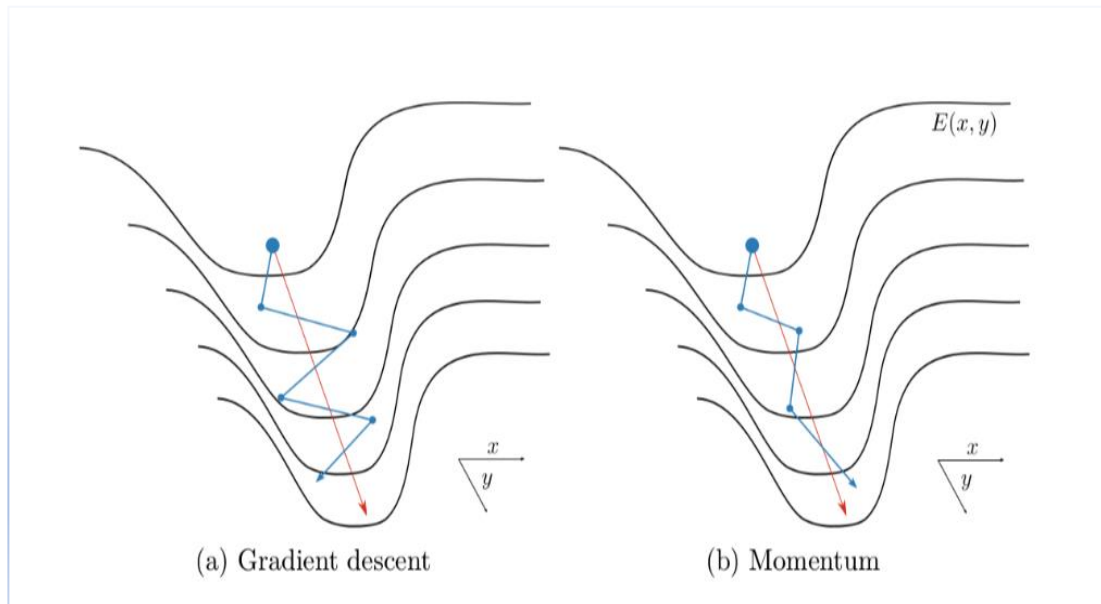
#### **2.1.2.5.2 Phương pháp giảm độ dốc với động lượng (Momentum)**

Lý do cơ bản của của tỉ lệ học (learning rate) và chính sách học có ảnh hưởng lớn là do giảm độ dốc là một phương pháp tối ưu hoá bậc nhất và chỉ xem xét các đạo hàm riêng bậc một.

$$\nabla E(x, y) = \left( \frac{\partial E}{\partial x}, \frac{\partial E}{\partial y} \right)$$

Độ dốc cho biết hướng tăng tối đa tại một điểm cho trước trên bề mặt hàm lỗi, nhưng nó không cho bất kỳ thông tin nào về độ cong của bề mặt tại điểm đó. Độ cong của bề mặt được mô tả bởi các đạo hàm bậc hai như đạo hàm riêng bậc hai. Các đạo hàm này cung cấp thông tin quan trọng về độ cong của bề mặt hàm lỗi  $E$ . Ví dụ trong hình 2.11 bề mặt hàm lỗi có hình ellipse, gây ra vấn đề khi chúng ta chỉ xem xét hướng giảm tối đa. Ví dụ

điển hình về bề mặt hàm lỗi bệnh lý cho các hàm bậc nhất là một bề mặt trông giống như một thung lũng hẹp, được thể hiện ở hình 2.12a. Với một khởi tạo bên ngoài đáy thung lũng, độ dốc sẽ nảy dọc theo các bức tường của thung lũng dẫn đến sự hội tụ học rất chậm. Hình 2.12 minh họa vấn đề độ cong bệnh lý của bề mặt hàm lỗi  $E(x, y)$ , thể hiện một thung lũng hẹp và đường tối ưu từ điểm xuất phát đến điểm cực tiểu được vẽ bằng mũi tên màu đỏ. Trong bề mặt hàm lỗi như vậy, các hàm bậc nhất không thể dùng thông tin được cung cấp bởi Hessian trên bề mặt cong để tránh nảy dọc theo các bức tường của thung lũng và làm chậm độ giảm dốc.



Hình 2.12: Minh họa vấn đề bề mặt độ cong bệnh lý và cách tiếp cận của độ giảm dốc và độ giảm dốc với động lượng (Nguồn: [1])

Đối với các bề mặt hoạt động tốt, trong đó tỉ lệ các tham số là tương tự nhau, các lưu vực thu hút xung quanh cực tiểu có dạng hình tròn và do đó tránh được vấn đề này, vì các độ dốc bậc một sẽ chỉ gần như trực tiếp tại cực tiểu cho bất kỳ vị trí nào trên bề mặt hàm lỗi.

Độ dốc giảm dần với động lượng (Gradient descent with momentum) là một thuật toán cải tiến và học nhanh hơn so với thuật toán độ giảm dốc tiêu

chuẩn, đây là một cách để giảm thiểu ảnh hưởng độ cong bệnh lý tới độ giảm dốc, điều này cũng giúp làm thay đổi độ dốc.

Trong động lượng, độ dốc trên nhiều lần lặp được tích lũy thành một vận tốc độ giảm dốc (velocity gradient),

$$v_{dW} = \beta v_{dW} + (1 - \beta) dW$$

$$v_{db} = \beta v_{db} + (1 - \beta) db$$

$$W = W - \alpha v_{dW}$$

$$b = b - \alpha v_{db}$$

Trong đó:

- $\beta$ : có giá trị phổ biến là 0.9

Động lượng có khả năng lưu trữ một số thông tin về độ dốc của các lần lặp trước và sử dụng thông tin này để làm giảm hiệu ứng của độ dốc mới trên hướng tìm kiếm được minh họa ở hình 2.12. Đối với các bề mặt lỗi có độ cong bệnh lý, điều này có thể tăng tốc đáng kể việc học.

### 2.1.3 Các phương pháp huấn luyện mạng nơ-ron hiện đại

#### 2.1.3.1 Hàm kích hoạt đơn vị tuyến tính chỉnh lưu (Rectified Linear Unit)

Một phần không thể thiếu của bất kỳ mạng nơ-ron nào là hàm kích hoạt phi tuyến tính. Trong lịch sử, mạng nơ-ron đã từng sử dụng chức năng kích hoạt dạng sigmoid. Tuy nhiên một vấn đề lớn với các hàm kích hoạt dạng sigmoid là độ dốc bên ngoài một vùng tương đối hẹp trên miền hàm là một số rất nhỏ. Khi huấn luyện với lan truyền ngược (backpropagation) điều này có nghĩa là hầu hết độ dốc có độ lớn rất nhỏ và việc huấn luyện có thể sẽ tốn

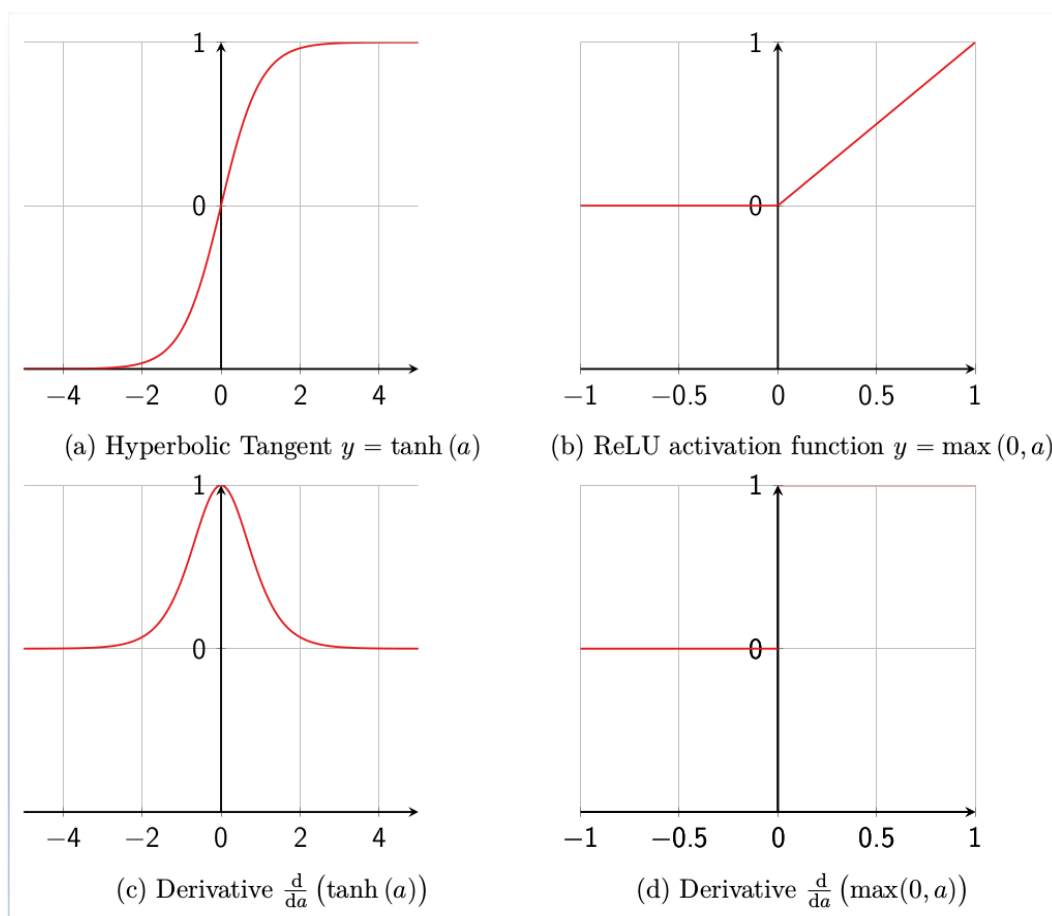


một thời gian rất dài, hoặc thậm chí bị đình trệ hoàn toàn - tình huống này được gọi là độ dốc biến mất (vanishing gradient).

ReLU được đề xuất như một giải pháp, đầu tiên cho các máy Boltzmann bị hạn chế (Nair và Geoffrey E. Hinton, 2010), sau đó là cho các mạng nơ-ron ((Glorot và Y. Bengio, 2010), trong đó về mặt thực nghiệm, nó được chứng minh là cho phép đào tạo dễ dàng hơn với lan truyền ngược.

$$z^{[i]} = W^{[i]}x + b^{[i]}$$

$$a = \max(0, z)$$



Hình 2.13: Minh họa các hàm kích hoạt thường dùng trong mạng nơ-ron và đạo hàm tương ứng của nó. (Nguồn: [1])

ReLU không thể hiện độ bão hòa như các hàm dạng sigmoid, nó luôn cho độ dốc có giá trị 0 hoặc 1, được minh họa ở hình 2.10. Trong thực tế, điều

này có thể làm tăng tốc độ đào tạo thậm chí cho phép các mạng mà không thể đào tạo thực tế với hàm kích hoạt dạng sigmoid chẳng hạn như mạng học sâu Krizhevsky, Sutskever và Geoffrey E. Hinton (2012).

### 2.1.3.2 Phương pháp chuẩn hoá hàng loạt (Batch Normalization)

Sergey Ioffe và Christian Szegedy đã giới thiệu phương pháp chuẩn hoá hàng loạt (Batch Normalization) vào năm 2015. Ý tưởng được trình bày trong bài báo *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift* [2].

Ý tưởng chính của phương pháp này là thay vì chỉ chuẩn hoá các đầu vào của mạng, chúng ta chuẩn hoá các đầu vào của các lớp trong mạng nơ-ron. Nó được gọi là chuẩn hoá hàng loạt vì trong quá trình đào tạo, phương pháp này chuẩn hoá kích hoạt của lớp trước cho mỗi lô, tức là áp dụng phép biến đổi duy trì kích hoạt trung bình gần bằng 0 và độ lệch chuẩn kích hoạt gần bằng 1.

Thứ nhất có thể hiểu rằng **Non zero mean** là hiện tượng dữ liệu không phân bố quanh giá trị 0, mà dữ liệu có phần nhiều giá trị lớn hơn không, hoặc nhỏ hơn không. Kết hợp với vấn đề high variance khiến dữ liệu trở nên có nhiều thành phần rất lớn hoặc rất nhỏ. Vấn đề này rất phổ biến khi training các mạng nơ ron với số layer sâu. Việc feature không phân phối trong những khoảng ổn định (giá trị to nhỏ thất thường) sẽ có ảnh hưởng đến quá trình tối ưu của mạng. Vì như chúng ta đã biết việc tối ưu một mạng nơ ron sẽ cần phải sử dụng đến tính toán đạo hàm. Giả sử như một công thức tính layer đơn giản là  $y = (Wx + b)$  thì đạo hàm của  $y$  theo  $w$  có dạng:  $dy = dWx$ . Như vậy giá trị  $xx$  ảnh hưởng trực tiếp đến giá trị của đạo hàm (tất nhiên khái niệm gradient trong các mô hình mạng nơ ron không thể đơn giản như vậy tuy nhiên về mặt lý thuyết thì  $xx$  sẽ có ảnh hưởng đến đạo hàm). Do đó nếu  $xx$  mang các giá trị thay đổi không ổn định dẫn đến đạo hàm sẽ có thể

bị quá lớn, hoặc quá nhỏ dẫn đến việc learning model không được ổn định. Và điều đó cũng đồng nghĩa với việc chúng ta có thể sử dụng các learning rate cao hơn trong quá trình training khi sử dụng Batch Normalization.

Batch Normalization có thể giúp chúng ta tránh được hiện tượng giá trị của  $x$  rơi vào khoảng bão hòa sau khi đi qua các hàm kích hoạt phi tuyến. Vậy nên nó đảm bảo rằng không có sự kích hoạt nào bị vượt quá cao hoặc quá thấp. Điều này giúp cho các weights mà khi không dùng Batch Normalization có thể sẽ không bao giờ được học thì nay lại được học bình thường. Điều này giúp chúng ta làm giảm đi sự phụ thuộc vào giá trị khởi tạo của các tham số.

Batch Normalization còn có vai trò như một dạng của **regularization** giúp cho việc giảm thiểu overfitting. Sử dụng batch normalization, chúng ta sẽ không cần phải sử dụng quá nhiều dropout và điều này rất có ý nghĩa vì chúng ta sẽ không cần phải lo lắng vì bị mất quá nhiều thông tin khi dropout weights của mạng. Tuy nhiên vẫn nên sử dụng kết hợp cả hai kỹ thuật này.

### 2.1.3.3 Phương pháp cắt giảm (Dropout)

Geoffrey E. Hinton, Srivastava, et al. (2012a) và Srivastava et al. (2014) đã giới thiệu phương pháp cắt giảm (dropout), một phương pháp ngăn chặn vượt mức (overfit) trong các mạng lớn khi đào tạo. Ý tưởng chính là: Trong quá trình huấn luyện ta loại bỏ một tập các nút nơ-ron, được lấy mẫu ngẫu nhiên từ mỗi lớp với xác suất cố định  $p$ .

Cơ chế ảnh hưởng của phương pháp cắt giảm được giải thích theo nhiều cách khác nhau, nhưng đáng chú ý nhất là các cách giải thích được đưa ra bởi Geoffrey E. Hinton, Srivastava, et al. (2012a) và Srivastava et al. (2014).

Giải thích của Geoffrey E. Hinton, Srivastava, et al. (2012a) là phương pháp cắt giảm là một hình thức chính quy hóa bằng độ ồn, ngăn chặn sự thích nghi của các tế bào nơ-ron. Giải thích chính được đưa ra bởi Srivastava et al. (2014) rằng phương pháp cắt giảm là một hình thức tích hợp mô hình, tính trung bình trên một số lượng lớn các kiến trúc mô hình "mỏng hơn" ngẫu nhiên tại thời điểm đào tạo để cải thiện khái quát hóa. Tuy nhiên, việc tính trung bình trên tất cả các mô hình được xem xét trong suốt quá trình đào tạo là cực kỳ tốn kém, bởi vì số lượng mô hình có thể có tăng theo cấp số mũ.

#### **2.1.4 Các kiến trúc mạng nơ-ron hồi quy**

Một danh sách nghiên cứu đầy đủ của mọi kiến trúc học sâu DNN là điều không khả thi và nằm ngoài phạm vi của luận án này, tuy nhiên ở đây, nhóm sinh viên đã nỗ lực khái quát về những kiến trúc nơ-ron hồi quy nổi bật trong những năm gần đây đồng thời truyền cảm hứng cho việc thiết lập kiến trúc hệ thống dịch trong các chương sau của nhóm.

##### **2.1.4.1 Mạng nơ-ron hồi quy (RNN – Recurrent Neural Network)**

Mạng nơ-ron hồi quy (RNN - Recurrent Neural Network) là mạng thần kinh lan truyền tới (Feedforward Neural Networks) [3] được tăng cường bằng cách bao gồm các cạnh mà kéo dài các bước thời gian liên tiếp, đưa ra khái niệm về thời gian cho mô hình. Giống như mạng lan truyền tới, mạng RNN không có chu trình giữa các cạnh thông thường. Tuy nhiên, các cạnh mà kết nối các bước thời gian liên tiếp, được gọi là cạnh hồi quy, có thể hình thành chu trình có độ dài bằng một, tự kết nối từ nột nút(node) tới chính nó theo thời gian. Tại thời điểm  $t$ , nút với cạnh hồi quy nhận đầu vào từ điểm dữ liệu hiện tại  $x^{(t)}$  và từ các giá trị nút ẩn  $h^{(t-1)}$  trong trạng thái trước đó của mạng. Đầu ra  $y^{(t)}$  cho mỗi thời điểm  $t$  được tính bằng các giá trị nút ẩn  $h^{(t)}$  tại thời điểm  $t$ . Đầu vào  $x^{(t-1)}$  tại thời điểm  $t-1$  có thể ảnh hưởng đến giá trị đầu ra  $y^{(t)}$  tại thời điểm  $t$  và sau đó, bằng các kết nối hồi quy.

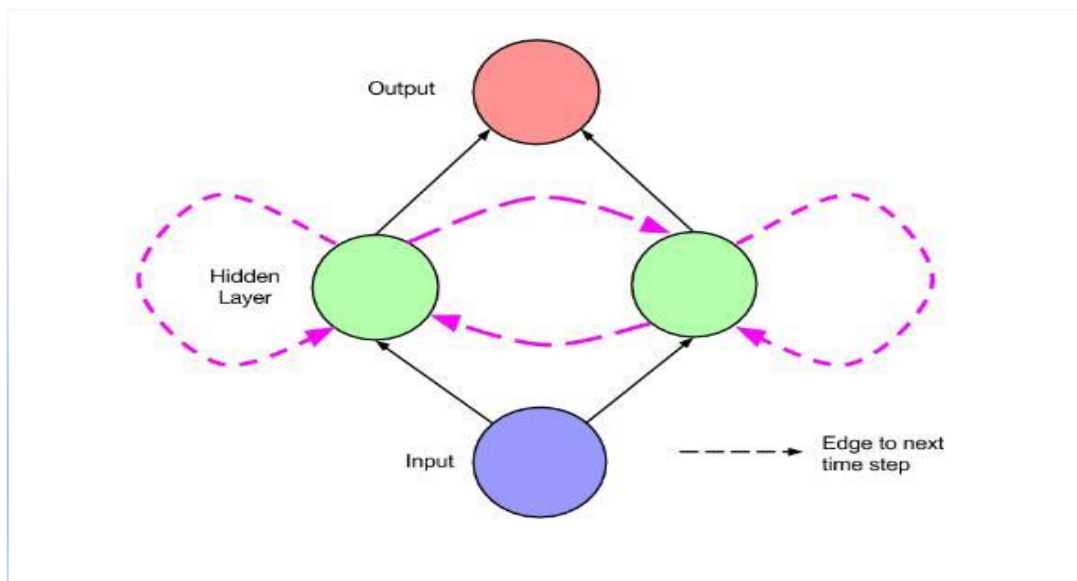
Các tính toán cần thiết tại mỗi bước thời gian trên đường là truyền tới của mạng RNN đơn giản (hình 2.11), được thể hiện như sau:

$$h(t) = \sigma(W^{hx} + W^{hh} h^{(t-1)} + b_h)$$

$$y^{(t)} = \text{softmax}(W^{yh} h^{(t)} + b_y)$$

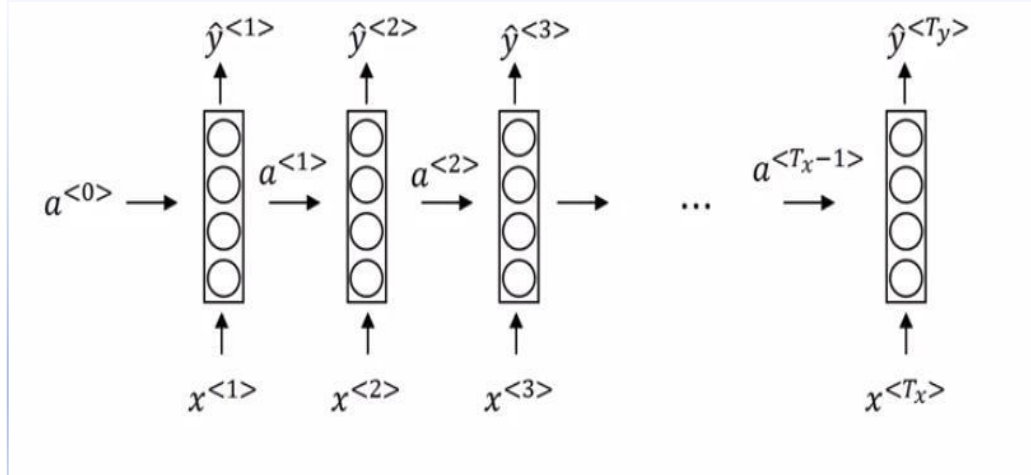
Trong đó:

- $W^{hx}$  là ma trận trọng số thông thường giữa đầu vào và lớp ẩn
- $W^{hh}$  là ma trận trọng số hồi quy giữa lớp ẩn và chính nó ở các bước thời gian liên kế
- Các vector  $b_h$  và  $b_y$  là các tham số sai lệch



Hình 2.14: Mạng RNN đơn giản (Nguồn: [3])

Một biểu diễn dễ hiểu cho hình 2.14 được thể hiện ở hình 2.15, trong đó các bước thời gian được mở ra. Với hình ảnh này, mạng được hiểu không phải là chu trình, mà là một mạng học sâu với mỗi lớp tương ứng với mỗi bước thời gian, được chia sẻ trọng số qua các lớp.



Hình 2.15: Minh họa mạng RNN được mở ra từ hình 2.14

(Nguồn: Coursera Sequence Models)

Bây giờ lấy ví dụ, xét bài toán: Xác định chữ nào là một phần của tên người trong một câu.

Đọc một câu từ trái sang phải, đánh số mỗi từ trong câu là  $x_i$  ( $1 \leq i \leq N$ ). Tiến trình nạp từ thứ nhất  $x_1$  vào một lớp ẩn mạng nơ-ron, lớp này sẽ dự đoán kết quả  $y^{<1>}$  xem  $x_1$  có phải là một phần của tên người hay không. Xét từ thứ hai  $x_2$  thay vì chỉ dự đoán  $y^{<2>}$  từ  $x_2$ , thì lớp này nhận thêm giá trị kích hoạt  $a^{<1>}$  từ bước 1. Các bước sau được thực hiện tương tự, cho đến khi kết thúc câu. Thông thường ở bước đầu tiên cũng được truyền thêm vào giá trị kích hoạt với  $a^{<0>}$  (là một vector 0). Hình 2.15 minh họa mạng RNN lan truyền tới với giá trị  $a^{<t>}$ ,  $y^{<t>}$  được tính theo công thức như sau:

$$a^{<t>} = g(W_{aa} a^{<t-1>} + W_{ax} x^{<t>} + b_a)$$

$$y^{<t>} = g(W_{ya} a^{<t>} + b_y)$$

RNN duyệt dữ liệu từ trái sang phải và tham số dùng cho mỗi bước là được chia sẻ. Vì vậy, khi dự đoán kết quả  $y^{<3>}$ , RNN không chỉ sử dụng đầu vào  $x_3$  mà còn sử dụng thông tin từ  $x_1$  và  $x_2$ .

Xét hai trường hợp đầu vào sau:

Trường hợp một:

(1) Anh ấy nói "*Teddy Roosevelt là một tổng thống tuyệt vời*".

(2) Anh ấy nói "*Teddy là loại gấu bông được mua nhiều nhất ở cửa hàng này*".

Trường hợp hai:

*Thời thơ ấu, tôi thường nghe bố tôi nhắc tới một người anh hùng, ... bố tôi đang nhắc tới Phan Đình Giót.*

Ở trường hợp một, "*Teddy*" là một phần tên người trong câu (1), còn câu (2) thì không phải. Như vậy, một điểm yếu của mạng RNN là chỉ dùng thông tin từ các bước phía trước trong chuỗi để thực hiện dự đoán kết quả, thông tin dùng để dự đoán này là không đủ. Để khắc phục nhược điểm này, mạng nơ-ron hồi quy hai chiều (BiRNN - Bidirectional recurrent neural network) ra đời.

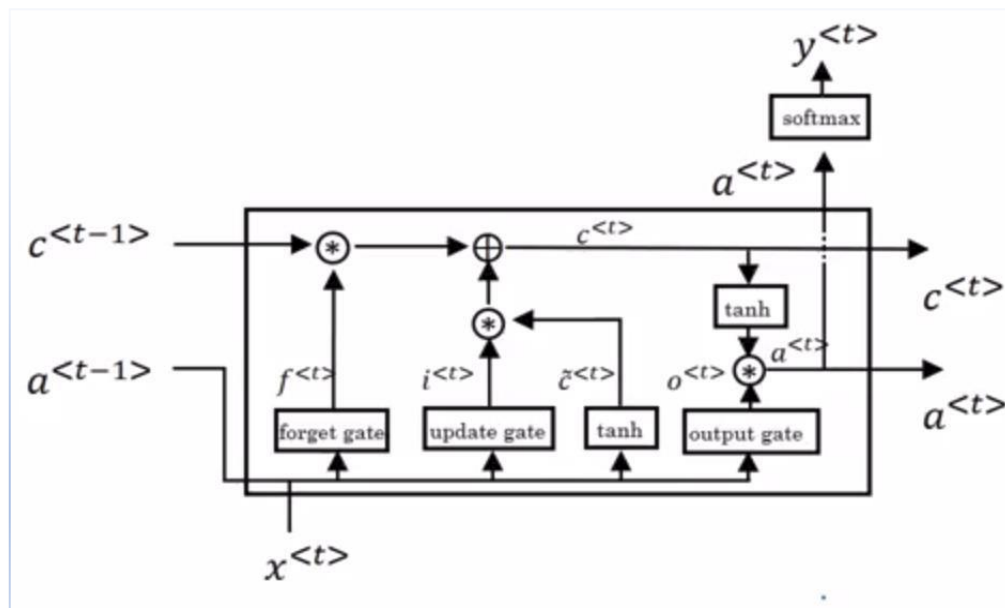
Còn ở trường hợp hai, cụm từ "*Phan Đình Giót*" có được dự đoán là tên của người hay không phụ thuộc thông tin được mang đến từ cụm "*người anh hùng*". Nhưng mạng RNN bị hạn chế trong duy trì phụ thuộc tầm xa, nghĩa là nếu chuỗi đầu vào đủ dài, mạng đủ sâu, thì RNN khó khăn trong việc mang thông tin từ các bước trước tới các bước sau, thậm chí RNN có thể bỏ qua thông tin quan trọng đến từ những bước đầu tiên. Đây là vấn đề biến mất độ dốc (vanishing gradient).

#### 2.1.4.2 Mạng bộ nhớ dài ngắn (Long Short Term Memory - LSTM)

Như đã trình bày ở mục 2.4.1, mạng RNN bị vấn đề độ dốc biến mất (vanishing gradient). Bản chất của mô hình mô hình RNN có nhiều ảnh hưởng cục bộ, nghĩa là đầu ra  $y^{<3>}$  sẽ bị ảnh hưởng bởi các giá trị gần  $y^{<3>}$

như  $x^{<1>}$ ,  $x^{<2>}$ ,  $x^{<3>}$ , đầu ra vị trí cuối chuỗi  $y^{<T>}$  sẽ bị ảnh hưởng bởi các giá trị gần T. Điều này gây nên khó khăn nếu muốn đầu ra  $y^{<T>}$  bị ảnh hưởng mạnh mẽ bởi những giá trị ở vị trí đầu chuỗi (hình 2.15).

Mạng bộ nhớ dài-ngắn (Long Short-Term Memory - LSTM) được tạo ra để giải quyết vấn đề duy trì phụ thuộc tầm xa. Mạng LSTM giống với mạng RNN tiêu chuẩn, nhưng với mỗi nút gốc trong lớp ẩn sẽ được thay thế bằng một ô nhớ (memory cell) và sử dụng thêm ba cổng riêng biệt là cổng quên (forget gate), cổng vào (input gate), cổng ra (output gate) để điều chỉnh luồng thông tin trong một ô LSTM.



Hình 2.16: Minh họa một đơn vị LSTM

(Nguồn: Coursera Sequence Models)

Cơ chế điều chỉnh luồng thông tin và cập nhật trạng thái ô nhớ được thể hiện như sau:

#### ❖ Cổng quên (forget gate)



Cổng này quyết định thông tin nào là quan trọng để giữ lại từ bước phía trước. Thông tin từ lớp ẩn trước kết hợp với thông tin đầu vào hiện tại, sau đó được truyền qua hàm sigmoid, giá trị được chuyển đổi vào đường biên từ 0 đến 1. Giá trị càng gần 0 thì bị bỏ qua, càng gần 1 thì được giữ lại. Công thức tính toán giá trị cổng quên:

$$\Gamma_f = \sigma(W_f [a^{<t-1>}, x^{<t>}] + b_f)$$

#### ❖ Cổng vào (input gate)

Cổng này quyết định thông tin nào là quan trọng để thêm vào từ bước hiện tại. Kết hợp thông tin từ lớp ẩn trước và thông tin đầu vào hiện tại cho truyền qua hàm sigmoid. Hàm này sẽ quyết định giá trị nào sẽ được cập nhật, bằng cách chuyển giá trị về khoảng biên từ 0 đến 1, gần 0 là không quan trọng, gần 1 là quan trọng. Đồng thời cũng truyền thông tin từ lớp ẩn trước và thông tin đầu vào hiện tại qua hàm tanh, giá trị được chuyển về khoảng biên từ -1 đến 1. Sau đó nhân đầu ra sigmoid với đầu ra tanh, để quyết định thông tin nào quan trọng để giữ lại từ đầu ra hàm tanh. Công thức tính giá trị cổng vào:

$$\Gamma_i = \sigma(W_i [a^{<t-1>}, x^{<t>}] + b_i)$$

#### ❖ Trạng thái ô (cell status)

Bây giờ, ta đã đủ thông tin để tính toán trạng thái mới của ô. Trạng thái mới được tính toán bằng trạng thái ô đang xét nhân với vector đầu ra của cổng quên để loại bỏ một số giá trị của ô trạng thái nếu nhân với giá trị gần 0, sau đó cộng giá trị này với đầu ra của cổng vào. Công thức để tính trạng thái ô hiện tại:

$$\tilde{c}^{<t>} = \tanh(W_c [a^{<t-1>}, x^{<t>}] + b_c)$$

$$c^{<t>} = \Gamma_f * \tilde{c}^{<t>} + \Gamma_i * c^{<t-1>}$$

#### ❖ Cổng ra (output gate)

Cổng này quyết định trạng thái ẩn truyền cho bước tiếp theo là gì. Truyền trạng thái lớp ẩn bước trước và giá trị đầu vào hiện tại vào hàm sigmoid. Sau đó đưa trạng thái mới của ô đã cập nhật qua hàm tanh, nhân giá trị đầu ra của hàm sigmoid và giá trị đầu ra của hàm tanh để quyết định những thông tin trạng thái ẩn nào nên mang theo, kết quả này và trạng thái mới của ô được truyền tới bước tiếp theo. Công thức tính giá trị cổng ra:

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$$

#### 2.1.4.3 Mạng nơ-ron hồi quy hai chiều (Bidirectional Recurrent Neural Network – BiRNN)

Cùng với mạng LSTM, một trong những kiến trúc RNN được sử dụng nhiều nhất là mạng nơ-ron hồi quy hai chiều (BiRNN - Bidirectional recurrent neural network), khắc phục nhược điểm chỉ nhận thông tin từ các bước thời gian phía trước để dự đoán kết quả bước hiện tại.

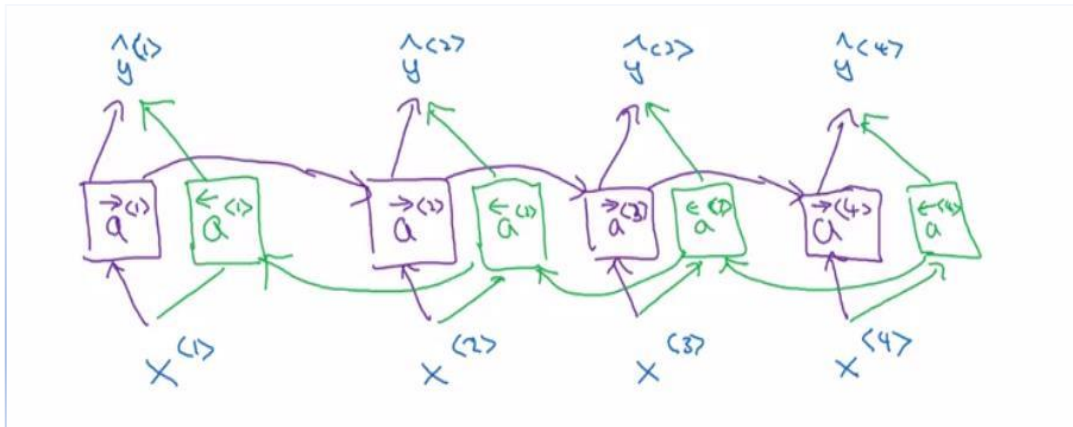
Trong kiến trúc này, có hai lớp nút ẩn. Cả hai lớp ẩn đều được kết nối với đầu vào và đầu ra. Hai lớp này được phân biệt ở chỗ, lớp đầu tiên có các kết nối hồi quy đến từ các bước thời gian trước, trong khi ở lớp thứ hai, hướng hồi quy của các kết nối bị đảo ngược, nghĩa là truyền giá trị kích hoạt ngược theo chiều chuỗi (hình 2.17). Ba phương trình sau mô tả một BiRNN.

$$h^{(t)} = \sigma(W^{hx}x^{(t)} + W^{hh}h^{(t-1)} + b_h)$$

$$z^{(t)} = \sigma(W^{zx}x^{(t)} + W^{zz}z^{(t+1)} + b_z)$$

$$y^{(t)} = \text{softmax}(W^{yh}h^{(t)} + W^{yz}z^{(t)} + b_y)$$

Trong đó  $h^{(t)}$  và  $z^{(t)}$  là các giá trị của lớp ẩn theo hai hướng lan truyền tới và lan truyền ngược tương ứng.

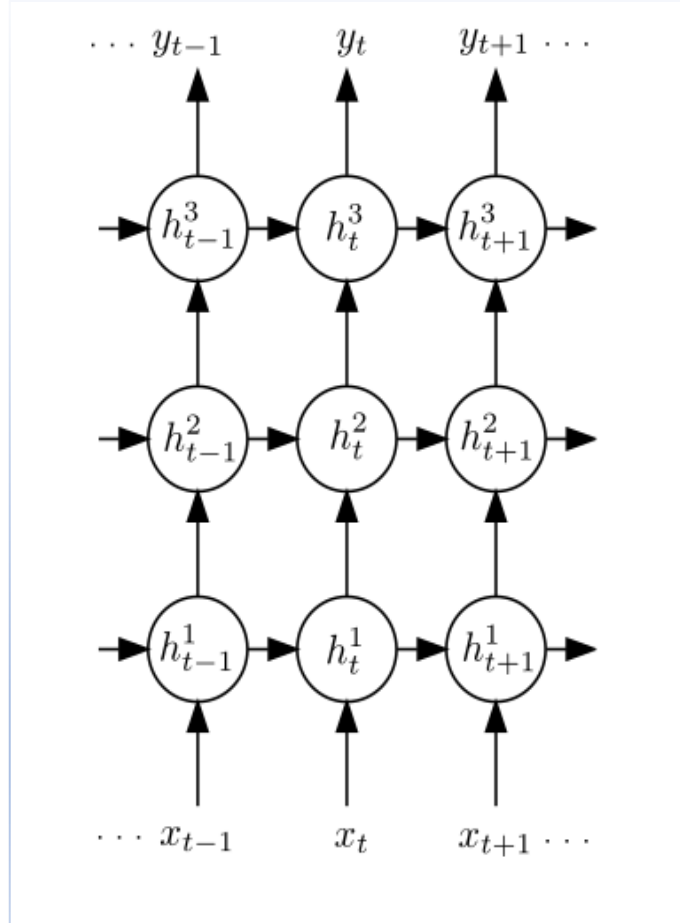


Hình 2.17: Minh họa mạng BiRNN (Nguồn: Coursera Sequence Models)

Tuy nhiên BiRNN có một hạn chế là không thể chạy liên tục, nó yêu cầu một điểm cuối cố định cho cả tương lai và quá khứ, vì BiRNN vẫn mang bản chất của một mạng RNN cơ bản. Trên thực tế LSTM và BiRNN khá tương thích với nhau, vì LSTM giới thiệu một đơn vị cơ bản mới để tạo thành một lớp ẩn, còn BiRNN liên quan đến việc kết nối giữa các lớp ẩn, bất kể chúng được cấu thành từ loại đơn vị nào. Sự kết hợp này, đưa đến khái niệm BiLSTM. Dựa vào thực nghiệm [3], cách tiếp cận này đã đạt được kết quả hiện đại về nhận dạng chữ viết tay và phân loại âm vị.

#### 2.1.4.4 Mạng nơ-ron hồi quy sâu (Deep Recurrent Neural Network – Deep RNN)

Một yếu tố quan trọng cho sự thành công gần đây của các hệ thống lai là sử dụng kiến trúc học sâu. Deep RNN có thể được tạo bằng cách xếp chồng nhiều lớp ẩn RNN lên nhau, với đầu ra của một lớp trở thành đầu vào của lớp tiếp theo, được biểu diễn như hình 2.18.



Hình 2.18: Mạng nơ-ron hồi quy học sâu (Nguồn: [4])

## 2.2 MÔ HÌNH DỊCH MÁY:

Các mô hình dịch máy hiện nay thường được chia làm ba thành phần chính: Nhúng từ (word embedding), bộ mã hoá (encoder) và bộ giải mã (decoder). Tuy nhiên tùy cách tiếp cận mà giải pháp cho mỗi mô hình dịch máy sẽ được tùy chỉnh như thêm các lớp tích chập hay thêm cơ chế chú ý để đưa ra một hệ thống phù hợp. Sau đây, nhóm sinh viên sẽ trình bày về mô hình dịch máy điển hình là: mô hình dịch máy dựa trên mô hình nơ-ron hồi quy (RNN) kết hợp với từ nhúng (sử dụng word2vec), mạng nơ-ron hồi quy hai chiều (Bi-RNN) tại bộ mã hoá (encoder) và cơ chế chú ý (Attention).

### 2.2.1 Giới thiệu và đặt vấn đề

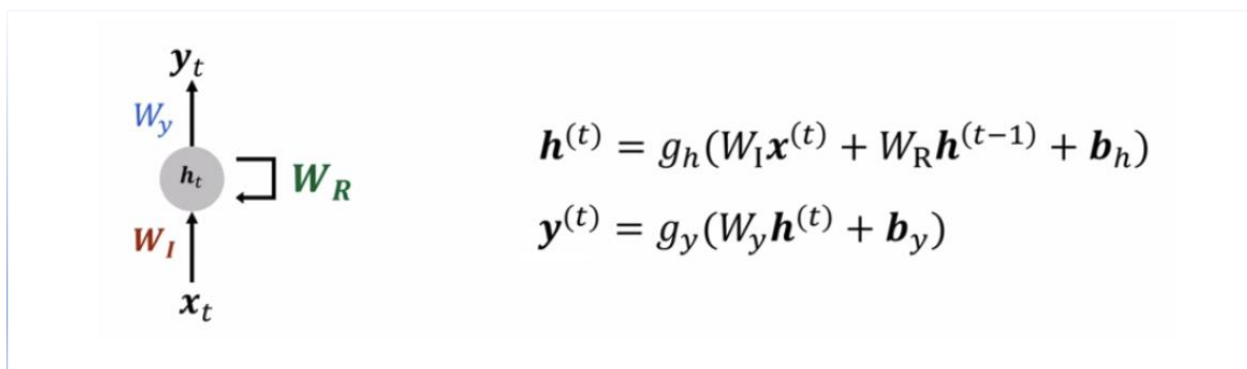
Mạng nơ-ron là mô hình học mạnh mẽ, đã đạt được những kết quả vượt bậc trong nhiều tác vụ học máy. Những tiến bộ gần đây về thuật toán và phần cứng đã giúp con người có thể huấn luyện mạng nơ-ron hỗ trợ cho các nhiệm vụ mà trước đây đòi hỏi trình độ chuyên môn đáng kể từ con người. So với các cách tiếp cận truyền thống, mạng nơ-ron đòi hỏi ít sự nỗ lực hơn từ con người và mang lại kết quả cao hơn. Để đạt được kết quả khả quan này là nhờ sự phong phú đa dạng của dữ liệu ngày nay.

Mạng neural truy hồi (Recurrent Neural Network, viết tắt là RNN) được phát minh bởi John Hopfield năm 1982 [5]. Trong khoảng 5-6 năm gần đây, RNN được ứng dụng rộng rãi trong ngành NLP và thu được những thành tựu lớn. Mạng RNN mô hình hóa được bản chất của dữ liệu trong NLP (có đặc tính chuỗi và các thành phần như từ, cụm từ trong dữ liệu phụ thuộc lẫn nhau). Có thể nói việc áp dụng mạng RNN là một bước đột phá trong ngành NLP.

Trong mô hình mạng nơ-ron thông thường (Feed forward network), chúng ta coi input data là các dữ liệu độc lập, không có mối liên hệ với nhau. Tuy nhiên, trong ngôn ngữ tự nhiên thì mối liên hệ giữa các từ và ngữ cảnh đóng một vai trò quan trọng, quyết định ý nghĩa của câu văn. Do đó việc áp dụng một hình mạng nơ-ron thông thường vào các bài toán xử lý ngôn ngữ tự nhiên thường không đạt kết quả mong muốn.

Để khắc phục nhược điểm này, chúng ta sử dụng mô hình RNN (Recurrent Neural Network). RNN coi dữ liệu đầu vào là một chuỗi (sequence) liên tục, nối tiếp nhau theo thứ tự thời gian. Ví dụ như một đoạn text có thể được coi là một chuỗi các từ vựng(words) hoặc là một chuỗi các ký tự (character). Tại thời điểm  $t$ , với dữ liệu đầu vào  $x_t$  ta có kết quả output là  $y_t$ . Tuy nhiên, khác với mạng Feed forward network,  $y_t$  lại được sử dụng là input để tính kết quả output cho thời điểm  $(t+1)$ . Điều này cho phép RNN có thể lưu trữ và truyền thông tin đến

thời điểm tiếp theo. Mô hình hoạt động của RNN có thể được mô tả trong hình dưới đây (thông thường hàm activation function  $g_h$  được sử dụng là tanh còn  $g_y$  có thể là sigmoid hoặc softmax function tùy thuộc vào từng bài toán cụ thể).



Hình 2.19: Mạng Recurrent Neural Network (Nguồn: [3])

Đặt vấn đề: Tại sao không sử dụng mô hình nơ-ron hồi quy cơ bản như trên để xây dựng mô hình dịch máy.

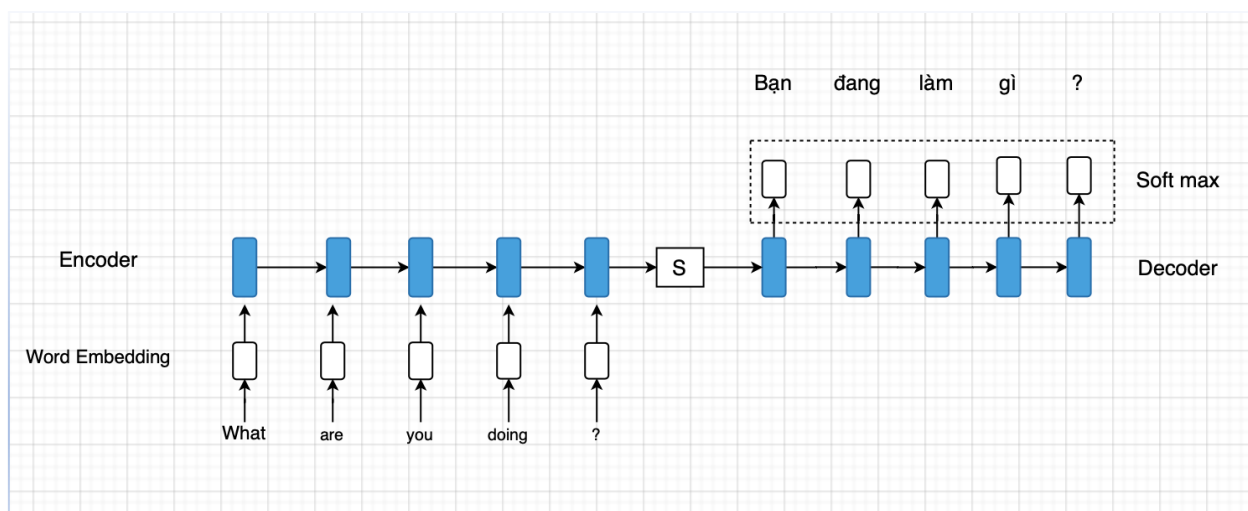
Chúng ta có thể hiểu một cách đơn giản rằng RNN là một mô hình mạng Neuron có bộ nhớ (memory) để lưu trữ thông tin của phần xử lý trước đó. Về mặt lý thuyết thì mạng nơ-ron hồi quy có thể xử lý và lưu trữ thông tin của một chuỗi dữ liệu với độ dài bất kỳ. Tuy nhiên trong thực tế thì nó chỉ tỏ ra hiệu quả với chuỗi dữ liệu có độ dài không quá lớn (short-term memory hay còn gọi là long-term dependency problem). Nguyên nhân của vấn đề này là do vấn đề mất độ dốc trong quá trình huấn luyện (vanishing gradient problem, độ dốc (gradient) được sử dụng để cập nhật giá trị của ma trận trọng số (weight matrix) trong mạng nơ-ron hồi quy và nó có giá trị nhỏ dần theo từng layer khi thực hiện back propagation). Khi gradient trở nên rất nhỏ (có giá trị gần bằng 0) thì giá trị của ma trận trọng số sẽ không được cập nhật thêm và do đó mạng nơ-ron sẽ dừng việc học tại lớp này. Đây cũng chính là lý do khiến cho mạng nơ-ron hồi quy không thể lưu trữ thông tin của các bước thời gian (timesteps) đầu tiên trong một chuỗi dữ liệu có độ dài lớn.

Với những hạn chế của mô hình trên, mô hình sequen to sequence sử dụng kiến trúc bộ nhớ dài ngắn (Long-Short Term Memory) đã được trình bày ở trên với cơ chế chú ý (Attention Mechanism) giúp giải quyết vấn đề trên và đưa ra một mô hình mạnh mẽ để thực hiện tác vụ dịch máy.

### **2.2.2 Mô hình dịch máy Sequence to Sequence với cơ chế chú ý (Attention Mechanism)**

Sequence to Sequence Model (Seq2seq) là một mô hình Deep Learning với mục đích tạo ra một output sequence từ một input sequence mà độ dài của 2 sequences này có thể khác nhau. Seq2seq được giới thiệu bởi nhóm nghiên cứu của Google vào năm 2014 trong bài báo *Sequence to Sequence with Neural Networks* [6]. Mặc dù mục đích ban đầu của Model này là để áp dụng trong Machine Translation, tuy nhiên hiện nay Seq2seq cũng được áp dụng nhiều trong các hệ thống khác như Speech recognition, Text summarization, Image captioning,....

Seq2seq gồm 2 phần chính là Encoder và Decoder. Cả hai thành phần này đều được hình thành từ các mạng Neural Networks, trong đó Encoder có nhiệm vụ chuyển đổi dữ liệu đầu vào (input sequence) thành một representation với lower dimension còn Decoder có nhiệm vụ tạo ra output sequence từ representation của input sequence được tạo ra ở phần Encoder.



Hình 2.20: Sequence to Sequence Model in Machine Translation

Trong luận văn, từ dữ liệu đầu vào (input sequence) là một câu dưới dạng văn bản, chúng ta sử dụng Embedding Layer để chuyển các từ này sang dạng Word Embedding rồi sử dụng Bi-directional LSTM để tạo ra một đại diện (representation) của câu đầu vào (trong hình 2.20 là S).

Decoder được tạo thành từ RNN với LSTM và sử dụng đầu ra của Encoder làm dữ liệu đầu vào để tạo ra một câu đầu ra (output sequence). Trong dịch máy chúng ta phải chọn câu văn phù hợp nhất thay vì để RNN cell tạo ra từng từ một. Thông thường việc lựa chọn output sequence được thực hiện bởi thuật toán tìm kiếm chùm tia (Beam Search).

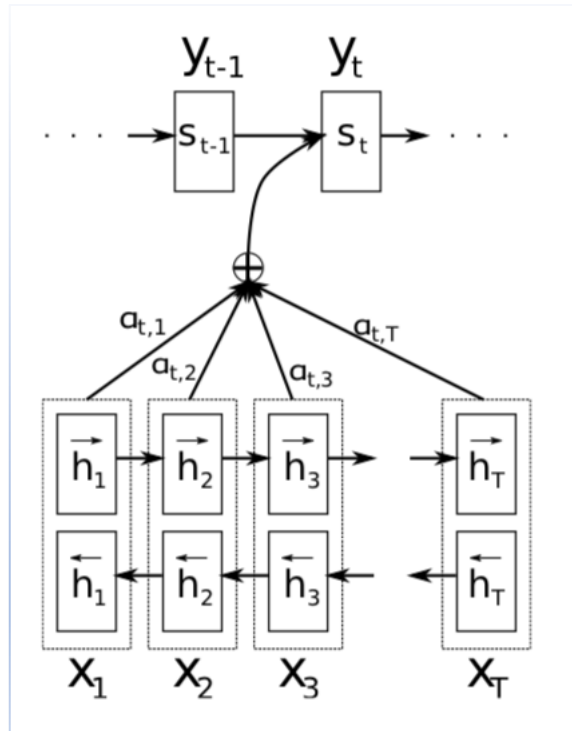
Tuy nhiên, trong thực tế việc sử dụng một vector đại diện (vector representation) thường không thể lưu trữ được toàn bộ thông tin của câu đầu vào (input sequence). Do đó, có một số phương pháp giúp tăng độ chính xác cho hệ thống này như: sử dụng nhiều lớp hơn và sử dụng mạng nơ-ron hồi quy hai chiều.



Tuy nhiên, phương pháp được sử dụng nhiều nhất và làm tăng đáng kể độ chính xác của các hệ thống là sử dụng cơ chế chú ý (Attention Mechanism). Phương pháp này được giới thiệu vào năm 2014 trong bài báo *Neural Machine Translation by Jointly Learning to Align and Translate* [7].

Nguyên tắc hoạt động chung của cơ chế chú ý (Attention Mechanism) là tại mỗi bước giải mã (Decoding Step), bộ giải mã (decoder) sẽ chỉ tập chung vào phần liên quan trong câu đầu vào thay vì toàn bộ câu đầu vào. Mức độ tập chung này được thiết lập bởi ma trận chú ý (Attention weights) như mô tả trong hình 2.21.

Như vậy, tại mỗi bước giải mã, bộ giải mã nhận ba đầu vào là: Hidden state của bước giải mã trước, đầu ra của bước phía trước và vector chú ý (Attention vector). Vector chú ý chứa ma trận chú ý (Attention weight) của từng từ trong câu đầu vào. Từ nào chứa nhiều thông tin cần thiết cho việc giải mã thì sẽ có giá trị trọng số lớn hơn và tổng các trọng số của tất cả các từ trong câu đầu vào phải bằng 1. Giá trị của ma trận chú ý này được học thông qua quá trình huấn luyện.



Hình 2.21: Mô hình sequence to sequence với cơ chế chú ý ([Nguồn: [7]])

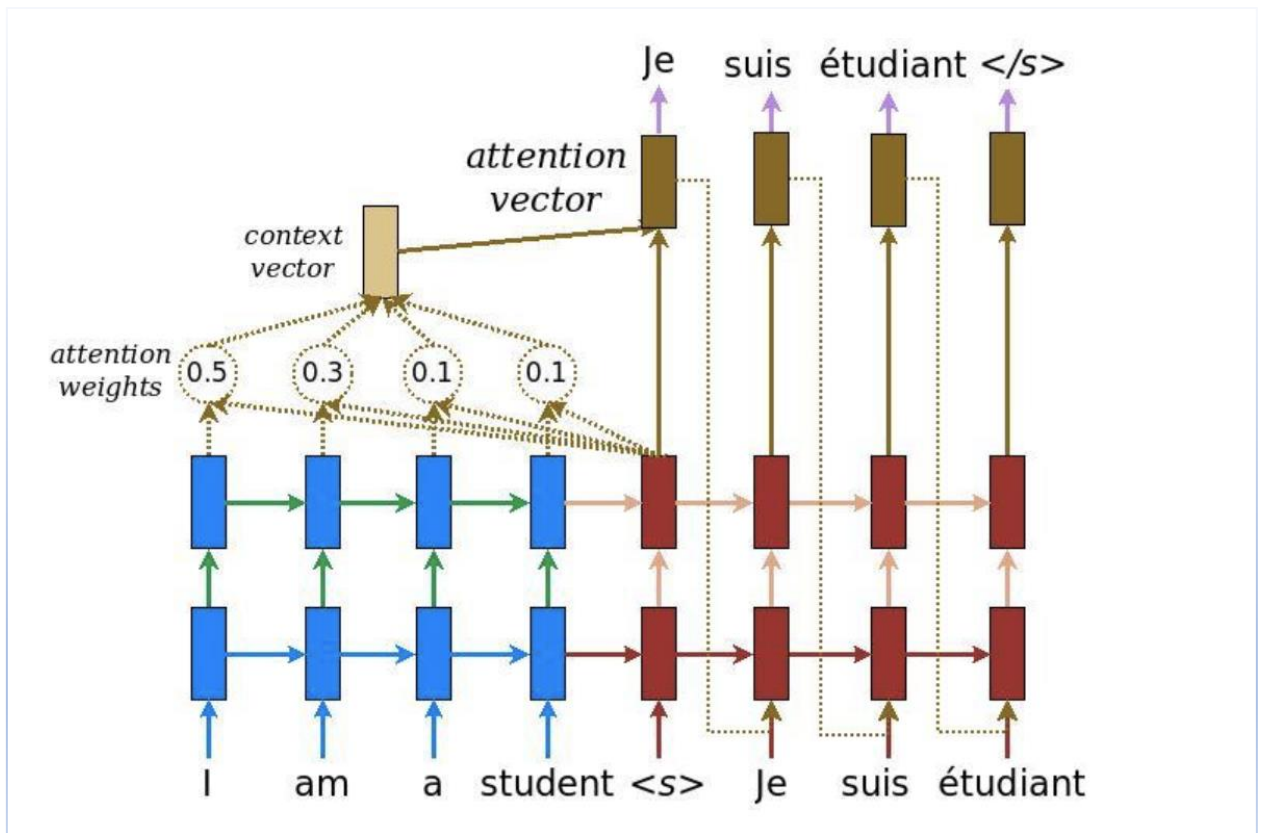
## **CHƯƠNG 3: GIẢI PHÁP ĐỀ TÀI**

### **3.1 TỔNG QUAN GIẢI PHÁP KIẾN TRÚC MÔ HÌNH**

Nhóm sinh viên sử dụng mô hình nơ-ron hồi quy (RNN) với bộ mã hoá (encoder) và bộ giải mã (decoder) hay còn được gọi với Sequence to Sequence Model – seq2seq) có ý tưởng từ bài báo *Sequence to Sequence Learning with Neural Networks* [6] do nhóm tác giả đến từ google được ông bố vào năm 2014 tại Silicon Valley AI Lab đã trình bày ý tưởng cụ thể để xây dựng một mô hình mạng nơ-ron hồi quy tối ưu với hướng đi mới so với các hệ thống dịch máy truyền thống kết hợp cùng với cơ chế chú ý (Attention mechanism) từ bài báo *Effective Approaches to Attention-based Neural Machine* [8] được thực hiện bởi nhóm tác giả đến từ đại học Stanford vào năm 2015. Từ đó nhóm sinh viên xây dựng một mô hình mạng nơ-ron hồi quy (Recurrent neural network) dùng để xây dựng một hệ thống dịch máy tiên tiến.

Mô hình seq2seq hoạt động dựa 2 mạng RNN kết hợp lại với một mạng RNN nhận nhiệm vụ mã hoá (encoder) câu đầu vào tiếng Anh thành một vector biểu diễn câu đầu vào và một mạng RNN giải mã (decoder) có nhiệm vụ giải mã vector biểu diễn câu đầu vào và kết hợp với cơ chế chú ý (Attention mechanism) để giải mã câu đầu vào và cho ra kết quả câu đầu ra tiếng Việt tương ứng.

Kiến trúc tổng thể cho việc kết hợp 2 kiến trúc trên để xây dựng một mô hình dịch máy được minh hoạ cụ thể ở hình 3.1.



Hình 3.1: Tổng quan kiến trúc mô hình dịch máy (Nguồn: [8])

## 3.2 GIẢI PHÁP BIỂU DIỄN TỪ

### 3.2.1 Tổng quan về giải pháp

Để có thể sử dụng các mô hình Deep Learning (học sâu) phục vụ cho việc dịch máy, chúng ta cần biểu diễn các từ thành các số vì các mô hình chỉ làm việc với dữ liệu số. Vì thế dựa trên các kết quả tìm kiếm và thực nghiệm của các nhà khoa học, nhóm sinh viên đề xuất sử dụng Word Embedding (nhúng từ) dùng để biểu diễn các từ thành các vector số thực. Mô hình mà nhóm chọn là Word2vec với mục đích biểu diễn các từ tiếng Anh và tiếng Việt thành các vector số thực  $n$  chiều bằng nhau (mỗi chiều là một giá trị số thực) để phục vụ cho quá trình huấn luyện.

Word2vec là một mô hình học không giám sát (model unsupervised learning) nó dùng để thể hiện mối quan hệ giữa các từ, nó được kết hợp từ hai thuật toán Skip-gram và Continuous bag of words (CBOW). Ở đây nhóm sinh viên đề

xuất sử dụng mô hình skip-gram cho biểu diễn từ. Với skip-gram, kích thước biểu diễn từ giảm từ kích thước bằng số từ trong bộ từ vựng xuống bằng chiều dài lớp ẩn. Hơn nữa các vector có ý nghĩa nhiều hơn về mặt mô tả mối quan hệ giữa các từ. Trong luận văn, nhóm sinh viên sử dụng mô hình Word2Vec Continuous Skipgram được phát triển bởi nhóm công nghệ ngôn ngữ đại học Oslo. Nhóm sinh viên sử dụng mô hình Word2Vec với số chiều là 100.

Link chứa mô hình Word2Vec: <http://vectors.nlpl.eu/repository/>

### 3.2.2 Chi tiết giải pháp

Nhóm sinh viên sử dụng đầu vào là tập dữ liệu được chia làm 2 tập tin chính chia làm 2 ngôn ngữ tiếng Anh và tiếng Việt.

Với bộ dữ liệu “IWSLT’15 English-Vietnamese data” với khoảng 100.000 câu song ngữ English-Vietnamese với dữ liệu thô chia làm hai tập tin tiếng English-Vietnamese như sau:

```
18 It &apos;s a huge amount of stuff . It &apos;s equal to the weight of methane .
19 And because it &apos;s so much stuff , it &apos;s really important for the atmospheric system .
20 Because it &apos;s important to the atmospheric system , we go to all lengths to study this thing .
21 We blow it up and look at the pieces .
22 This is the EUPHORE Smog Chamber in Spain .
23 Atmospheric explosions , or full combustion , takes about 15,000 times longer than what happens in your car .
24 But still , we look at the pieces .
25 We run enormous models on supercomputers ; this is what I happen to do .
26 Our models have hundreds of thousands of grid boxes calculating hundreds of variables each , on minute timescales .
27 And it takes weeks to perform our integrations .
28 And we perform dozens of integrations in order to understand what &apos;s happening .
29 We also fly all over the world looking for this thing .
30 I recently joined a field campaign in Malaysia . There are others .
```

```
18 Đó là một lượng khí thải khổng lồ , bằng tổng trọng lượng của metan .
19 Chính vì lượng khí thải rất lớn , nó có ý nghĩa quan trọng với hệ thống khí quyển .
20 Chính vì nó có ý nghĩa quan trọng với hệ thống khí quyển , giá nào chúng tôi cũng theo đuổi nghiên cứu này đến cùng .
21 Chúng tôi cho nó nổ và xem xét từng mảnh nhỏ .
22 Đây là Phòng nghiên cứu khói bụi EUPHORE ở Tây Ban Nha .
23 Nổ trong không khí hay cháy hoàn toàn diễn ra chậm hơn 15,000 lần so với những phản ứng trong động cơ xe .
24 Dù vậy , chúng tôi vẫn xem xét từng mảnh nhỏ .
25 Chúng tôi chạy những mô hình khổng lồ trên siêu máy tính ; đây là công việc của tôi .
26 Mô hình của chúng tôi gồm hàng trăm ngàn thùng xếp chồng tính toán với hàng trăm biến số trong thời gian cực ngắn .
27 Mà vẫn cần hàng tuần mới thực hiện xong các phép tích phân .
28 Chúng tôi cần làm hàng tá phép tính như thế để hiểu được những gì đang xảy ra .
29 Chúng tôi còn bay khắp thế giới để tìm phân tử này .
30 Gần đây tôi tham gia một cuộc khảo sát thực địa ở Malaysia . Còn nhiều chuyến khác nữa .
```

#### ❖ Bước 1

Bước đầu tiên chúng ta thực hiện xử lý các câu dữ liệu như: xoá dấu “?”, “.”, xoá dấu khoảng trắng thừa và một số thứ khác.

Nhóm sinh viên thực hiện loại bỏ các câu có độ dài hơn 100.

◆ Bước 2

Bước thứ hai, ta thực hiện tách từ để tạo từ điển của từng ngôn ngữ theo tập dữ liệu mà ta sử dụng.

The first 15 words:  
 ['the', 'science', 'behind', 'a', 'climate', 'headline', 'i', '&', 'apos', ';', 'd', 'like', 'to', 'talk']

The first 15 words:  
 ['khoa', 'học', 'dường', 'sau', 'một', 'tiêu', 'đề', 'về', 'khí', 'hậu', 'tôi', 'muốn', 'cho', 'các', 'bạn']

❖ Bước 3

Tiếp theo nhóm sinh viên thực hiện tạo các từ điển word2int và int2word cho cả hai ngôn ngữ English-Vietnamese và ta được kết quả như sau:

English:

```
The word2index:
{'<pad>': 0, '<unk>': 1, '<s>': 2, '</s>': 3, '.': 4, ',': 5, 'the': 6, ';': 7, '&': 8, 'and': 9, 'apos': 10, 'to': 11, 'of': 12,

The int2word:
{0: '<pad>', 1: '<unk>', 2: '<s>', 3: '</s>', 4: '.', 5: ',', 6: 'the', 7: ';', 8: '&', 9: 'and', 10: 'apos', 11: 'to', 12: 'of',
```

Vietnamese:

```
The word2index:
{'<pad>': 0, '<unk>': 1, '<s>': 2, '</s>': 3, '.': 4, ',': 5, 'tôi': 6, 'là': 7, 'và': 8, 'có': 9, 'một': 10}

The int2word:
{0: '<pad>', 1: '<unk>', 2: '<s>', 3: '</s>', 4: '.', 5: ',', 6: 'tôi', 7: 'là', 8: 'và', 9: 'có', 10: 'một'}
```

◆ Bước 4

Tiếp theo, nhóm sinh viên thực hiện chuyển từng câu song ngữ sang từng vector với từng từ ứng với vị trí của từ trong từ điển.

Với câu đầu vào tiếng Anh ta thực hiện thêm (padding) với những câu có độ dài bé hơn 100.

Với câu đầu vào tiếng Việt ta sẽ thực hiện thêm (padding) trong quá trình huấn luyện vì ta sẽ sử dụng độ dài thật của câu song ngữ để huấn luyện nhanh hơn.

Và ta được kết quả như sau:

English: [6, 310, 573, 13, 749, 4626, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
Vietnamese: [326, 75, 1083, 116, 10, 372, 117, 41, 411, 743]

◆ Bước 5

Tại đây ta thực hiện lấy nhúng từ (word embedding) tất cả các từ có trong từ điển English-Vietnamese.

English:

The first 5 word embedding for English:

[illegible]

Vietnamese:

The first 5 word embedding Vietnamese:

[illegible]

Sau đó quá trình chuyển từ câu song ngữ (đã được chuyển thành vector số ứng với vị trí từng từ trong từ điển) thành vector nhúng từ sẽ được thực hiện trực tiếp trong mô hình.

### 3.3 GIẢI PHÁP XÂY DỰNG MÔ HÌNH DỊCH MÁÝ

### 3.3.1 Tổng quan về giải pháp

Dựa trên các đánh giá thực tế và điều kiện phần cứng lẫn lượng dữ liệu (data) cho phép, nhóm sinh viên lựa chọn phương pháp học sâu (deep learning) để xây dựng mô hình mạng nơ-ron hồi quy (Recurrent neural network) trong mô hình dịch máy (machine neural translation). Mô hình được đào tạo từ đầu đến cuối từ những câu đã được biểu diễn dưới các nhúng từ (word embedding) để tạo ra các



chuỗi đầu vào bộ mã hoá (encoder) và bộ giải mã (decoder). Do đó với lượng dữ liệu đủ lớn và khả năng tính toán, mô hình có thể tự học một cách chính xác để thực hiện việc dịch một câu từ tiếng Anh sang tiếng Việt.

### 3.3.2 Mô hình mạng nơ-ron hồi quy và khung huấn luyện

Cốt lõi của quá trình đào tạo một mô hình RNN là để nhận vào một văn bản tiếng Anh và tạo ra một văn bản tiếng Việt tương ứng. Để dễ hình dung ta có ví dụ một tập huấn luyện  $X = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots\}$  với  $x$  là một vector các nhúng từ (word embedding) tương ứng với câu tiếng Anh đầu vào và  $y$  là một nhãn tức là một vector các nhúng từ (word embedding) tương ứng với câu tiếng Việt ở đầu ra. Mỗi câu tiếng Anh  $x^{(i)}$  là một chuỗi thời gian có độ dài  $T^{(i)}$ , trong đó mỗi đoạn thời gian nhất định là một vector nhúng từ (word embedding)  $x_t^{(i)}$ ,  $t = 1, 2, \dots, T^{(i)}$ . Mục tiêu của RNN là chuyển đổi đầu vào  $x$  thành một chuỗi xác suất ký tự cho nhãn  $y$ , với  $y_t = P(w_t | x)$ , trong đó  $w_t$  thuộc các từ trong từ điển tiếng Việt và một vài ký tự đặc biệt khác.

Mô hình RNN được nhóm sinh viên chọn sử dụng là mô hình hồi quy với 3 thành phần chính. Thành phần đầu tiên là lớp nhúng từ (embedding), lớp (layer) này có nhiệm vụ chuyển các đầu vào của bộ mã hoá (encoder) và bộ giải mã (decoder) từ dạng int sang dạng nhúng từ (word embedding) để phục vụ cho công việc tính toán phía sau.

Thành phần thứ hai là bộ mã hoá (encoder), với bộ mã hoá chúng ta sử dụng Multi layer Bi-directional LSTM với số lượng layer và số lượng hidden units của LSTM cell được thiết lập trong param. Ngoài ra nhóm sinh viên còn sử dụng DropoutWrapper để thiết lập giá trị Drop Out cho các LSTM cell để tránh hiện tượng quá khớp (over-fitting) với dữ liệu huấn luyện.

Thành phần thứ ba là bộ giải mã (decoder). Đối với bộ giải mã, chúng em chia thành hai trường hợp riêng biệt là huấn luyện mô hình(training) và dự đoán (inference). Trong quá trình huấn luyện chúng em sử dụng TrainingHelper còn khi dự đoán, chúng em sử dụng BasicDecoder với BeamSearchDecoder.

❖ Huấn luyện: chúng em sử dụng BahdanauAttention và TrainingHelper để huấn luyện mô hình. Chúng em còn sử dụng AdamOptimizer để cập nhật tham số cho mô hình và còn sử dụng Gradient Clipping để tránh mô hình bị bùng nổ độ dốc (exploding gradients).

❖ Dự đoán: sau khi huấn luyện xong mô hình và sử dụng mô hình này để dự đoán kết quả. Tuy nhiên do chúng ta không biết kết quả thực tế như trong quá trình huấn luyện, nên ta cần sử dụng các thuật toán tìm kiếm để cho ra kết quả phù hợp nhất và chúng em chọn sử dụng thuật toán tìm kiếm chùm tia (Beam Search) với beam-width = 10.

### **3.4 GIẢI PHÁP XÂY DỰNG MÁY CHỦ**

Máy chủ (server) được nhóm sinh viên chọn Amazon EC2 làm máy chủ với mục đích tạo ra một cầu nối giữa mô hình đã được huấn luyện (model) và phía ứng dụng sản phẩm (client) – được xây dựng với React Native. Vì vậy trong giới hạn của khoá luận, máy chủ chỉ cung cấp duy nhất một giao diện lập trình (API) với chức năng chuyển đổi từ một văn bản (text) tiếng Anh thành một văn bản (text) tiếng Việt tương ứng.

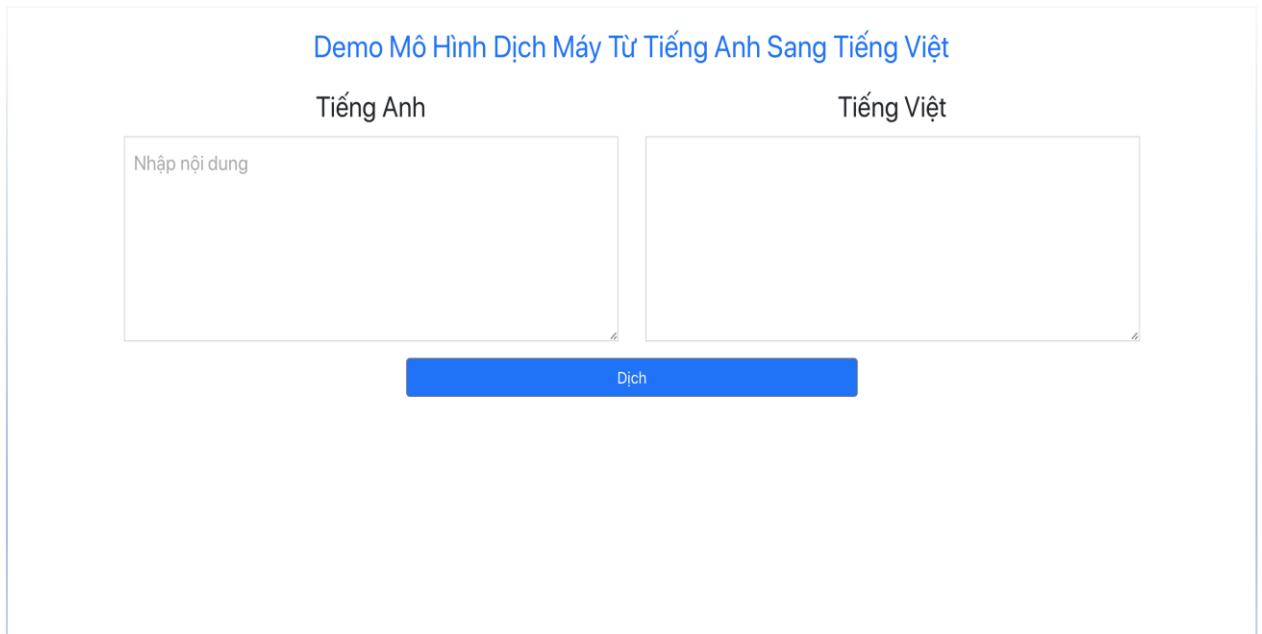
### **3.5 GIẢI PHÁP XÂY DỰNG ỨNG DỤNG**

Để ứng dụng hoá hệ thống dịch máy từ tiếng Anh sang tiếng Việt, nhóm sinh viên quyết định xây dựng web để ứng dụng kết quả của hệ thống vào một tình huống cụ thể có thể ứng dụng và thương mại hoá tốt.

Ứng dụng web do nhóm sinh viên xây dựng có chức năng chính là chuyển đổi văn bản tiếng Anh do người dùng nhập vào và đưa ra văn bản tiếng Việt tương ứng.

### 3.5.1 Thiết kế giao diện ứng dụng

Giao diện ứng dụng chỉ có một màn hình với chức năng chính là chuyển đổi một văn bản tiếng Anh thành một văn bản tiếng Việt tương ứng.



The screenshot shows a web application titled "Demo Mô Hình Dịch Máy Từ Tiếng Anh Sang Tiếng Việt". It features two input fields: "Tiếng Anh" (English) on the left and "Tiếng Việt" (Vietnamese) on the right. The "Tiếng Anh" field contains the placeholder text "Nhập nội dung". Below these fields is a blue button labeled "Dịch" (Translate).

Hình 3.2 Màn hình chính của ứng dụng

Để sử dụng, người dùng nhập văn bản tiếng Anh vào ô tiếng Anh tương ứng và nhập vào nút dịch. Kết quả sẽ được hiển thị tại ô tiếng Việt.

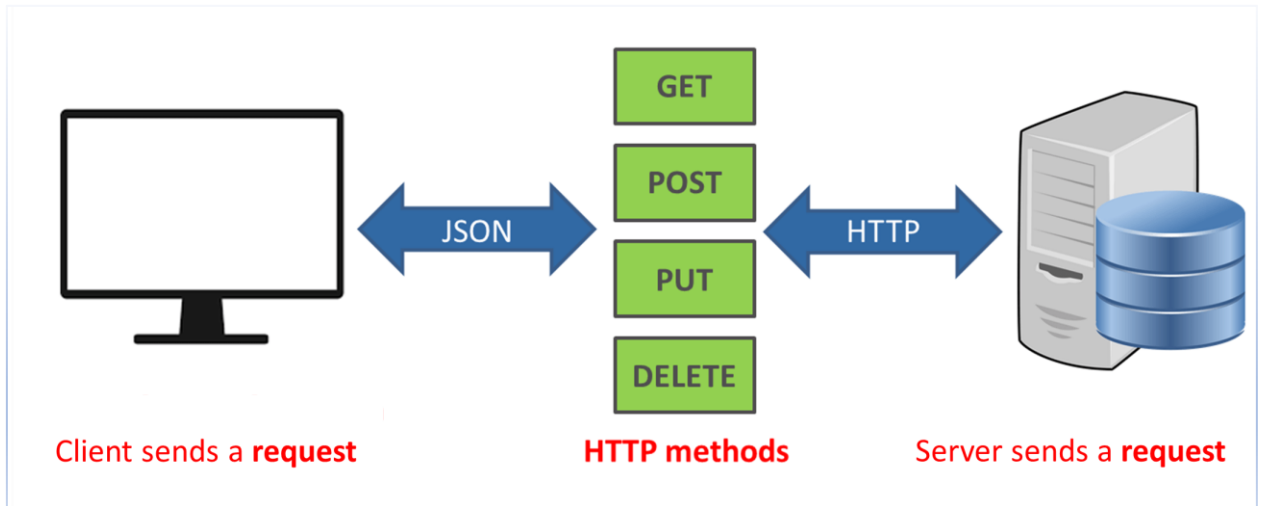
### 3.5.2 Thiết kế kiến trúc ứng dụng

Ứng dụng web minh họa thực tế cho mô hình dịch máy từ tiếng Anh sang tiếng Việt mà nhóm sinh viên đã xây dựng, các tác vụ không phức tạp nên nhóm sinh viên đề xuất sử dụng kiến trúc client-server cho ứng dụng của nhóm.

Kiến trúc client-server là một mô hình nổi tiếng trong mạng máy tính, được áp dụng rất rộng rãi và là mô hình của mọi trang web, ứng dụng di động hiện có. Ý

tường của mô hình này là máy con – Client (đóng vai trò là máy khách) gửi một yêu cầu (request) để máy chủ – Server (đóng vai trò người cung ứng dịch vụ), máy chủ sẽ xử lý và trả kết quả về cho máy khách.

Kiến trúc client-server mà nhóm sinh viên sử dụng được minh họa như hình 3.3.



Hình 3.3: Kiến trúc client-server [Nguồn: [https://en.wikipedia.org/wiki/Client-server\\_model](https://en.wikipedia.org/wiki/Client-server_model)]

### 3.6 TỔNG KẾT

Thông qua chương 3, sinh viên đã làm rõ được các giải pháp cụ thể cho từng phần trong hệ thống dịch máy từ tiếng Anh sang tiếng Việt, hướng xây dựng máy chủ và cả ứng dụng trên nền tảng web.

Nhóm sinh viên đã trình bày một hệ thống dịch máy từ tiếng Anh sang tiếng Việt dựa trên việc học sâu (deep learning) từ đầu đến cuối có khả năng vượt trội và hiện đại trong hiện đại. Nhóm sinh viên tin rằng phương pháp này sẽ tiếp tục được cải thiện với các mô hình mới hơn, đơn giản hoặc phức tạp hơn khi tận dụng được sức mạnh tính toán phần cứng và kích thước dữ liệu được tăng thêm trong tương lai.

Chương 4 nhóm sinh viên sẽ trình bày về các thư viện, công cụ và những khó khăn cụ thể nếu có cho các giải pháp đã trình bày ở chương này.

## **CHƯƠNG 4: CÀI ĐẶT VÀ TRIỂN KHAI**

### **4.1 GIỚI THIỆU VỀ PYTHON VÀ THƯ VIỆN TENSORFLOW**

#### **4.1.1 Python**

Mã nguồn xây dựng mô hình huấn luyện của đề tài được phát triển dựa trên Python. Python là một ngôn ngữ lập trình thông dịch được sử dụng rất phổ biến trong lĩnh vực khoa học máy tính nhờ những ưu điểm sau:

❖ Đa nền tảng

Python có thể chạy trên nhiều hệ điều hành như Windows, MacOS, Linux/Unix và một số hệ điều hành khác trên máy tính. Ngoài ra, Python còn có cả những phiên bản chạy được trên .NET, máy ảo Java. Tất cả chỉ với cùng một mã nguồn cho một công việc.

❖ Đơn giản

Python có cú pháp rất đơn giản, rõ ràng. Cú pháp của Python dễ viết và dễ đọc hơn rất nhiều khi so sánh với những ngôn ngữ lập trình khác như Java, C/C++, C#, JavaScript, ... Điều này cũng giúp cho nhà phát triển tập trung vào việc phát triển giải pháp thay vì cú pháp.

❖ Mã nguồn mở

Python là một dự án mã nguồn mở nên nhà phát triển có thể thoải mái sử dụng cho các mục đích cá nhân và vì vậy nên cộng đồng phát triển Python thường xuyên đưa ra những bản cập nhật mới nhằm tăng trải nghiệm cũng như tối ưu hoá Python.

❖ Nhiều thư viện hỗ trợ

Python có một khối lượng lớn các thư viện tiêu chuẩn giúp cho công việc của nhà phát triển trở nên dễ dàng hơn rất nhiều, đặc biệt là các thư viện xử lý toán học của Python cực kỳ đa dạng và mạnh mẽ.

#### **4.1.2 Tensorflow**

Thư viện Tensorflow được sử dụng trong việc tính toán các biểu đồ và các dữ liệu dưới dạng số hoá trong sản phẩm khoá luận. Là một thư viện mã nguồn mở hỗ trợ mạnh mẽ các phép toán học để tính toán trong máy học. Để xây dựng một mô hình huấn luyện cho đề tài, nhóm sử dụng các giao diện lập trình cấp thấp (low level APIs) mà Tensorflow cung cấp:

#### ❖ Tensor

Đây là một sự khái quát hóa các vector và ma trận cho các kích thước có khả năng cao hơn. Là cấu trúc dữ liệu đại diện cho tất cả các loại dữ liệu trong Tensorflow. Một tensor sẽ có 3 thuộc tính cơ bản nhất bao gồm:

- Số bậc (rank): giúp phân loại dữ liệu của tensor. (Scalar, Vector, Matrix, N- Tensor)
- Số chiều (shape): giúp xác định mức độ tương hợp giữa các tensor khi thực hiện tính toán.
- Kiểu dữ liệu (type): kiểu dữ liệu cho toàn bộ các thành phần (elements) trong tensor.

#### ❖ Graph

Đây là một loại đồ thị với các đỉnh (node) là đại diện cho biến đầu vào hoặc một phép tính toán và các cạnh (edge) là đại diện cho dữ liệu truyền bên trong đồ thị tức dữ liệu đầu vào và đầu ra của các phép tính tại một đỉnh. Và trong tensorflow, tất cả thành phần bên trong một đồ thị đều ở dạng tensor. Cách xử lý tính toán theo hướng đồ thị này có thể giúp tensorflow tận dụng được khả năng tính toán song song bằng việc chia tách các phép toán độc lập và khả năng phân tán khi chia nhỏ công việc xử lý cho nhiều CPU, GPU khác nhau.

#### ❖ Session

Đây là một phiên xử lý được định nghĩa trong thư viện tensorflow. Một đối tượng phiên (session) cung cấp quyền truy cập vào các thiết bị trong máy cục bộ và các thiết bị từ xa bằng cách sử dụng thời gian chạy phân tán. Nó cũng lưu trữ

thông tin về đồ thị (graph) để có thể chạy cùng một tính toán hiệu quả nhiều lần. Nếu không có phiên (session), mọi tính toán trong đồ thị (graph) sẽ gần như không được triển khai.

## **4.2 DỮ LIỆU HUẤN LUYỆN MÔ HÌNH**

Để huấn luyện một hệ thống dịch máy từ tiếng Anh sang tiếng Việt dựa trên mô hình nhóm sinh viên phát triển thì bộ dữ liệu nhóm sử dụng là “IWSLT’15 English-Vietnamese data” bao gồm khoảng 133.000 câu do đại học Stanford phát triển (nhóm sinh viên chỉ lấy những câu có độ dài bé 100 nên chỉ có khoảng 100.000 câu) có hai thành phần chính bao gồm:

### **❖ Văn bản tiếng Anh**

Để huấn luyện hệ thống dịch máy từ tiếng Anh sang tiếng Việt đủ tốt thì lượng dữ liệu văn bản dùng để huấn luyện cũng phải đủ nhiều và đủ tốt. Nhóm sinh viên đã thu thập được khoảng 100.000 câu song ngữ để tiến hành huấn luyện. Ngoài ra dữ liệu khi huấn luyện cũng cần điều chỉnh sao cho độ dài bé hơn 100 từ để bảo đảm mô hình huấn luyện không bị thiếu tài nguyên.

### **❖ Văn bản tiếng Việt**

Là bản dịch tương ứng với nội dung của câu tiếng Anh. Dữ liệu tiếng Việt với khoảng 100.000 câu và để bảo đảm mô hình huấn luyện không thiếu tài nguyên ta cũng nên hạn chế độ dài lớn hơn 100 từ vì nhóm sinh viên giới hạn độ dài câu. Nếu câu dài hơn sẽ bị cắt bỏ và mất đi các nội dung quan trọng, mô hình sẽ huấn luyện lâu hơn, sai sót.

Trong quá trình thu thập dữ liệu, nhóm sinh viên đã gặp rất nhiều vấn đề về chất lượng dữ liệu như: các tập dữ liệu song ngữ English – Vietnamese có khá nhiều với các dự án như là .... Tuy nhiên các dự án này lại không công khai dữ liệu nên nhóm sinh viên phải thu thập khắp nơi. Đối với những mẫu có mức độ sai lệch nhỏ nhóm cố gắng tinh chỉnh sao cho phù hợp nhất. Những mẫu bị sai lệch nhiều hoặc chất



lượng quá thấp buộc nhóm sinh viên phải bỏ. Việc này một phần sẽ giảm bớt tình trạng gây nhiễu cho mô hình trong quá trình huấn luyện. Điều này dẫn đến thời gian huấn luyện mô hình còn khoảng (?323332) giờ cho khoảng 100.000 câu song ngữ. Trong đó, dữ liệu được chia nhỏ thành 3 bộ train, dev, test với kích thước như sau:

- ❖ Bộ train: chứa khoảng 100.000 câu song ngữ English-Vietnamese.
- ❖ Bộ dev: chứa khoảng 1500 câu song ngữ English-Vietnamese.
- ❖ Bộ test: chứa khoảng 1200 câu song ngữ English-Vietnamese.

## 4.3 CÀI ĐẶT

### 4.3.1 Giới thiệu

Mã nguồn được nhóm sinh viên phát triển dựa trên tham khảo các bài báo như: *Sequence to Sequence Learning with Neural Networks* [6] do nhóm tác giả đến từ google được ông bố vào năm 2014 tại Silicon Valley AI Lab đã trình bày ý tưởng cụ thể để xây dựng một mô hình mạng nơ-ron hồi quy tối ưu với hướng đi mới so với các hệ thống dịch máy truyền thống kết hợp cùng với cơ chế chú ý (Attention mechanism) từ bài báo *Effective Approaches to Attention-based Neural Machine* [8] được thực hiện bởi nhóm tác giả đến từ đại học Stanford vào năm 2015.

Từ đó nhóm sinh viên tự phát triển mô hình dịch máy cho tác vụ dịch tiếng Anh sang tiếng Việt để phục vụ cho luận văn. Mô hình được phát triển trên Python3 và thư viện Tensorflow là chính.

### 4.3.2 Cài đặt

Đầu tiên ta cần tải về mã nguồn mô hình dịch máy của nhóm sinh viên phát triển từ github (<https://github.com/nmtri1912/Model>).

Phần hướng dẫn cài đặt của nhóm sinh viên yêu cầu bắt buộc về những thư viện cũng như công cụ mà nhóm sinh viên có khuyến nghị trên liên kết github phía trên để có thể chạy mô hình của nhóm:

❖ Python 3.6

❖ Tensorflow 1.x

❖ Hệ điều hành MacOS hoặc Linux, Ubuntu

Để huấn luyện mô hình dịch máy cho tác vụ dịch ta cần cài đặt theo hướng dẫn của nhóm sinh viên để tránh gặp lỗi không đáng có.

Để huấn luyện mô hình với bộ dữ liệu câu Anh-Việt, ta cần chuyển thành các số để mô hình có thể huấn luyện và phương pháp đó là sử dụng từ nhúng (word embedding).

Nhóm sinh viên sử dụng pre-trained model Word2vec tại <http://vectors.nlpl.eu/repository/> cho tiếng Anh và tiếng Việt có thông số như sau:

❖ English CoNLL17 corpus (ID = 40): được xây dựng dựa trên thuật toán “Word2Vec continuons Skipgram” với 100 chiều và hơn 4.000.000 từ.

❖ Vietnamese CoNLL17 corpus (ID = 74): được xây dựng dựa trên thuật toán “Word2Vec continouns Skipgram” với 100 chiều và khoảng 3.800.000 từ.

#### 4.4 HUẤN LUYỆN MÔ HÌNH

Để mô hình có thể sử dụng mô hình để đưa ra các dự đoán dịch chính xác, ta cần tìm ra các tham số phù hợp cho mô hình, nhóm sinh viên tham khảo và tiến hành công việc huấn luyện mô hình (điều chỉnh các siêu tham số - hyperparameters) để tìm ra những giá trị tối ưu để mô hình dự đoán chính xác nhất.

Quá trình huấn luyện được nhóm triển khai trên Google Colab có cấu hình GPU 25GB. Sau một thời gian huấn luyện với bộ dữ liệu “IWSLT’15 English-Vietnamese data” với 10 epoch và ghi nhận kết quả, nhóm sinh viên đã tổng hợp một số siêu tham số có ảnh hưởng đến kết quả dự đoán của mô hình như:

- num\_layers: số lớp của mô hình

- num\_hidden: độ rộng của một lớp khi khởi tạo trong mô hình
- batch\_size: số lượng mẫu được sử dụng khi khởi tạo các lớp trong mô hình

#### 4.4.1 Điều chỉnh num\_layer

- Chọn num\_layers = 1

Tên tham số	Giá trị	Ghi chú
num_layers	1	
num_hidden	256	Giá trị mặc định
batch_size	64	Giá trị mặc định
<b>BLEU</b>	<b>26.80</b>	

- Chọn num\_layers = 2

Tên tham số	Giá trị	Ghi chú
num_layers	2	
num_hidden	256	Giá trị mặc định
batch_size	64	Giá trị mặc định
<b>BLEU</b>	<b>28.60</b>	

Kết luận: Chọn num\_layers = 2 cho các lần huấn luyện sau

#### 4.4.2 Điều chỉnh num\_hidden

- Chọn num\_hidden = 256

Tên tham số	Giá trị	Ghi chú
num_layers	2	
num_hidden	256	
batch_size	64	Giá trị mặc định
<b>BLEU</b>	<b>28.60</b>	

- Chọn num\_hidden = 512

Tên tham số	Giá trị	Ghi chú
num_layers	2	

num_hidden	512	
batch_size	64	Giá trị mặc định
<b>BLEU</b>	<b>29.42</b>	

Kết luận: Chọn num\_hidden = 512 cho các lần huấn luyện sau.

#### 4.4.3 Điều chỉnh batch\_size

- Chọn batch\_size = 64

Tên tham số	Giá trị	Ghi chú
num_layers	2	
num_hidden	512	
batch_size	64	
<b>BLEU</b>	<b>29.42</b>	

- Chọn batch\_size = 128

Tên tham số	Giá trị	Ghi chú
num_layers	2	
num_hidden	512	
batch_size	64	
<b>BLEU</b>	<b>29.63</b>	

Kết luận chọn batch\_size = 128 cho lần huấn luyện sau.

Như vậy mô hình của nhóm sinh viên sử dụng có các tham số như sau:

- Num\_layers = 2
- Num\_hidden = 512
- Batch\_size = 128
- Beam\_width = 10
- Keep\_prob = 0.85

## 4.5 ĐÓNG GÓI MÔ HÌNH

Nhóm sinh viên lưu các tham số mô hình học được sau mỗi lần chạy xong 1 epoch (một lần duyệt qua toàn tập huấn luyện) với định dạng **NMT.ckpt**. Tập tin này có thể hiểu là các tham số được chọn lọc trong quá trình huấn luyện.

Để sử dụng tập tin này để thực hiện tác vụ dịch máy, chúng ta cần định nghĩa lại một số thư cần thiết như: từ điển word2int, int2word, từ nhúng (word embedding), xử lí đầu vào và mô hình.

#### **4.6 XÂY DỰNG MÁY CHỦ (SERVER)**

Flask Framework, AWS EC2 (hoặc AWS Elastic Beanstalk), Google Cloud Platform và Heroku là bốn nền tảng được nhóm sinh viên chọn để xây dựng hệ thống máy chủ nhằm đóng vai trò làm cầu nối giữa ứng dụng và mô hình nhận dạng âm thanh. Với các yếu tố như tốc độ triển khai nhanh gọn, sự tiện ích và tính thông dụng nên việc chọn hai nền tảng này để xây dựng máy chủ là quyết định phù hợp với nhu cầu đặt ra của nhóm sinh viên.

Hệ thống máy chủ trong giới hạn luận văn này sẽ cung cấp ra bên ngoài duy nhất một giao diện lập trình ứng dụng (Application Programming Interface-API) để chuyển đổi văn bản tiếng Anh (dạng text) nhận được và trả về dữ liệu văn bản tiếng Việt tương ứng.

Trong khi đó Web ứng dụng để sử dụng API mà hệ thống cung cấp, nhóm sinh viên sử dụng Reactjs Framework để xây dựng giao diện và được triển khai trực tiếp lên Heroku.

#### **4.7 MỘT SỐ VẤN ĐỀ PHÁT SINH VÀ GIẢI PHÁP**

- Thiếu nguồn dữ liệu: nhóm sinh viên thu tập thêm dữ liệu mới.
- Thiếu tài nguyên để huấn luyện mô hình (GPU): chúng em sử dụng dịch vụ google colab.
- Khi deploy lên server gặp phải trường hợp front-end gọi API được 2 đến 3 lần thì server bị tắt: cài nginx.

- Khi huấn luyện trên google colab thì gặp phải vấn đề giới hạn của google colab: train tối đa khoảng 10-12 tiếng sẽ bị mất kết nối, hoặc có sự cố phát sinh thì sẽ mất kết quả huấn luyện trước đó: chúng em đặt checkpoint để lưu lại kết quả sau mỗi epoch. Khi mất kết nối thì ta chỉ cần tải lại checkpoint để huấn luyện tiếp mà không phải huấn luyện lại từ đầu.

#### **4.8 TỔNG KẾT**

Trong chương 4, nhóm sinh viên đã trình bày về cách thức cài đặt và triển khai cho các thành phần bao gồm hệ thống dịch máy, trang web chạy thử API của hệ thống. Nội dung chi tiết cho một số phần cài đặt được nhóm sinh viên trình bày chi tiết ở phần phụ lục, chương 5 sẽ là các tổng kết về quá trình thực hiện luận văn của nhóm sinh viên.

## CHƯƠNG 5: TỔNG KẾT VÀ ĐÁNH GIÁ

### 5.1 KIẾN THỨC ĐẠT ĐƯỢC

Trong thời gian thực hiện khoá luận tốt nghiệp, chúng em đã được cung cấp thêm nhiều điều mới, kiến thức mới:

- Có những kiến thức tổng quan về dịch máy và các kỹ thuật trong mạng nơ-ron để dịch máy.
- Có được các kiến thức và kinh nghiệm trong việc thiết kế, xây dựng, triển khai và kiểm thử khi thực hiện phát triển một hệ thống cung cấp dịch vụ.
- Biết được quá trình phát triển của dịch máy.
- Học hỏi các quy trình phát triển dự án phần mềm như Kanban, Waterfall, Scrum và có cơ hội được áp dụng vào luận văn.
- Nâng cao khả năng làm việc nhóm và giao tiếp của các thành viên trở nên tốt hơn.
- Khả năng tìm kiếm, đọc tài liệu và sách báo nâng cao. Hình thành được các thói quen như trích dẫn tài liệu. Khả năng tổng hợp các kiến thức từ nhiều nguồn khác nhau được nâng cao.
- Nâng cao được khả năng lên kế hoạch và đảm bảo hoàn thành trong khoảng thời gian cho trước.
- Hình thành kỹ năng tự chủ, tự tập và tinh thần trách nhiệm công việc.
- Học hỏi được cách trình bày, viết tài liệu một cách hợp lý và đẹp mắt.
- Có thêm kinh nghiệm đọc hiểu, hiệu chỉnh từ mã nguồn đã được phát triển. Khả năng chỉnh sửa và khắc phục khi gặp lỗi được nâng cao.

### 5.2 KẾT QUẢ MÔ HÌNH HUẤN LUYỆN

So sánh với các mô hình ứng với các bài báo tại

<https://paperswithcode.com/sota/machine-translation-on-iwslt2015-english-1>

Mô hình	Bộ dữ liệu	BLEU score (test)
---------	------------	-------------------

Mô hình nhóm sinh viên	IWSLT English-Vietnamese	29.63
Transformer + BPE + dropout	IWSLT English-Vietnamese	33.27
Transformer + BPE + Fix Norm + ScaleNorm	IWSLT English-Vietnamese	32.8
Transformer + LayerNorm-simple	IWSLT English-Vietnamese	31.4

Những mô hình được so sánh sử dụng kiến trúc mô hình mới đó là Transformer. Mô hình này đã được minh chứng có hiệu quả tốt hơn trong tác vụ dịch máy.

## 5.3 KẾT QUẢ HỆ THỐNG

### 5.3.1 Môi trường phát triển

Hệ điều hành: Ubutu, MacOS và Windows

Công cụ phát triển phần mềm: Visual Code, Jupyter notebook và Google Colab

Công cụ kiểm thử API: Postman

Các thư viện/nền tảng được sử dụng:

Tên thư viện / nền tảng	Tóm tắt chức năng
Python	Python là một ngôn ngữ lập trình thông dịch được sử dụng rất phổ biến trong lĩnh vực khoa học máy tính nhờ những ưu điểm như: Đa nền tảng, đơn giản, mã nguồn mở, nhiều thư viện hỗ trợ.



Flask	Thư viện Tensorflow được sử dụng trong việc tính toán các biểu đồ và các dữ liệu dưới dạng số hoá trong sản phẩm khoá luận. Là một thư viện mã nguồn mở hỗ trợ mạnh mẽ các phép toán học để tính toán trong máy học.
Matplotlib	Nó là một thư viện vẽ đồ thị rất mạnh mẽ, giúp ta có cái nhìn trực quan hơn về dữ liệu.
Numpy	NumPy là một thư viện cho ngôn ngữ lập trình Python, thêm hỗ trợ cho các mảng và ma trận lớn, đa chiều, cùng với một tập hợp lớn các hàm toán học cấp cao để hoạt động trên các mảng này.
Pandas	Pandas dùng để thao tác và phân tích dữ liệu. Cụ thể, nó cung cấp các cấu trúc dữ liệu và các thao tác để thao tác các bảng số và chuỗi thời gian.

Scipy	Scipy chứa các mô-đun để tối ưu hóa, đại số tuyến tính, tích hợp, nội suy, các chức năng đặc biệt, FFT, xử lý tín hiệu và hình ảnh, bộ giải ODE và các nhiệm vụ phổ biến khác trong khoa học và kỹ thuật.
Scikit-learn	Scikit-learn được thiết kế dựa trên nền Numpy và Scipy. Scikit-learn chứa hầu hết các thuật toán machine learning hiện đại nhất, đi kèm với tài liệu và luôn được cập nhật.
Tensorflow	Thư viện Tensorflow là thư viện mã nguồn mở dùng cho tính toán số học sử dụng đồ thị luồng dữ liệu.
Keras	Keras là một thư viện mạng nơ-ron mã nguồn mở được viết bằng Python. Được thiết kế để cho phép thử nghiệm nhanh với các mạng thần kinh sâu, nó tập trung vào việc thân thiện với người dùng, mô-đun và mở rộng.

--	--

### **5.3.2 Môi trường triển khai**

Nền tảng đám mây Amazon EC2

Nền tảng đám mây Google Cloud (dự phòng)

### **5.3.3 Chức năng đã cài đặt**

Cung cấp API nhận vào văn bản tiếng Anh dạng text và trả về đối tượng text chứa văn bản tiếng Việt tương ứng.

## **5.4 KẾT QUẢ ỨNG DỤNG WEB**

### **5.4.1 Môi trường phát triển**

- Hệ điều hành: Ubutu, MacOS và Windows
- Công cụ phát triển phần mềm: Visual Code
- Công cụ kiểm thử API: Postman
- Trình duyệt kiểm thử web: chrome, safari, microsoft edge
- Các thư viện/nền tảng được sử dụng:
  - React Framework: xây dựng giao diện
  - Nodejs: nền tảng để chạy React

### **5.4.2 Môi trường triển khai**

Server: heroku

Thiết bị: Thiết bị cần có trình duyệt web như: safari, chrome, firefox, ...

Hệ điều hành: MacOS, Ubuntu, Linux

### **5.4.3 Chức năng đã cài đặt**

Cung cấp dịch vụ nhận vào văn bản tiếng Anh dạng text và trả về đối tượng text chứa văn bản tiếng Việt tương ứng.

## 5.5 SO SÁNH KẾT QUẢ VỚI CÁC MỤC TIÊU ĐẶT RA

Mục tiêu ban đầu	Nhận xét mức độ hoàn thành
Trình bày lý do xây dựng mô hình dịch máy	Đã trình bày các lý do ở chương 1 của luận văn
Trình bày lý thuyết nền tảng và giải pháp để xử lý việc dịch một văn bản từ tiếng Anh sang tiếng Việt.	Đã trình bày ở chương 2
Xây dựng, thu thập dữ liệu và đào tạo mô hình để dịch một văn bản từ tiếng Anh sang tiếng Việt.	Đã được trình bày ở chương 3 và 4
Xây dựng một trang web demo việc sử dụng mô hình để dịch một văn bản từ tiếng Anh sang tiếng Việt.	Đã được trình bày ở chương 3 và 4
Viết 120 trang luận văn theo đúng chuẩn yêu cầu và trích dẫn các tài liệu tham khảo đầy đủ.	Luận văn được viết tương đối đầy đủ và chính xác.

## **5.6 ĐỊNH HƯỚNG PHÁT TRIỂN VÀ NGHIÊN CỨU TRONG TƯƠNG LAI**

- Cải thiện lại mã nguồn để dịch được chính xác và hợp lý hơn.
- Nghiên cứu kỹ hơn về lý thuyết nền tảng, từ đó có các bước cải thiện và thực hiện chức năng một cách đúng đắn.
- Chỉnh sửa các tài liệu nghiên cứu, hướng dẫn như luận văn, hướng dẫn sử dụng để giúp người dùng mau chóng nắm bắt được vấn đề.
- Hoàn thiện chức năng dịch văn bản, sửa một số lỗi còn tồn tại. Hoặc đổi phương pháp xây dựng mô hình để chuẩn xác hơn.
- Cải thiện tốc độ xử lý các tác vụ của ứng dụng, giúp ứng dụng chạy mượt mà và tạo trải nghiệm tốt hơn cho người dùng.
- Trình bày, chỉnh sửa mã nguồn theo khuôn mẫu để dễ dàng bảo trì và chỉnh sửa trong tương lai.
- Thu thập thêm dữ liệu để mô hình huấn luyện và dịch chính xác hơn.

## **LỜI KẾT**

Luận văn “Xây dựng mô hình dịch máy từ tiếng Anh sang tiếng Việt”, hệ thống cung cấp dịch vụ và ứng dụng được xây dựng là sản phẩm kết tinh của một quá trình học tập, làm việc và nghiên cứu nghiêm túc của nhóm sinh viên. Tuy hệ thống còn nhiều hạn chế về hệ thống lẫn khả năng xử lý của nó, song sản phẩm hệ thống cung cấp dịch vụ dịch máy từ tiếng Anh sang tiếng Việt đã đem lại cho nhóm sinh viên những kiến thức và kinh nghiệm quý báu cũng như cách để triển khai các dự án thực tế trong tương lai. Các hệ thống học sâu nói chung và dịch máy nói riêng hiện đang là những lĩnh vực nổi trội trên thế giới và nó đem lại lợi ích tuyệt vời trong cuộc sống. Những trải nghiệm trong luận văn là những kinh nghiệm quý báu cho chúng em để có những kiến thức và tiếp tục nghiên cứu và phát triển sự nghiệp của bản thân.

## TÀI LIỆU THAM KHẢO

- [1] Yanis Andrew Ioannou. "Structural Priors in Deep Neural Networks" September 2017.
- [2] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *arXiv preprint arXiv:1502.03167* (2015).
- [3] Zachary C. Lipton, John Berkowitz, Charles Elkan. "A *Critical Review of Recurrent Neural Networks for Sequence Learning*" June 5th, 2015, pp.10-11.
- [4] Alex Graves, Navdeep Jaitly. "Towards End-to-End Speech Recognition with Recurrent Neural Networks", pp.3-4.
- [5] [https://en.wikipedia.org/wiki/Hopfield\\_network](https://en.wikipedia.org/wiki/Hopfield_network)
- [6] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." *Advances in neural information processing systems*. 2014.
- [7] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *arXiv preprint arXiv:1409.0473* (2014).
- [8] Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." *arXiv preprint arXiv:1508.04025* (2015).