

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN CUỐI KÌ MÔN XỬ LÝ NGÔN NGỮ TỰ NHIÊN
TƯƠNG ĐỒNG VĂN BẢN
(DOCUMENT SIMILARITY)

Người hướng dẫn: PGS. TS LÊ ANH CƯỜNG

Người thực hiện: HỒNG QUANG VINH – 186005004

NGUYỄN ĐẠI THỊNH – 186005035

VÕ ĐĂNG KHOA – 186005032

Lớp: 18600531

Khoá: 2018-2020

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2019

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN CUỐI KÌ MÔN XỬ LÝ NGÔN NGỮ TỰ NHIÊN
TƯƠNG ĐỒNG VĂN BẢN
(DOCUMENT SIMILARITY)

Người hướng dẫn: PGS. TS LÊ ANH CƯỜNG

Người thực hiện: HỒNG QUANG VINH – 186005004

NGUYỄN ĐẠI THỊNH – 186005035

VÕ ĐĂNG KHOA – 186005032

Lớp: 18600531

Khoá: 2018-2020

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2019

LỜI CẢM ƠN

Nhóm chúng em xin chân thành cảm ơn Thầy Lê Anh Cường đã giúp đỡ chúng em hoàn thành đồ án. Những hướng dẫn của Thầy giúp chúng em có một nền tảng lý thuyết đủ để có thể ứng dụng và nghiên cứu phát triển đề tài này. Xin chân thành cảm ơn Thầy.

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

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

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

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

TP. Hồ Chí Minh, ngày tháng năm

Tác giả

(ký tên và ghi rõ họ tên)

Hồng Quang Vinh

Nguyễn Đại Thịnh

Võ Đăng Khoa

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

TÓM TẮT

Ngày nay, với sự phát triển của cuộc Cách mạng Công nghiệp 4.0, tiên phong là sự phát triển của lĩnh vực Trí tuệ nhân tạo, con người đang tiến gần hơn tới mục đích làm cho máy tính mô phỏng được các hành vi của con người. Có thể kể đến một số lĩnh vực như nhận dạng hình ảnh, tiếng nói... Trong đó, Xử lý ngôn ngữ tự nhiên là một nhanh đang được quan tâm và có tốc độ phát triển nhanh nhất.

Xử lý ngôn ngữ tự nhiên là quá trình mô phỏng và huấn luyện cho máy tính cách mà con người tiếp nhận, xử lý ngôn ngữ. Quá trình mô phỏng này giống như cách mà con người học ngôn ngữ, xử lý văn bản chính là xử lý dựa trên mức độ cơ sở của văn bản đó, nói cách khác, chúng ta cần xử lý trên mức độ từ và câu của văn bản đó. Chính vì vậy, để có thể giải quyết bài toán này, người thực hiện cũng cần có một lượng kiến thức cơ sở về ngôn ngữ học. Một số thành tựu trong lĩnh vực này có thể kể đến như: Chatbot, dịch tự động, so sánh sự tương đồng giữa hai văn bản...

Tuy nhiên, đi cùng với đó là không ít khó khăn, thách thức như: Sự khác nhau về ngôn ngữ của mỗi quốc gia, cơ sở dữ liệu của các ngôn ngữ còn thiếu... Dẫn đến hiệu quả của các mô hình xử lý ngôn ngữ không được cao.

Trong đồ án này, nhóm hướng đến các ứng dụng của xử lý ngôn ngữ tự nhiên, mục tiêu có thể xác định sự tương đồng của hai văn bản cho trước. Các khái niệm cũng như quá trình thực hiện sẽ được trình bày ở phần sau của đồ án này.

MỤC LỤC

<i>LỜI CẢM ƠN</i>	<i>i</i>
<i>PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN</i>	<i>iii</i>
<i>TÓM TẮT</i>	<i>iv</i>
<i>MỤC LỤC</i>	<i>1</i>
<i>DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT</i>	<i>3</i>
CÁC KÝ HIỆU	<i>3</i>
CÁC CHỮ VIẾT TẮT	<i>3</i>
<i>CHƯƠNG 1 – MỞ ĐẦU</i>	<i>6</i>
1.1 Giới thiệu đề tài	<i>6</i>
1.2 Mục tiêu và phạm vi đề tài	<i>6</i>
1.3 Các công việc liên quan.....	<i>7</i>
1.4 Cơ sở dữ liệu	<i>11</i>
<i>CHƯƠNG 2 – CƠ SỞ LÝ THUYẾT</i>	<i>12</i>
2.1 Word Embedding	<i>12</i>
2.1.1 One-hot vector	<i>12</i>
2.1.2 Word2vec	<i>13</i>
2.2 Recurrent Neural Network (RNN)	<i>16</i>
2.2.1 Giới thiệu	<i>16</i>
2.2.2 Các dạng của RNN	<i>17</i>
2.2.3 Phân tích RNN.....	<i>19</i>
2.2.4 Công thức RNN	<i>21</i>
2.2.5 Ví dụ RNN.....	<i>22</i>
2.2.6 Huấn luyện RNN	<i>23</i>
2.3 Siamese Recurrent Neural Network.....	<i>24</i>
2.4 Stack Recurrent Neural Network.....	<i>24</i>
2.5 Long Short-Term Memory (LSTM).....	<i>25</i>
2.5.1 Nhắc lại Recurrent Neural Network	<i>26</i>
2.5.2 Vấn đề phụ thuộc quá dài (Long-Term Dependencies).....	<i>27</i>
2.5.3 LSTM Network	<i>30</i>
2.5.4 Ý tưởng của LSTM.....	<i>32</i>
2.5.5 Phân tích mô hình LSTM	<i>34</i>

2.6 LSTM nâng cao	36
2.7 Hàm Softmax	37
2.8 Contrastive Loss Function.....	37
2.9 Adam Optimal Function.....	39
<i>CHƯƠNG 3 – MÔ HÌNH GIẢI QUYẾT BÀI TOÁN.....</i>	41
 3.1 Mô hình huấn luyện	41
<i>CHƯƠNG 4 – KẾT QUẢ THỰC NGHIỆM.....</i>	44
 4.1 Phương pháp đánh giá.....	44
<i>CHƯƠNG 5 – KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....</i>	46
 5.1 Các hạn chế.....	46
 5.2 Những điều đã đạt được	46
 5.3 Hướng phát triển.....	46

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

CÁC KÝ HIỆU

CÁC CHỮ VIẾT TẮT

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH

Hình 2: Ví dụ về word2vec	13
Hình 3: Mô hình Cbow	14
Hình 4: Mô hình skip-gram	15
Hình 5: Mô hình RNN	16
Hình 6: RNN dạng một - nhiều	18
Hình 7: RNN dạng nhiều - một	18
Hình 8: RNN dạng nhiều - nhiều	19
Hình 9: RNN dạng nhiều - nhiều	19
Hình 10: Phân tích RNN	20
Hình 11: RNN tạo các vector đầu ra	20
Hình 12: Công thức tổng quát của RNN	21
Hình 13: Công thức RNN	21
Hình 14: Truyền ký tự vào RNN	22
Hình 15: Áp dụng công thức RNN	22
Hình 16: Tính toán xác suất của từ	23
Hình 17: Mô hình Siamese RNN	24
Hình 18: Mô hình Stack RNN	25
Hình 19: Mô hình RNN có vòng lặp	26
Hình 20: Tách các vòng lặp từ RNN	27
Hình 21: Vấn đề phụ thuộc dài	27
Hình 22: Vấn đề phụ thuộc dài	28
Hình 23: Công thức LSTM	28
Hình 24: Áp dụng liên tiếp hàm sigmoid()	30
Hình 25: Cấu trúc RNN chuẩn	31

Hình 26: Cấu trúc LSTM	31
Hình 27: Ký hiệu được sử dụng.....	32
Hình 28: Đường truyền trạng thái.....	33
Hình 29: Công LSTM	33
Hình 30: Đầu vào thông tin.....	34
Hình 31: Quyết định thông tin	35
Hình 32: Cập nhật vào Cell State.....	35
Hình 33: Thông tin đầu ra.....	36
Hình 34: Mô hình LSTM nâng cao.....	36
Hình 35: Contrastive loss function	39
Hình 36: Mô hình huấn luyện	41
Hình 37: Mô hình LSTM	42
Hình 38: Mô hình kiểm tra	44

CHƯƠNG 1 – MỞ ĐẦU

1.1 Giới thiệu đề tài

Xử lý ngôn ngữ tự nhiên là một tập hợp các quá trình, từ giai đoạn tiền xử lý như tách câu, tách từ cho đến giai đoạn huấn luyện, mô phỏng quá trình học ngôn ngữ của con người... Đây là hướng nghiên cứu được quan tâm nhiều trong lĩnh vực công nghệ hiện nay với rất nhiều ứng dụng. Con người có thể học nhiều loại ngôn ngữ khác tiếng mẹ đẻ của mình, tuy nhiên quá trình học cũng gặp nhiều trở ngại như sự khác biệt ngôn ngữ ở các vùng miền. Chính vì thế, việc mô phỏng cho máy có khả năng học ngôn ngữ giống con người lại khó hơn rất nhiều lần. Nguyên nhân chủ yếu là do sự đa dạng, phong phú của từng loại ngôn ngữ, cũng như khó khăn khi xây dựng bộ dữ liệu của các ngôn ngữ.

Tương đồng văn bản là một nhánh nhỏ trong lĩnh vực xử lý ngôn ngữ tự nhiên. Thông qua việc so sánh sự tương đồng của hai văn bản, người ta có thể ứng dụng vào một số lĩnh vực như: Phát hiện đạo văn, tóm tắt văn bản, đánh giá độ chính xác của dịch máy...

Ý tưởng của phương pháp tìm độ tương đồng nói riêng hay xử lý ngôn ngữ tự nhiên nói chung là dựa vào tính chất của các từ trong câu như: Số lần xuất hiện, nghĩa của từ cũng như vị trí của từ trong câu đó. Từ đó rút ra các đặc trưng của từ trong câu, áp dụng các kỹ thuật học máy để đánh giá sự tương đồng của hai văn bản đó.

1.2 Mục tiêu và phạm vi đề tài

Mục tiêu của đề tài là tìm kiếm, tìm hiểu và mô phỏng lại một hệ thống có thể xác định xem hai văn bản có tương đồng với nhau hay không. Để thực hiện được điều này, trước hết ta cần có một mức độ hiểu biết về tính chất của từng loại ngôn ngữ. Đơn vị cơ bản nhất cấu thành nên một loại ngôn ngữ là các từ. Nhiều từ kết hợp thành câu và nhiều câu kết hợp thành một văn bản. Hiểu được điều này, ta có thể dựa vào tính chất của các từ (vd: từ đồng nghĩa) để có thể xây dựng một hệ thống xác định sự tương đồng của các văn bản.

Trên thực tế, tuỳ từng loại ngôn ngữ sẽ có các đặc điểm khác nhau như: Dấu câu, ngữ pháp, cách kết hợp các từ trong câu... Dẫn đến bộ cơ sở dữ liệu cho từng loại ngôn ngữ là khác nhau, các mô hình cũng không thể áp dụng lại cho từng ngôn ngữ, dẫn đến việc áp dụng tiếng Việt trong đề tài này là rất khó khăn.

Chính vì vậy, phạm vi của đề tài này sẽ nằm trong mức độ tiếng Anh. Với bộ cơ sở dữ liệu đã có sẵn, mục tiêu là tìm hiểu các giải thuật Học máy (Machine Learning) và Học sâu (Deep Learning), áp dụng vào để xác định sự tương đồng giữa hai câu tiếng Anh. Sau đó sẽ đánh giá kết quả và hiệu suất của mô hình bằng các phương pháp đánh giá phổ biến.

1.3 Các công việc liên quan

Trong bài báo [1] Siamese Recurrent Architectures for Learning Sentence Similarity - Jonas Mueller, Aditya Thyagarajan, Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16) một số công việc được đề cập đến như

- Đánh giá tương đồng về ngữ nghĩa, SemEval 2014, nhiều nhà nghiên cứu đã áp dụng các phương pháp cho một tập dữ liệu được dán nhãn có chứa các cặp câu liên quan đến kiến thức chung (SICK) (Marelli et al. 2014). Các phương pháp cho dữ liệu này đều sử dụng các đặc trưng không đồng nhất (vd: chồng chéo / tương tự , mô hình phủ định, thành phần câu / cụm từ) cũng như các tài nguyên bên ngoài(vd: Wordnet (Miller 1995)). Nhiều thuật toán được áp dụng (vd: SVM, Random Forest, k-Nearest Neighbors và các mô hình tập hợp).
- Zhao, Zhu và Lan (2014), học các biểu diễn không gian vectơ bằng cách sử dụng phân tích ngữ nghĩa tiềm ẩn cùng với nhiều tính năng thủ công khác.
- Gimpel và Lin (2015) đề xuất một biến thể mạng tích chập (ConvNet) phức tạp, tạo ra sự giống nhau trong câu bằng cách tích hợp các khía cạnh khác nhau qua nhiều kết cấu ở các quy mô khác nhau.

- Kiros và cộng sự. (2015) đề xuất mô hình skip-thoughts, mở rộng cách tiếp cận skip-gram của word2vec từ cấp độ từ đến mức câu. Mô hình này cung cấp mỗi câu vào bộ giải mã / mã hóa RNN (với kích hoạt GRU) nhằm cố gắng xây dựng lại các câu ngay trước và sau câu. Để thích ứng cách tiếp cận của họ với nhiệm vụ tương tự câu, Kiros và cộng sự đầu tiên chuyển một câu thông qua bộ mã hóa RNN (có trọng số được cố định sau khi đào tạo trên kho văn bản ban đầu) để có được một vectơ bỏ qua. Sau đó, một bộ phân loại riêng biệt được đào tạo về dữ liệu SICK bằng cách sử dụng các tính năng xuất phát từ sự khác biệt và sản phẩm giữa các vectơ bỏ qua cho cặp câu xuất hiện trong mỗi ví dụ đào tạo. Như trong khung mã hóa bộ giải mã của Sutskever, Vinyals và Le (2014), các thuộc tính ngữ nghĩa xuất hiện trong các biểu diễn bỏ qua như là các hiệu ứng gián tiếp thay vì được nhắm mục tiêu rõ ràng trong mục tiêu.
- Tai, Socher và Manning (2015) đề xuất các cây LSTM, khai quật hóa cấu trúc chuỗi nhạy cảm theo thứ tự của các LSTM tiêu chuẩn cho các cấu trúc mạng cấu trúc cây. Mỗi câu đầu tiên được chuyển đổi thành cây phân tích cú pháp (sử dụng trình phân tích cú pháp được đào tạo riêng) và Tree-LSTM kết hợp trạng thái ẩn của nó tại một nút cây đã cho từ từ tương ứng cũng như trạng thái ẩn của tất cả các nút con. Hy vọng là bằng cách phản ánh các thuộc tính cú pháp của một câu, mạng cấu trúc cây phân tích có thể truyền bá thông tin cần thiết hiệu quả hơn một kiến trúc bị hạn chế tuân tự. Mô hình này được điều chỉnh cho tương tự câu giống như của Kiros và cộng sự, trong đó việc biểu diễn các câu đầu vào hiện được tạo bởi các Tree-LSTM thay vì bỏ qua các suy nghĩ (Các biểu diễn của Tree-LSTM được đào tạo cùng với trình phân loại cuối cùng trên SICK tập dữ liệu).

Trong bài báo [2] Learning Text Similarity with Siamese Recurrent Networks - Paul Neculoiu, Maarten Versteegh and Mihai Rotaru, một số công việc được đề cập

- Collobert và cộng sự, 2011 sử dụng mạng tích chập đối với các nhiệm vụ NLP truyền thống. Áp dụng mạng neural trong dịch máy (Zou và cộng sự, 2013 – Cho và cộng sự, 2014), mô hình hỏi và trả lời (Weston và cộng sự, 2015). Trọng tâm của các mô hình này thường được đào tạo trên một lượng lớn dữ liệu được dán nhãn. Các kỹ thuật nhúng từ như word2vec (Mikolov et al., 2013) và Glove (Pennington et al., 2014) đã được sử dụng nhiều trong các mô hình như vậy, nhưng một số vượt xa cấp độ từ và thể hiện văn bản dưới dạng một chuỗi các ký tự (Kim et al., 2015; Ling và cộng sự, 2015).
- Việc học thể hiện thông qua các mạng neural đã nhận được sự quan tâm kể từ khi các bộ điều khiển tự động (Hinton và Salakhutdinov, 2006) được chứng minh là tạo ra các tính năng thỏa mãn hai thể hiện; rằng chúng là bất biến đối với sự khác biệt trong đầu vào không quan trọng đối với nhiệm vụ đó và có chọn lọc đối với sự khác biệt đó (Anselmi et al., 2015).
- Mạng Siamese (Bromley et al., 1993) là một kiến trúc cho việc học số liệu phi tuyến tính với thông tin tương tự. Mạng lưới Siamese tự học được các đại diện thể hiện tính bất biến và chọn lọc thông qua các thông tin rõ ràng về sự giống nhau giữa các cặp đối tượng. Ngược lại, bộ mã hóa tự động học bất biến thông qua việc giảm nhiễu và giảm kích thước trong lớp nút cỗ chai và tính chọn lọc chỉ thông qua điều kiện đầu vào phải được sao chép bởi phần giải mã của mạng. Ngược lại, một mạng lưới Siamese học được một đại diện bất biến và chọn lọc trực tiếp thông qua việc sử dụng thông tin tương tự và không giống nhau.
- Ban đầu được áp dụng để xác minh chữ ký (Bromley et al., 1993), kiến trúc Siamese đã được sử dụng rộng rãi trong các ứng dụng thị giác. Các mạng chập Siamese được sử dụng để tìm hiểu các số liệu tương tự phức tạp để xác minh khuôn mặt (Chopra et al., 2005) và giảm kích thước trên

các đặc điểm hình ảnh (Hadsell et al., 2006). Một biến thể của mạng Siamese, mạng bộ ba (Hoffer và Ailon, 2015), được sử dụng để tìm hiểu một biện pháp tương tự hình ảnh dựa trên dữ liệu xếp hạng (Wang et al., 2014).

- Trong các lĩnh vực khác, các mạng Siamese đã được áp dụng cho các nhiệm vụ đa dạng như mô hình hóa công việc không giám sát (Synnaeve et al., 2014; Thiolliere et al., 2015; Kamper et al., 2016; Zeghidour et al., 2015), học tập sở thích về thực phẩm (Yang et al., 2015) và phát hiện cảnh (Baraldi et al., 2015).
- Trong các ứng dụng NLP, các mạng Siamese với các lớp liên kết đã được áp dụng để phù hợp với các ứng dụng (Hu et al., 2014). Gần đây hơn, (Mueller và Thyagarajan, 2016) đã áp dụng các mạng tái phát của Siamese vào việc học hỏi về ngữ nghĩa.

Method	Accuracy
Illinois-LH (Lai and Hockenmaier 2014)	84.6
ECNU (Zhao, Zhu, and Lan 2014)	83.6
UNAL-NLP (Jimenez et al. 2014)	83.1
Meaning Factory (Bjerva et al. 2014)	81.6
Reasoning-based n-best (Lien and Kouylekov 2015)	80.4
LangPro Hybrid-800 (Abzianidze 2015)	81.4
SNLI-transfer 3-class LSTM (Bowman et al. 2015)	80.8
MaLSTM features + SVM	84.2

Hình 1: Dữ liệu SICK

1.4 Cơ sở dữ liệu

Đô án sử dụng bộ cơ sở dữ liệu pretrained word2vec, bao gồm 111052 từ, độ dài vector của mỗi từ là 300. Sử dụng bộ dữ liệu từ wiki simple vector [3]

```
you 0.13452 0.31968 0.38059 -0.15403 0.28954 -0.04673 -0.090418 -0.043943 -0.083994 -0.2539 0.23576 -0.010343 0.0016578 -0.009198 0.090325 -0.1662 -0.093698 0.05
their -0.10127 -0.33031 0.17901 -0.02247 0.051126 -0.13888 0.11275 -0.14798 0.013185 -0.090627 -0.15548 -0.042388 0.23333 -0.07021 0.12374 0.090048 0.15877 0.05
may -0.035019 -0.063687 0.22278 -0.15637 0.30156 0.38169 0.2629 -0.035664 -0.1108 0.21496 0.17432 -0.089959 -0.21086 -0.055328 0.062392 -0.36342 0.066253 -0.15
all 0.13876 -0.093298 0.3596 -0.26865 -0.078701 -0.021781 -0.081653 0.079407 0.19852 -0.0091618 0.13164 -0.17981 0.10223 0.0015087 -0.162 0.19258 0.1332 0.14661
she 0.019306 -0.064466 -0.027108 0.057219 0.044025 0.17173 -0.13562 -0.18474 0.31465 -0.060773 -0.019195 -0.052196 0.18386 0.054582 -0.10495 0.047438 0.30437 0.
d -0.18056 0.11102 -0.12808 -0.12277 0.19885 0.27643 -0.25436 -0.15317 -0.41195 -0.0673 -0.16432 0.073223 -0.42937 0.47348 -0.081927 0.15714 -0.14427 -0.087454
when -0.12967 0.11871 -0.048957 0.054052 -0.035872 0.38166 -0.04548 0.085794 -0.078636 0.046401 0.088003 -0.013756 0.032694 0.18002 -0.1881 -0.012753 -0.36563 0
after -0.11158 0.011757 -0.071095 0.027702 0.016509 0.11409 -0.02395 -0.0066769 0.079382 0.15466 -0.20386 -0.080926 -0.2621 0.24101 -0.051628 -0.038971 -0.09861
had 0.11301 -0.034263 0.070264 -0.21762 0.062643 -0.023857 0.23597 0.11426 0.25128 0.21494 -0.34077 0.039706 -0.12617 -0.0075052 -0.24517 -0.018317 0.31202 0.26
states 0.16051 -0.51393 0.4744 0.16848 0.07358 0.057597 -0.45953 -0.55241 -0.0089645 0.12157 0.045312 0.06378 -0.11624 0.30331 -0.074105 0.096048 0.085501 0.140
who -0.11414 0.075471 0.033006 -0.050179 -0.096984 0.44788 -0.021868 -0.01511 0.024378 0.16893 0.092236 0.18281 0.026578 0.098706 -0.073313 0.18138 -0.022148 0.
made 0.12328 -0.10729 0.34759 -0.2768 -0.11524 -0.2448 -0.19019 0.030367 -0.11674 -0.010706 -0.021496 -0.13172 -0.10548 0.20945 -0.05293 0.045137 0.1085 -0.0708
more 0.061941 -0.041957 0.23254 -0.44467 -0.1432 -0.078411 0.026987 -0.11115 0.026052 0.090268 0.28367 0.027793 -0.11754 -0.10271 0.2442 0.1066 -0.13317 0.15068
if 0.10398 0.1204 0.27154 -0.11151 -0.035356 0.29609 -0.32651 -0.086662 -0.19292 0.14826 -0.005874 0.062777 -0.14123 -0.010023 -0.24983 0.11067 -0.22249 -0.0174
born 0.39576 -0.0066076 -0.11712 -0.12901 0.18774 0.10836 -0.092798 -0.037368 -0.32632 -0.47914 -0.059084 0.36412 0.2562 0.11237 -0.22291 -0.16003 0.17544 0.053
used 0.28114 -0.097069 0.28046 -0.32653 0.034524 0.12961 0.051673 0.11451 -0.20437 0.068597 0.34911 -0.0067921 0.1088 0.029104 -0.14989 -0.10674 -0.2837 0.26366
```

Hình 2: Bộ cơ sở dữ liệu pretrained word2vec

Các câu hoàn chỉnh bao gồm 367372 câu đã được dán nhãn trong đó có 293897 từ dùng để train, 36738 từ dùng để test và 36738 từ dùng để validation.

Nguồn dữ liệu từ The Stanford Natural Language Inference (SNLI) Corpus [4].

Ví dụ về các câu:

- A person on a horse jumps over a broken down airplane
- A person is at a diner, ordering an omelette.

Kết quả: 0

- A person on a horse jumps over a broken down airplane.
- A person is outdoors, on a horse.

Kết quả: 1

CHƯƠNG 2 – CƠ SỞ LÝ THUYẾT

2.1 Word Embedding

Word Embedding là sự kết hợp của các mô hình ngôn ngữ và các kỹ thuật học đặc trưng trong Xử lý ngôn ngữ tự nhiên (NLP), trong đó các từ hoặc cụm từ được ánh xạ tới các vector số thực. Nó giúp xác định ngữ cảnh của một từ trong văn bản, sự tương đồng về nghĩa và cú pháp, quan hệ của một từ với các từ khác v.v...

Trong ngôn ngữ học, word embedding được thảo luận trong lĩnh vực nghiên cứu về phân phối ngữ nghĩa. Nó nhằm mục đích định lượng và phân loại sự tương đồng về ngữ nghĩa giữa các mục ngôn ngữ dựa trên các thuộc tính phân phối của chúng trong các mẫu dữ liệu ngôn ngữ lớn.

2.1.1 One-hot vector

Là vector chỉ chứa các giá trị dạng nhị phân. Phương pháp one-hot vector sẽ mã hoá (encoding) các từ trong từ điển thành một vector có chiều dài N (tổng số lượng các từ trong từ điển)

Ví dụ về các chuyển từ sang one-hot vector

- $V = \{tôi_1, đang_2, đi_3, tìm_4, một - nǚa_5, của_6, mình_7, đã_8, ăn_9, quả_10, táo_11\}$
- $S = 11$
- $tôi = [1 0 0 0 0 0 0 0 0 0 0]$
- $đang = [0 1 0 0 0 0 0 0 0 0 0]$
- ...
- $mình = [1 0 0 0 0 0 1 0 0 0 0]$
- $táo = [1 0 0 0 0 0 0 0 0 0 1]$

Một số hạn chế của phương pháp one-hot vector

- Độ dài của vector quá lớn, bằng với độ dài của từ điển
- Không xác định được tương quan ý nghĩa giữa các từ

Chính vì những hạn chế của phương pháp one-hot vector, người ta đề xuất ra mô hình word2vec, được xây dựng dựa trên cơ sở của one-hot vector.

2.1.2 Word2vec

Word2vec là một mạng neural với duy nhất 1 tầng ẩn (hidden layer), lấy đầu vào là một khối văn bản (corpus) lớn, đầu ra là một vector khoảng vài trăm chiều với mỗi từ duy nhất trong corpus được gắn với một vector tương ứng trong không gian. Các word vectors được xác định trong không gian vector sao cho những từ có chung ngữ cảnh được đặt gần nhau trong không gian.

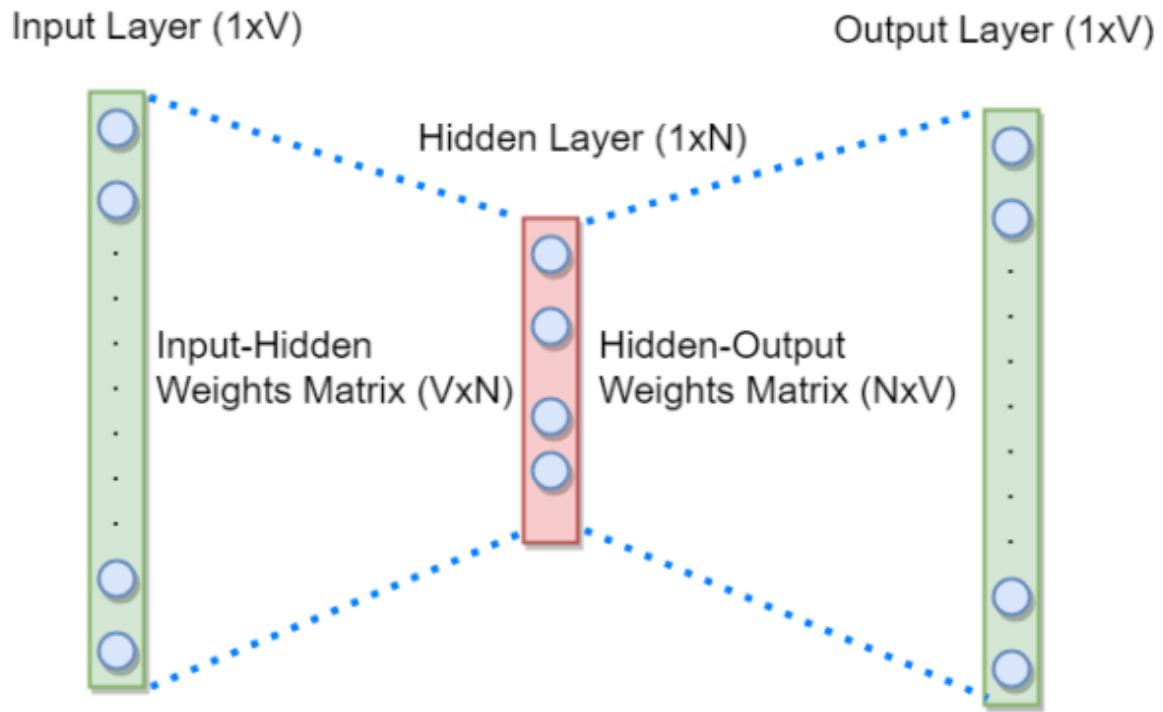
	Vua	Hoàng hậu	Phụ nữ	Công chúa
Hoàng gia	0.99	0.99	0.02	0.98
Nam tính	0.99	0.05	0.01	0.02
Nữ tính	0.05	0.93	0.999	0.94
Tuổi	0.7	0.6	0.5	0.1

Hình 2: Ví dụ về word2vec

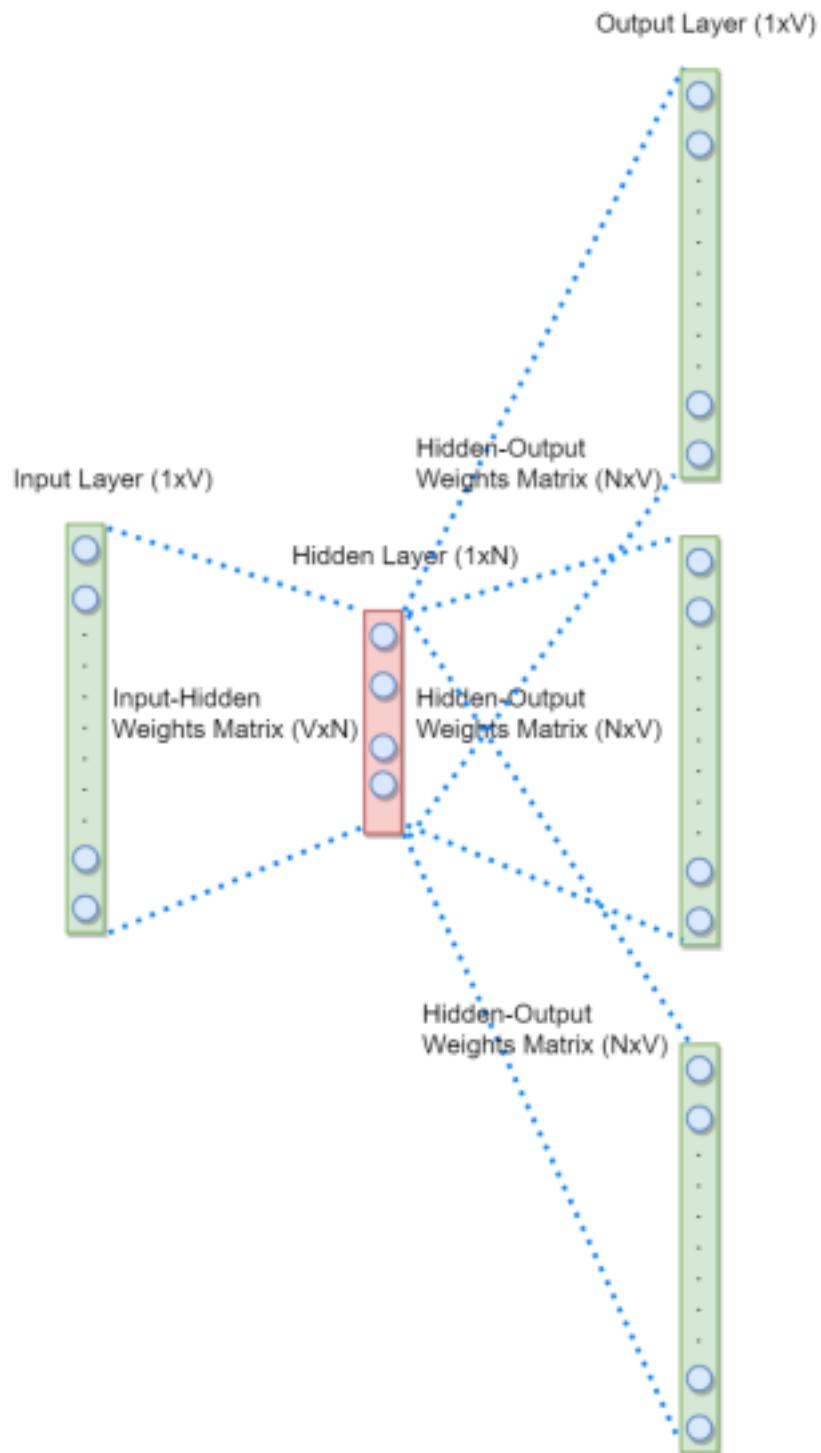
Word2vec có thể sử dụng một trong hai kiến trúc mô hình để tạo ra một thể hiện phân tán của các từ: bag-of-words (cbow) hoặc skip-gram.

Trong kiến trúc bag-of-words, mô hình sử dụng một cửa sổ của các từ ngữ cảnh xung quanh để dự đoán từ hiện tại. Thứ tự của các từ ngữ cảnh không ảnh hưởng đến dự đoán.

Trong kiến trúc skip-gram liên tục, mô hình sử dụng từ hiện tại để dự đoán cửa sổ xung quanh của các từ ngữ cảnh. Kiến trúc skip-gram cân nhắc các từ ngữ cảnh gần đó nặng hơn các từ ngữ cảnh xa hơn.



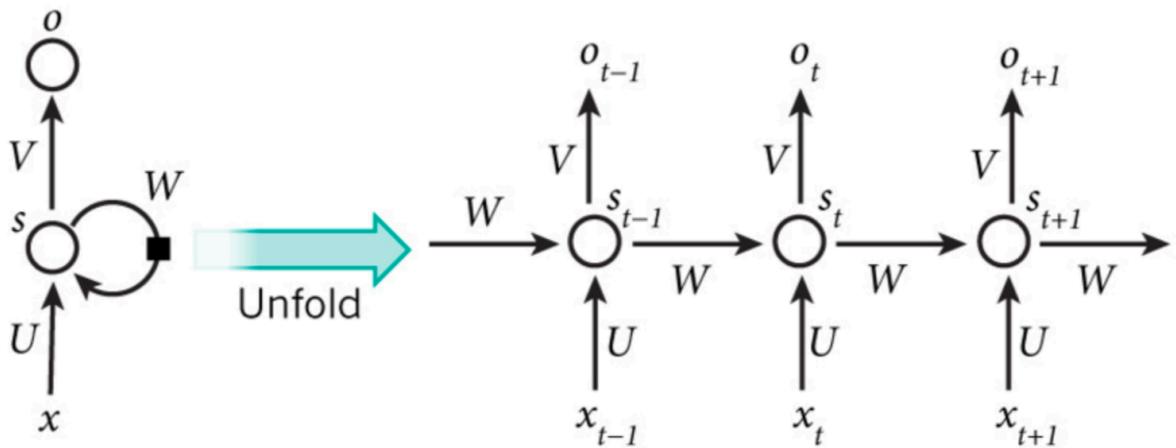
Hình 3: Mô hình Cbow



Hình 4: Mô hình skip-gram

2.2 Recurrent Neural Network (RNN)

2.2.1 Giới thiệu



Hình 5: Mô hình RNN

Recurrent Neural Network (RNN) là một trong những mô hình Deep learning được đánh giá có nhiều ưu điểm trong các tác vụ xử lý ngôn ngữ tự nhiên (Natural Language Processing). Tuy nhiên để nắm bắt ngay mô hình này không phải là điều đơn giản, cần có nắm vững lý thuyết về mạng Neural để có thể hiểu được cơ chế hoạt động của mô hình này.

Ý tưởng của RNN đó là thiết kế một mạng Neural sao cho có khả năng xử lý được thông tin dạng chuỗi (sequential information), ví dụ một câu là một chuỗi gồm nhiều từ. Recurrent có nghĩa là thực hiện lặp lại cùng một tác vụ cho mỗi thành phần trong chuỗi. Trong đó, kết quả đầu ra tại thời điểm hiện tại phụ thuộc vào kết quả tính toán của các thành phần ở những thời điểm trước đó.

Nói cách khác, RNN là một mô hình có trí nhớ (memory), có khả năng nhớ được thông tin đã tính toán trước đó. Không như các mô hình mạng Neural truyền thống đó là thông tin đầu vào (input) hoàn toàn độc lập với thông tin đầu ra (output). Về lý thuyết,

RNN có thể nhớ được thông tin của chuỗi có chiều dài bất kì, nhưng trong thực tế mô hình này chỉ nhớ được thông tin ở vài bước trước đó.

Quan sát sơ đồ biểu diễn RNN, thấy rằng mô hình này có khả năng biểu diễn mối quan hệ phụ thuộc giữa các thành phần trong chuỗi. Ví dụ, nếu chuỗi là một câu có 5 từ thì mạng Neural này sẽ unfold (dàn ra) thành mạng Neural có 5 tầng, mỗi tầng tương ứng với mỗi từ. Dưới đây là ý nghĩa các kí hiệu toán học:

- x_t là input tại thời điểm thứ t . Ví dụ, x_1 là vector của từ thứ hai trong câu (vị trí các từ được đánh số từ 0).
- s_t là hidden state (memory) tại thời điểm thứ t . s_t được tính dựa trên các hidden state trước đó kết hợp với input của thời điểm hiện tại $s_t = f(Ux_1 + Ws_{t-1})$.
- Hàm f là hàm nonlinearity (tanh hay ReLU). s_{t-1} là hidden state được khởi tạo là một vector không.
- o_t là output tại thời điểm thứ t . o_t là một vector chứa xác suất của toàn bộ các từ trong từ điển $o_t = \text{softmax}(Vs_t)$.

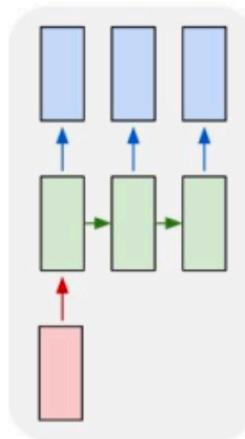
Không như mạng Neural truyền thống, tại mỗi tầng phải sử dụng một tham số khác. RNN chỉ sử dụng một bộ tham số (U, V, W) cho toàn bộ các bước.

2.2.2 Các dạng của RNN

Ngoài dạng cơ bản ở trên, RNN còn có 1 số dạng đặc biệt khác, được biến hóa để phù hợp với các yêu cầu bài toán khác nhau:

Dạng một - nhiều (One to many): thường dùng trong việc chú thích hình ảnh, với đầu vào là 1 tấm hình và đầu ra là 1 chuỗi các từ mô tả.

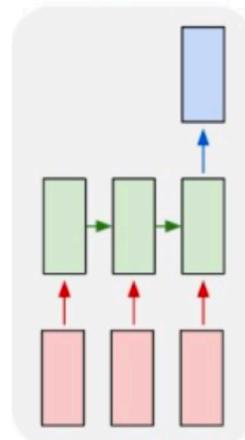
one to many



Hình 6: RNN dạng một - nhiều

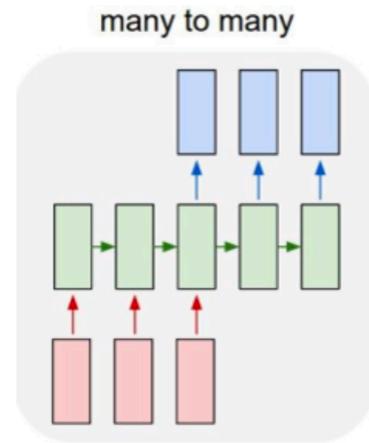
Dạng nhiều - một (Many to One): thường dùng để quyết định kết quả cuối cùng với một số dữ liệu đầu vào cho trước.

many to one



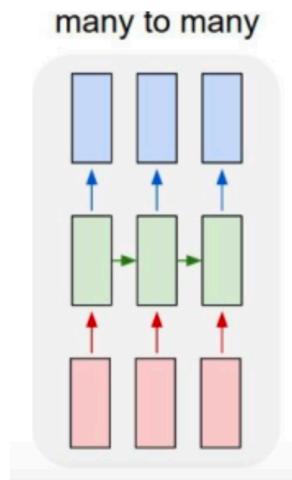
Hình 7: RNN dạng nhiều - một

Dạng nhiều - nhiều (Many to Many): thường dùng trong việc dịch thuật, chẳng hạn như 1 đoạn văn bản từ ngôn ngữ này sang ngôn ngữ khác.



Hình 8: RNN dạng nhiều - nhiều

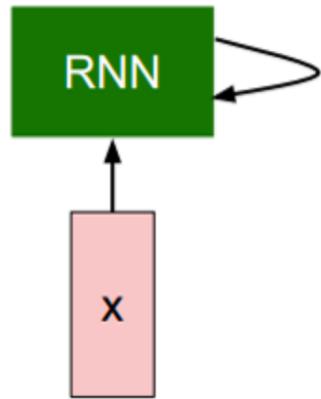
Một dạng khác của RNN nhiều - nhiều (Many to Many): thường dùng để phân loại video trên từng khung hình.



Hình 9: RNN dạng nhiều - nhiều

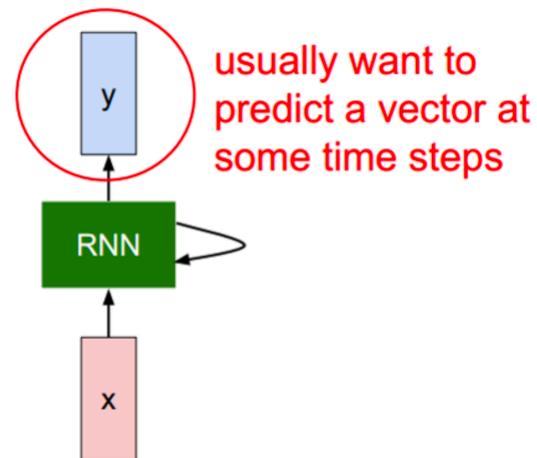
2.2.3 Phân tích RNN

Là ô xanh, nhận các vector đầu vào lại mỗi bước thời gian. Có các trọng số để xác định trạng thái của RNN tại mỗi bước thời gian, khi thay đổi các trọng số, RNN sẽ xử lý khác nhau đối với vector đầu vào.



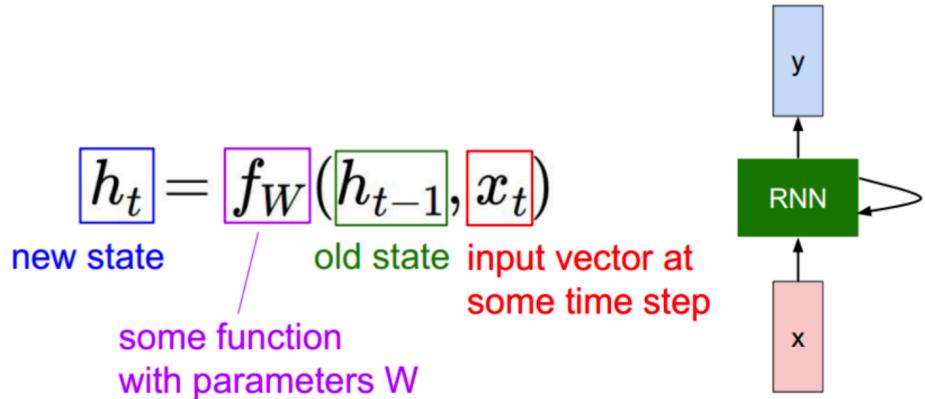
Hình 10: Phân tích RNN

Tiếp theo, RNN tạo ra các vector đầu ra dựa trên trạng thái của RNN.



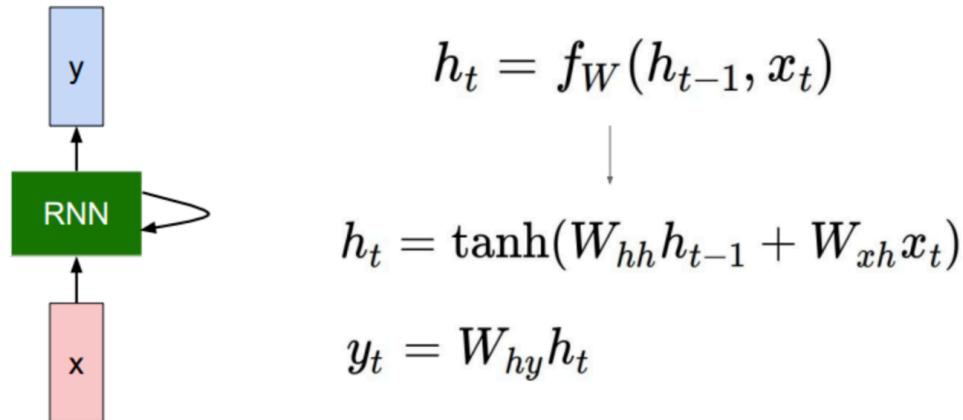
Hình 11: RNN tạo các vector đầu ra

Xử lý một chuỗi các vector X, áp dụng công thức RNN tại mỗi bước thời gian:



Hình 12: Công thức tổng quát của RNN

2.2.4 Công thức RNN



Hình 13: Công thức RNN

Chỉ có 1 trạng thái ẩn h duy nhất. Công thức RNN cho biết làm thế nào cập nhật trạng thái ẩn h như 1 giá trị của tầng ẩn trong bước thời gian trước và đầu vào hiện tại Xt.

Các ma trận trọng số Whh, Wxh sẽ phỏng đại các giá trị tầng ẩn của bước thời gian trước và đầu vào hiện tại Xt, sau đó cộng 2 giá trị lại với nhau và tính tanh() của giá trị đó. Sau đó cập nhật với trạng thái ẩn tại bước thời gian t.

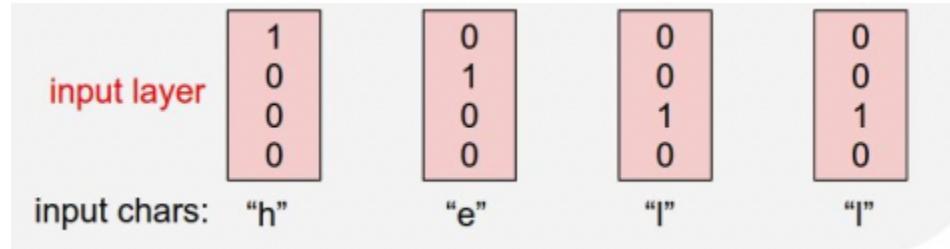
Sau đó sẽ dự đoán đầu ra dựa vào ht và ma trận trọng số Why.

2.2.5 Ví dụ RNN

Cho bộ từ vựng [h, e, l, o].

Chuỗi từ huấn luyện: “hello”

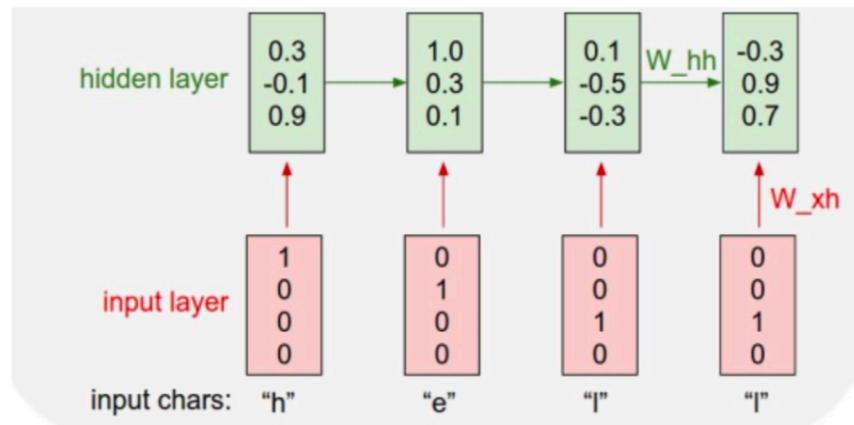
1) Truyền cùng lúc các ký tự vào RNN



Hình 14: Truyền ký tự vào RNN

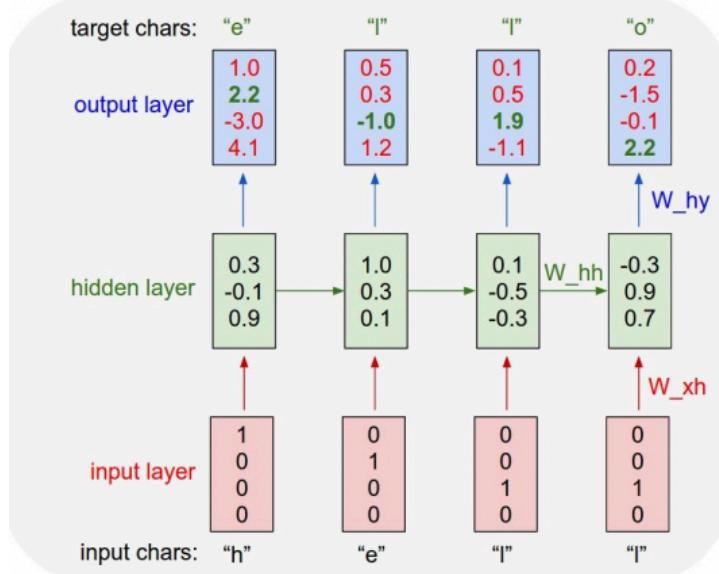
2) Bắt đầu bằng ký tự "h", áp dụng công thức để tính trạng thái RNN tại mỗi bước thời gian. Giả sử chỉ có 3 số trong trạng thái ẩn, sẽ biến chúng thành các vector đại diện, tại mỗi bước thời gian, tổng hợp các ký tự từ trước đến lúc đó.

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$



Hình 15: Áp dụng công thức RNN

3) Áp dụng RNN để huấn luyện tại mỗi bước thời gian, ký tự gì sẽ xuất hiện tiếp theo. Có 4 ký tự trong bộ từ vựng, vì vậy sẽ huấn luyện 4 số tại mỗi bước thời gian. VD: đã biết "e" sẽ đứng sau "h" nên 2.2 sẽ là số đúng, tương tự -1.0 cho "l", 1.9 cho "l" và 2.2 cho "o". Vì đã hết từ huấn luyện nên sau đó RNN sẽ gắn 1 "End token" cho ký tự cuối cùng là "o".



Hình 16: Tính toán xác suất của từ

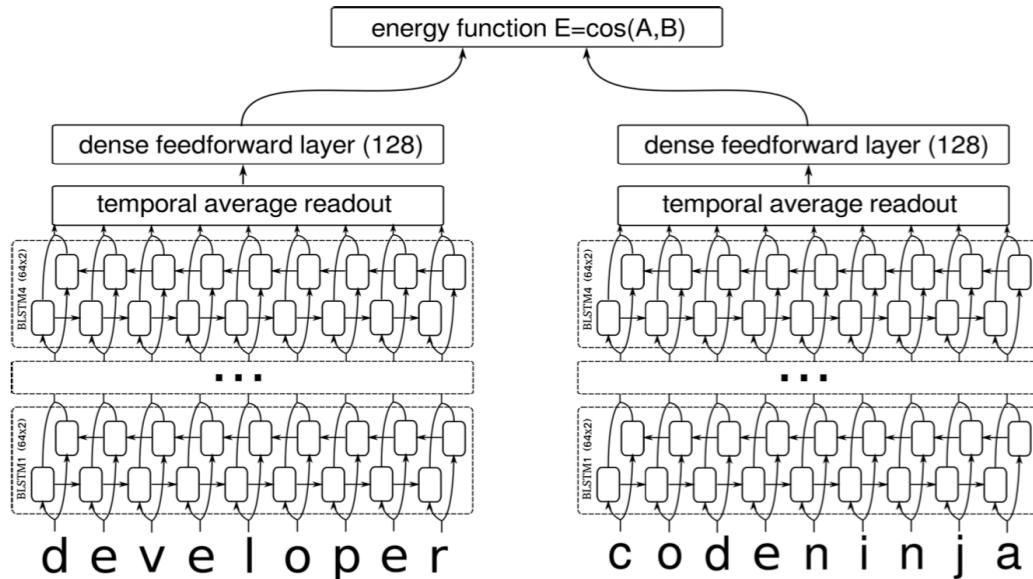
2.2.6 Huấn luyện RNN

Huấn luyện RNN tương tự như huấn luyện mạng Neural truyền thống. Cũng sử dụng đến thuật toán lan truyền ngược(backpropagation) nhưng có một chút tinh chỉnh. Gradient tại mỗi output không chỉ phụ thuộc vào kết quả tính toán của bước hiện tại mà còn phụ thuộc vào kết quả tính toán của các bước trước đó.

Ví dụ, để tính gradient tại thời điểm $t = 4$, cần lan truyền ngược 3 bước trước đó và cộng dồn các gradient này lại với nhau. Kỹ thuật này gọi là Backpropagation Through Time (BPPTT). Điểm hạn chế ở đây đó là tầng ẩn không có trí nhớ dài hạn. Vấn đề này còn gọi là vanishing/exploding gradient problem và LSTM được sinh ra để giải quyết vấn đề này.

2.3 Siamese Recurrent Neural Network

Các mạng Siamese là các mạng hai nhánh có trọng số ràng buộc, tức là, chúng bao gồm cùng một mạng được sao chép và hợp nhất với một hàm năng lượng. Hình dưới cho thấy tổng quan về kiến trúc mạng trong nghiên cứu này.



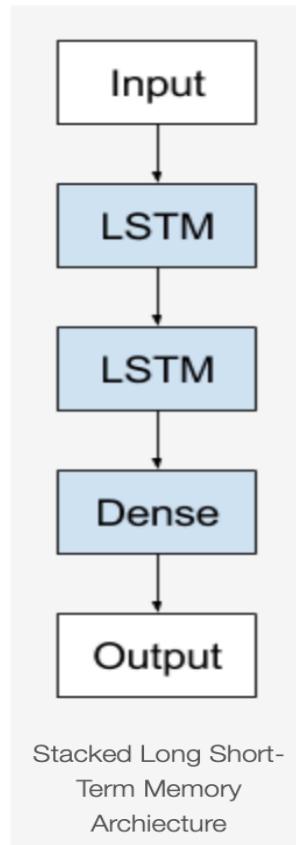
Hình 17: Mô hình Siamese RNN

Trong phương trình này và bên dưới hàm logistic được ký hiệu là

$$\sigma(x) = (1 + e^x)^{-1}$$

2.4 Stack Recurrent Neural Network

Mô hình Stack RNN (hay Stack LSTM) được xây dựng bằng cách xếp chồng các RNN (hay LSTM) lên nhau. Cách tiếp cận này có khả năng cho phép trạng thái ẩn ở mỗi cấp hoạt động ở các khoảng thời gian khác nhau.



Hình 18: Mô hình Stack RNN

2.5 Long Short-Term Memory (LSTM)

Deep learning là một kĩ thuật Machine Learning mạnh mẽ đang được nhiều người trong ngành biết đến và nghiên cứu. Kĩ thuật này nổi trội là do chúng thực hiện được hai việc cùng lúc: biểu diễn thông tin (represent problem/feature engineering) và học (learning). Do đó, kĩ thuật này còn được gọi là representation learning.

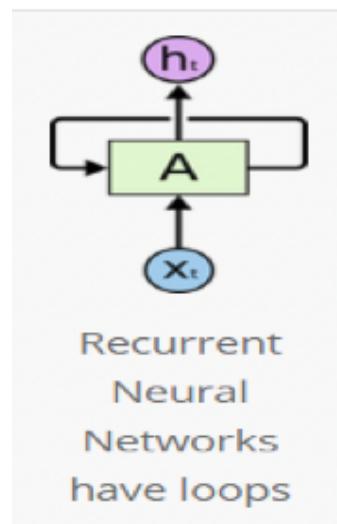
Bên cạnh các lĩnh vực đã gặt hái được nhiều thành công như xử lý ảnh số và video số, hay xử lý tiếng nói, Deep Learning cũng được áp dụng vào xử lý ngôn ngữ tự nhiên. Cụ thể trong đề tài này, Long short-term memory (LSTM) là mô hình cải tiến từ RNN cũng thuộc họ Deep Learning mà ta cần quan tâm.

2.5.1 Nhắc lại Recurrent Neural Network

Để đọc hiểu một câu hay một đoạn văn, con người chúng ta không quên hết những thông tin trước đó mà sẽ xâu chuỗi lại các dữ liệu đã đọc để làm rõ ý nghĩa cho thông tin hiện tại. Tương tự như việc ôn bài, việc làm này giúp các liên kết bên trong não được khắc sâu và tổ chức lại thông tin cho việc truy suất và xử lý thông tin trong tương lai được rõ ràng và nhanh chóng hơn.

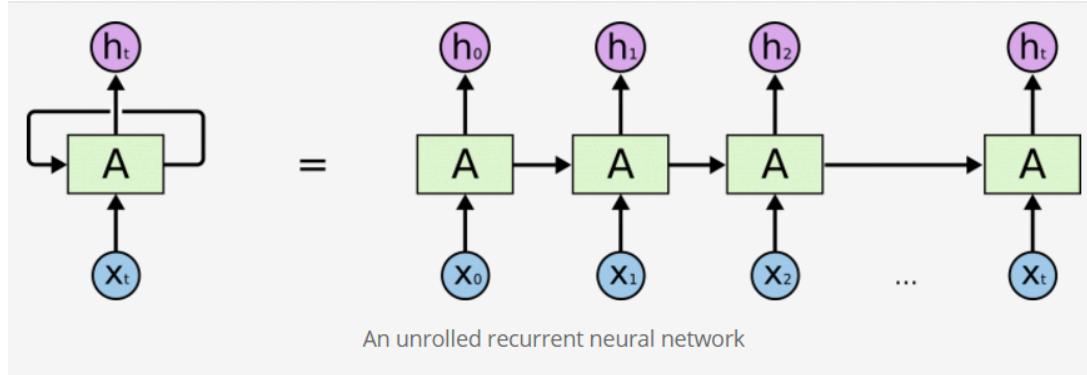
Các mô hình mạng Neural truyền thống không thể làm được điều này. Ví dụ, trong bài toán phân lớp sự kiện cho một đoạn video theo từng giây. Thật khó để một mô hình như vậy có thể dựa vào thông tin trước đó để phân lớp.

RNN giải quyết vấn đề này bằng cách tạo ra các mạng vòng lặp bên trong chúng, cho phép thông tin được lưu trữ lại cho các lần phân tích tiếp theo.



Hình 19: Mô hình RNN có vòng lặp

Trong biểu đồ trên, A nhận thông tin của x_t tại thời điểm t và phản hồi lại tương ứng kết quả đầu ra h_t tại thời điểm t . Có thể rã vòng lặp trên thành một tiến trình để dễ hình dung hơn.

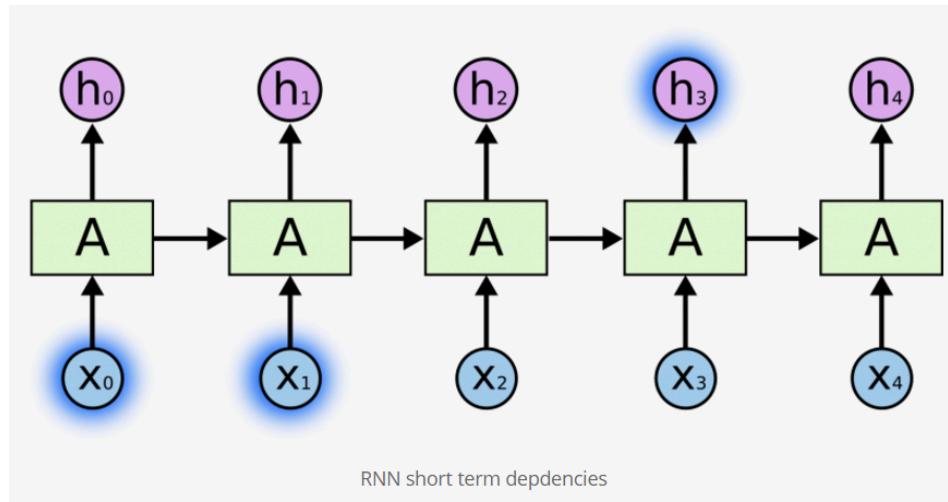


Hình 20: Tách các vòng lặp từ RNN

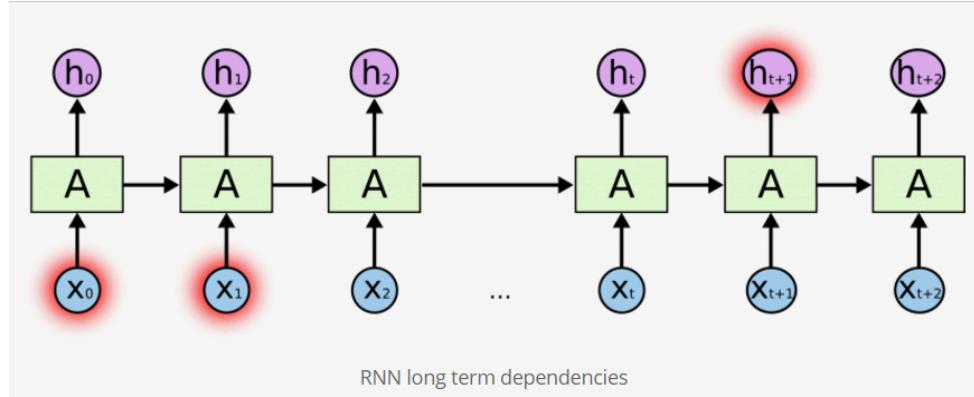
Thay vì chỉ nhận một đầu vào là x như các mạng Neural truyền thống, RNN hình thành nên một chuỗi các đầu vào A cùng với x_t tại thời điểm t .

2.5.2 Vấn đề phụ thuộc quá dài (Long-Term Dependencies)

Một trong những ý tưởng khởi thuỷ của RNN là kết nối những thông tin trước đó nhằm hỗ trợ cho các xử lý hiện tại. Nhưng đôi khi, chỉ cần dựa vào một số thông tin gần nhất để thực hiện tác vụ hiện tại. Ví dụ, khi cố gắng dự đoán từ tiếp theo dựa vào các từ trước đó. Nếu chúng ta dự đoán từ cuối cùng trong câu “đám mây bay trên bầu trời”, thì chúng ta không cần truy tìm quá nhiều từ trước đó, ta có thể đoán ngay từ tiếp theo sẽ là “bầu trời”. Trong trường hợp này, khoảng cách tới thông tin liên quan được rút ngắn lại.



Hình 21: Vấn đề phụ thuộc dài



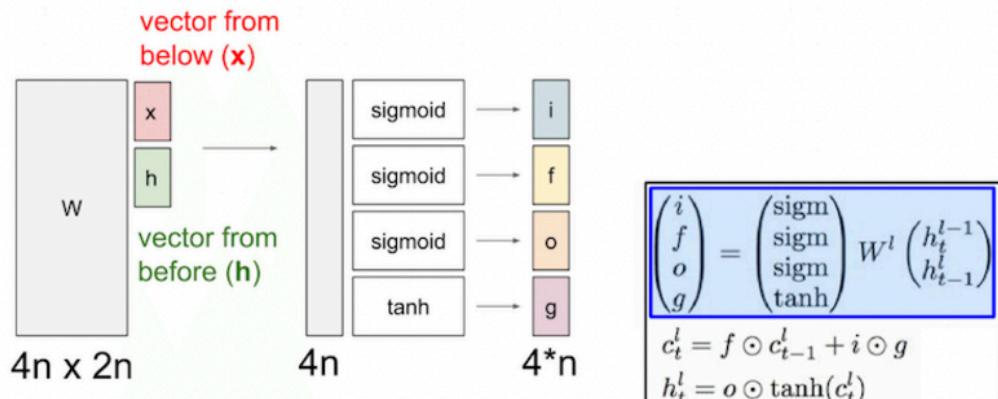
Hình 22: Vấn đề phụ thuộc dài

Về lý thuyết, RNN hoàn toàn có khả năng xử lý “long-term dependencies”, nghĩa là thông tin hiện tại có được là nhờ vào chuỗi thông tin trước đó. Thật không may, trong thực tế, RNN dường như không có khả năng này. Vấn đề này đã được ‘Hochreiter (1991) [German] and Bengio, et al. (1994]’ đưa ra như một thách thức cho mô hình RNN.

Trong những năm 1990, RNN phải đối diện với hai thách thức lớn đó là Vanishing và Exploding Gradients ảnh hưởng lớn đến hiệu suất của mô hình. Vấn đề này phát sinh trong quá trình huấn luyện.

Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]



LSTM formula

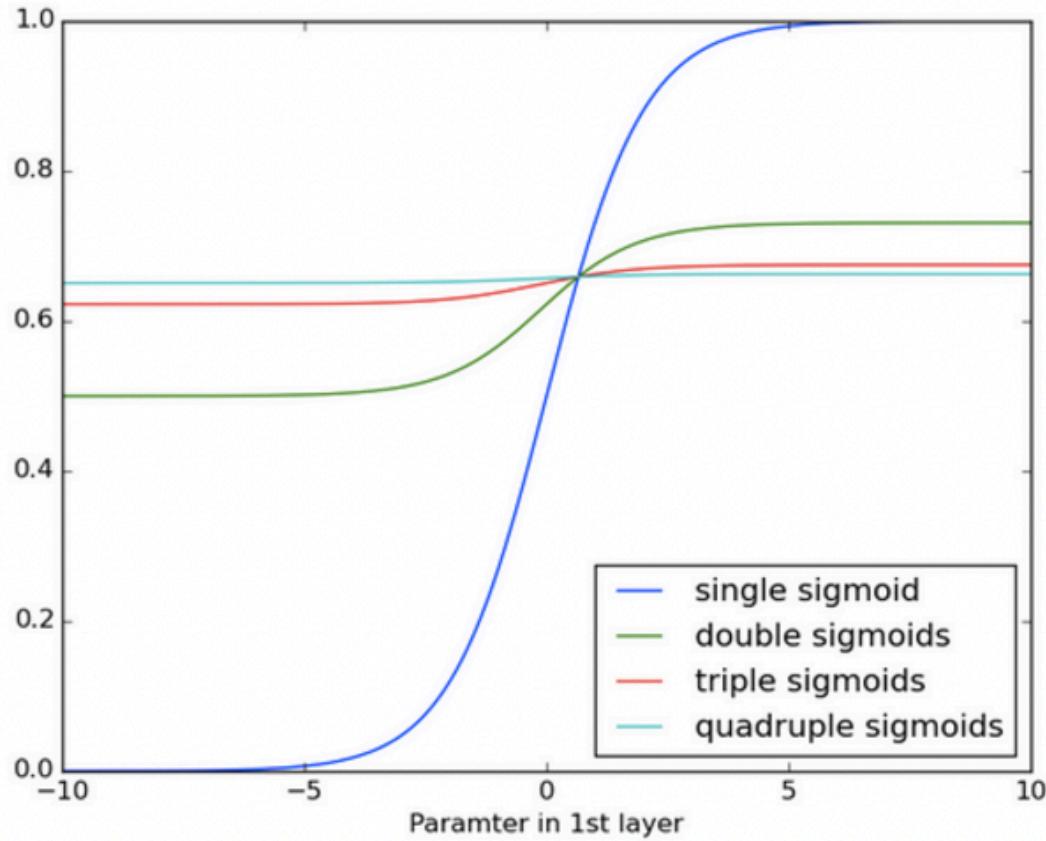
Hình 23: Công thức LSTM

Nếu trọng số của ma trận W nhỏ (trị riêng trọng số của ma trận nhỏ hơn 1.0), điều này sẽ dẫn đến trường hợp được gọi là Vanishing gradients khi gradient signal ngày càng nhỏ/tan biến theo quá trình huấn luyện, khiến cho quá trình tối thiểu hoá hàm lỗi hội tụ chậm hoặc dừng hẳn.

Ngược lại, nếu trọng số của ma trận W lớn (trị riêng trọng số của ma trận lớn hơn 1.0), điều này sẽ dẫn đến trường hợp được gọi là Exploding gradients khi gradient signal ngày càng bị phân tán trong quá trình huấn luyện, khi đó quá trình tối thiểu hoá hàm lỗi không hội tụ.

Mạng recurrent tìm kiếm và hình thành mối liên kết giữa kết quả cuối cùng(final output) và các sự kiện đầu vào(input event) thông qua nhiều bước trước khi kết thúc quá trình huấn luyện. Vấn đề đặt ra là các thông tin đầu vào(input) trước đó cần đặt trọng số tương ứng là bao nhiêu. Do vậy, đối với câu huấn luyện càng dài thì thông tin trước đó ngày càng bị nhiễu hoặc bị che lấp. Khi thực hiện quá nhiều phép toán nhân ma trận liên tục xuyên suốt chiều dài của chuỗi thì hiệu ứng Vanishing/Exploding sẽ xuất hiện.

Hình minh họa dưới đây cho thấy hiệu ứng khi áp dụng liên tiếp hàm sigmoid(). Dữ liệu thu được ngày càng tan biến dần cho đến lúc không còn nhận ra được nữa. Tương tự như gradient vanishing khi ta truyền dữ liệu qua nhiều tầng.

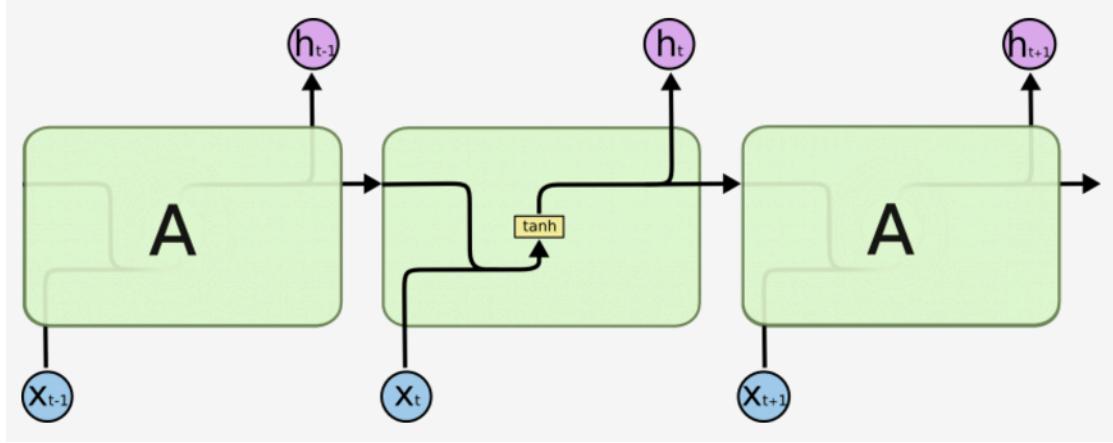


Hình 24: Áp dụng liên tiếp hàm sigmoid()

2.5.3 LSTM Network

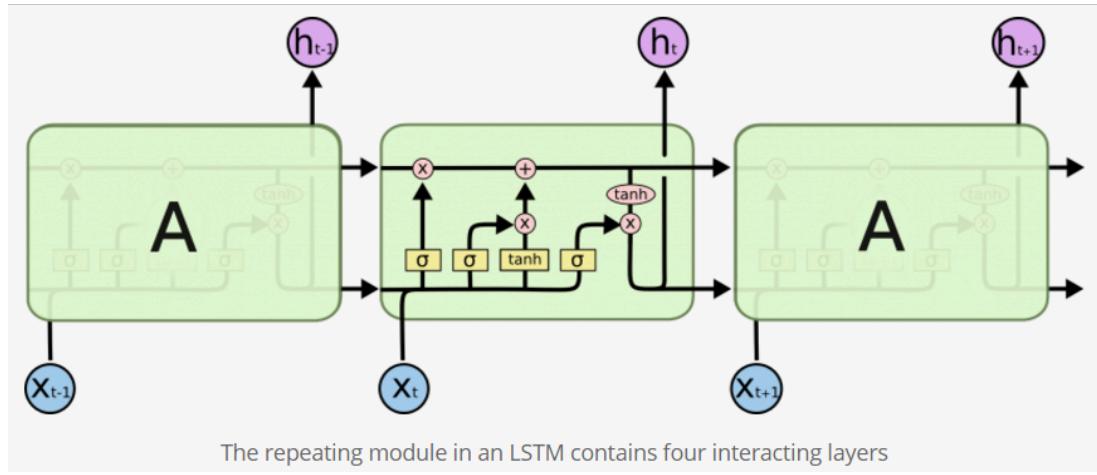
Mạng Long Short Term Memory – thường được gọi là “LSTM”, là trường hợp đặc biệt của RNN, có khả năng học long-term dependencies. Mô hình này được giới thiệu bởi Hochreiter & Schmidhuber (1997), và được cải tiến lại. Sau đó, mô hình này dần trở nên phổ biến nhờ vào các công trình nghiên cứu gần đây. Mô hình này có khả năng tương thích với nhiều bài toán nên được sử dụng rộng rãi ở các ngành liên quan.

LSTM được thiết kế nhằm loại bỏ vấn đề long-term dependency. Trước khi đi vào LSTM, cần quan sát lại mô hình RNN bên dưới, các tầng đều mắc nối với nhau thành các thành phần trong mạng Neural. Trong RNN chuẩn, thành phần lặp lại(repeating module) này có cấu trúc rất đơn giản chỉ gồm một hàm tanh.



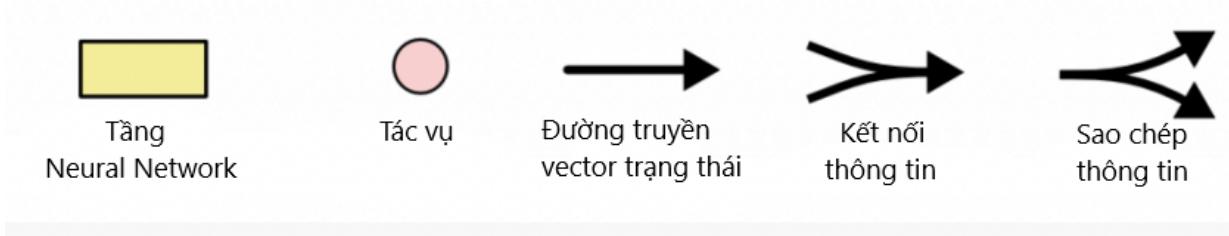
Hình 25: Cấu trúc RNN chuẩn

LSTM cũng có cấu trúc mắc xích tương tự, nhưng các thành phần lặp lại(repeating module) có cấu trúc khác hẳn. Thay vì chỉ có một tầng mạng Neural, có tới bốn tầng, tương tác với nhau theo một cấu trúc cụ thể.



Hình 26: Cấu trúc LSTM

Trước tiên, ta hãy làm quen với ký hiệu được sử dụng.



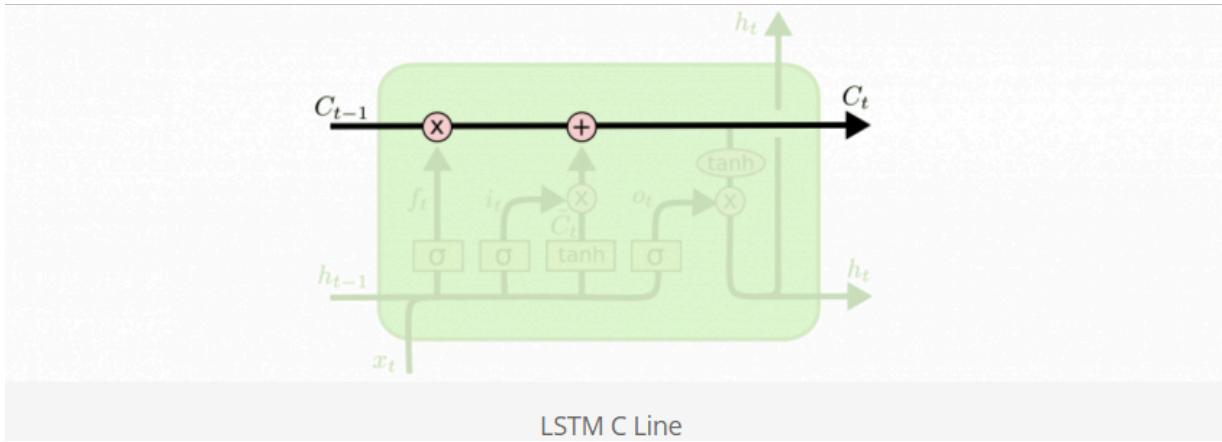
Hình 27: Ký hiệu được sử dụng

Ở biểu đồ trên, hình tròn nền hồng biểu diễn tác vụ sẽ thực hiện, ví dụ cộng vector. Hình hộp nền vàng là các tầng mạng Neural được huấn luyện. Đường kẻ giao nhau biểu thị kết nối các thông tin, trong khi đó đường rẽ nhánh biểu thị cho sự sao chép từ vị trí này sang vị trí khác.

2.5.4 Ý tưởng của LSTM

Có lẽ sau khi quan sát mô hình thiết kế của LSTM, sẽ nhận ra ngay, đây là một bảng mạch số, gồm các mạch logic và các phép toán logic trên đó. Thông tin, hay nói khác hơn là tần số của dòng điện di chuyển trong mạch sẽ được lưu trữ, lan truyền theo cách mà chúng ta thiết kế bảng mạch.

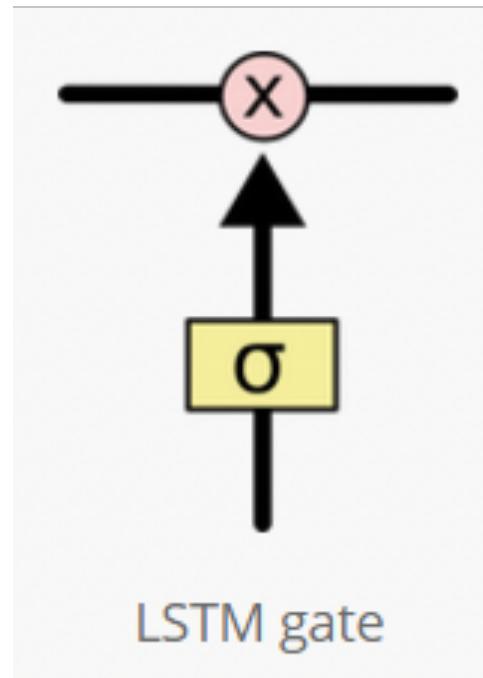
Mấu chốt của LSTM là cell state (đường trạng thái), đường kẻ ngang chạy dọc ở trên top diagram. Cell state giống như băng chuyền. Nó chạy xuyên thảng toàn bộ măc xích, chỉ một vài tương tác nhỏ tuyến tính (minor linear interaction) được thực hiện. Điều này giúp cho thông tin ít bị thay đổi xuyên suốt quá trình lan truyền.



Hình 28: Đường truyền trạng thái

LSTM có khả năng thêm hoặc bớt thông tin vào cell state, được quy định một cách cẩn thận bởi các cấu trúc gọi là cổng (gate).

Các cổng này là một cách (tùy chọn) để định nghĩa thông tin băng qua. Chúng được tạo bởi một hàm sigmoid().

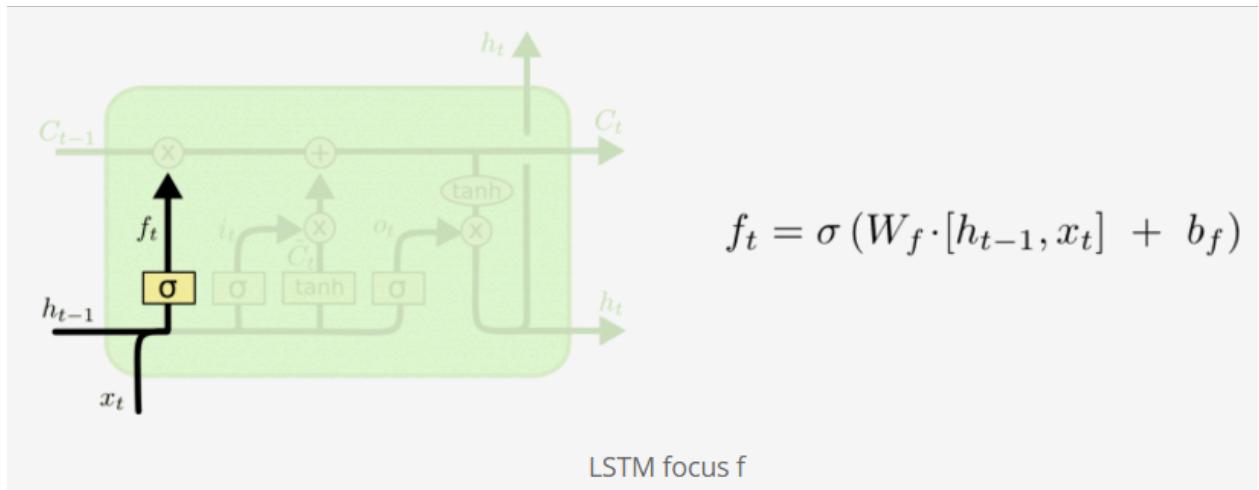


Hình 29: Cổng LSTM

Hàm sigmoid() có giá trị trong khoảng từ 0 đến 1, mô tả độ lớn thông tin được phép truyền qua. Nếu thu được 0, điều này có nghĩa là “không cho bất kỳ thông tin gì đi qua”, ngược lại nếu thu được giá trị là 1 thì có nghĩa là “cho phép mọi thông tin đi qua”. Một LSTM có ba công như vậy để bảo vệ và điều khiển cell state.

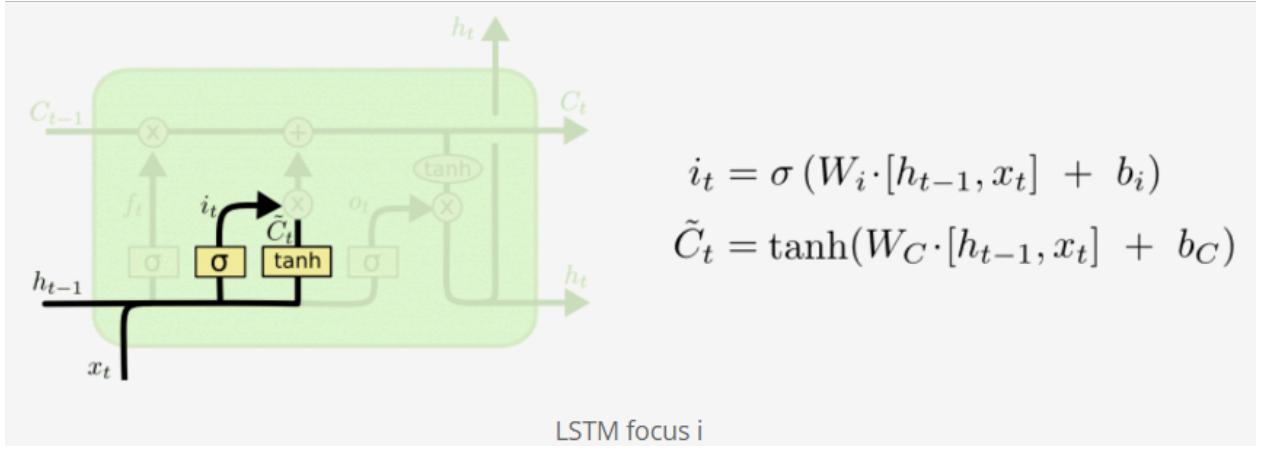
2.5.5 Phân tích mô hình LSTM

Bước đầu tiên của mô hình LSTM là quyết định xem thông tin nào chúng ta cần loại bỏ khỏi cell state. Tiến trình này được thực hiện thông qua một hàm sigmoid() gọi là “forget gate layer” – công quên. Đầu vào là h_{t-1} và x_t , đầu ra là một giá trị nằm trong khoảng [0, 1] cho cell state. 1 tương đương với “giữ lại thông tin”, 0 tương đương với “loại bỏ thông tin”.



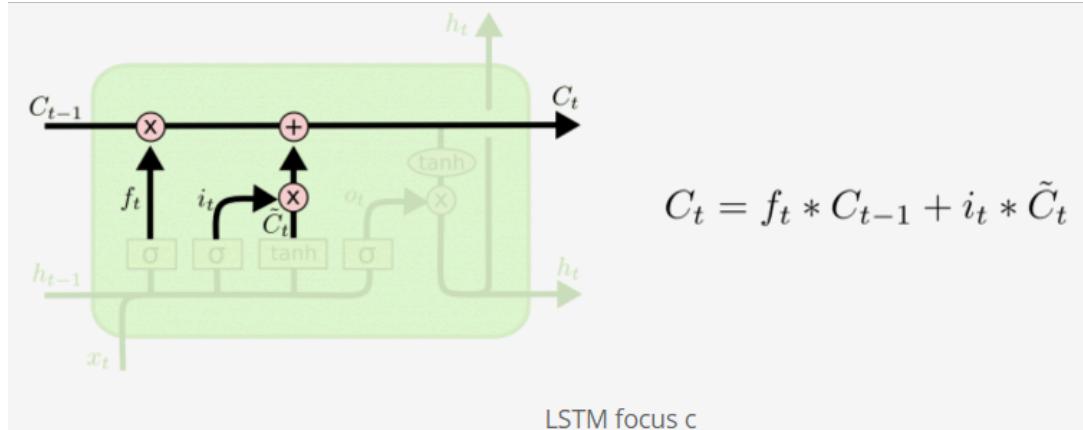
Hình 30: Đầu vào thông tin

Bước tiếp theo, ta cần quyết định thông tin nào cần được lưu lại tại cell state. Ta có hai phần. Một, hàm sigmoid() được gọi là “input gate layer” quyết định các giá trị chúng ta sẽ cập nhật. Tiếp theo, một hàm tanh tạo ra một vector ứng viên mới, C_t được thêm vào trong cell.



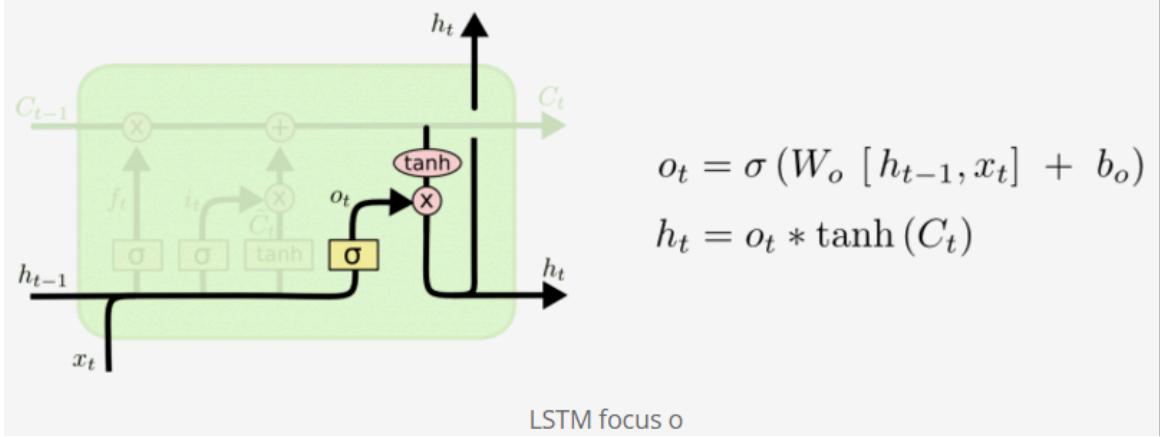
Hình 31: Quyết định thông tin

Ở bước tiếp theo, ta sẽ kết hợp hai thành phần này lại để cập nhật vào cell state. Lúc cập nhật vào cell state cũ, C_{t-1} , vào cell state mới C_t . Ta sẽ đưa state cũ hàm f_t , để quên đi những gì trước đó. Sau đó, ta sẽ thêm ($i_t * \tilde{C}_t$). Đây là giá trị ứng viên mới, có giãn (scale) số lượng giá trị mà ta muốn cập nhật cho mỗi state.



Hình 32: Cập nhật vào Cell State

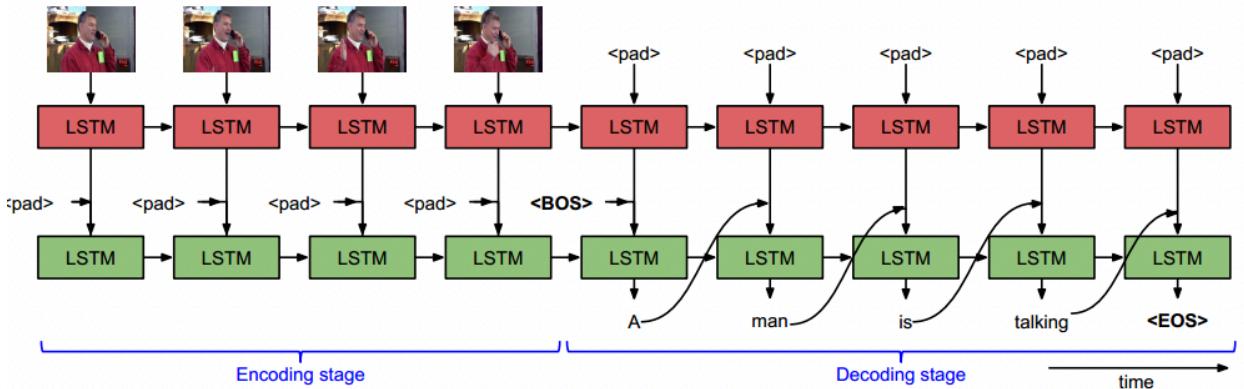
Cuối cùng, cần quyết định xem thông tin output là gì. Output này cần dựa trên cell state, nhưng sẽ được lọc bớt thông tin. Đầu tiên, áp dụng hàm sigmoid() để quyết định xem phần nào của cell state dự định sẽ đưa vào output. Sau đó, đẩy cell state qua tanh (đẩy giá trị vào khoảng -1 và 1) và nhân với một hàm sigmoid() Output, để giữ lại những phần ta muốn xuất output ra ngoài.



Hình 33: Thông tin đầu ra

2.6 LSTM nâng cao

Ý tưởng chính là để xuất một mô hình, trực tiếp lập một LSTM xếp chồng lên nhau, lần đầu tiên đọc vào hình ảnh và sau đó tạo ra các từ.



Hình 34: Mô hình LSTM nâng cao

Một ngăn xếp (stack) của hai LSTM, học một đại diện của một hình ảnh để giải mã nó thành một câu mô tả sự kiện. Các lớp LSTM hàng đầu (màu đỏ) mô hình hóa đặc trưng đầu vào hình ảnh. Lớp LSTM thứ hai (màu xanh lá cây) mô hình hóa ngôn ngữ cho đầu vào văn bản và các đại diện ẩn của hình ảnh. Sử dụng <BOS> (Begin of

sentence) để chỉ ra sự bắt đầu của câu và <EOS> (End of sentence) cho thẻ kết thúc câu. Các số 0 được sử dụng như một <pad> khi không có dữ liệu đầu vào ở bước tiếp theo.

Cách tiếp cận, được mô tả trong hình trên. Trước tiên mã hóa đầu vào đến một vector chiều dài cố định sử dụng một LSTM và sau đó sử dụng một LSTM khác để ánh xạ Vector vào một chuỗi các kết quả đầu ra, dựa vào một LSTM đơn cho cả giai đoạn mã hóa và giải mã. Điều này cho phép chia sẻ thông số giữa giai đoạn mã hóa và giải mã.

Lớp LSTM đầu nhận vào hình ảnh và mã hóa chúng trong khi LSTM thứ hai lớp nhận được biểu diễn ẩn (ht) và nối nó với các từ đầu vào 0 (zeros), mà nó sau đó mã hóa. Không có chi phí hao tổn khi truyền thông tin giữa 2 tầng LSTM. Sau đó, lớp LSTM thứ hai được đưa vào thẻ (<BOS>), sẽ nhắc nó bắt đầu giải mã thông tin thành một chuỗi các từ.

2.7 Hàm Softmax

Có một vài thủ thuật cần thiết để có được huấn luyện RNN/LSTM hiệu quả. Thấy rằng sự phân bố của những từ trong bộ từ vựng rất không đồng đều. Do đó, mô hình sử dụng một vài lần lặp đầu tiên để học các bias cho bộ phân loại Softmax sao cho nó dự đoán mọi từ một cách ngẫu nhiên với số liệu tần số thích hợp. Có thể thu được kết quả nhanh hơn trong giai đoạn huấn luyện bằng cách khởi tạo rõ ràng các bias của tất cả các từ trong bộ từ vựng (trong bộ phân loại Softmax) để ghi lại xác suất sự xuất hiện của chúng trong dữ liệu huấn luyện.

Vì vậy, với trọng số nhỏ và thiết lập bias một cách thích hợp, mô hình dự đoán ngay từ ngẫu nhiên theo phân bố xác suất của chúng. Sau khi thực hiện các thí nghiệm bổ sung với việc so sánh một RNN với một LSTM và thấy rằng LSTM luôn mang lại kết quả tốt hơn, nhưng huấn luyện lâu hơn.

2.8 Contrastive Loss Function

Các kích hoạt tại mỗi dấu thời gian của lớp LSTM cuối cùng được tính trung bình để tạo ra đầu ra có chiều cố định. Đầu ra này được chiếu thông qua một lớp tiếp liệu có

mật độ kết nối dày đặc. Đặt $f_W(x_1)$ và $f_W(x_2)$ là các hình chiếu của x_1 và x_2 trong không gian nhúng được tính bởi hàm mạng f_W , năng lượng của mô hình E_W được tính bởi

$$\text{- } E_W(x_1, x_2) = \frac{\langle f_W(x_1), f_W(x_2) \rangle}{\|f_W(x_1)\| \|f_W(x_2)\|}$$

Hàm mất mát trên tập dữ liệu $X = \{(x_1^{(i)}, x_2^{(i)}, y^{(i)})\}$

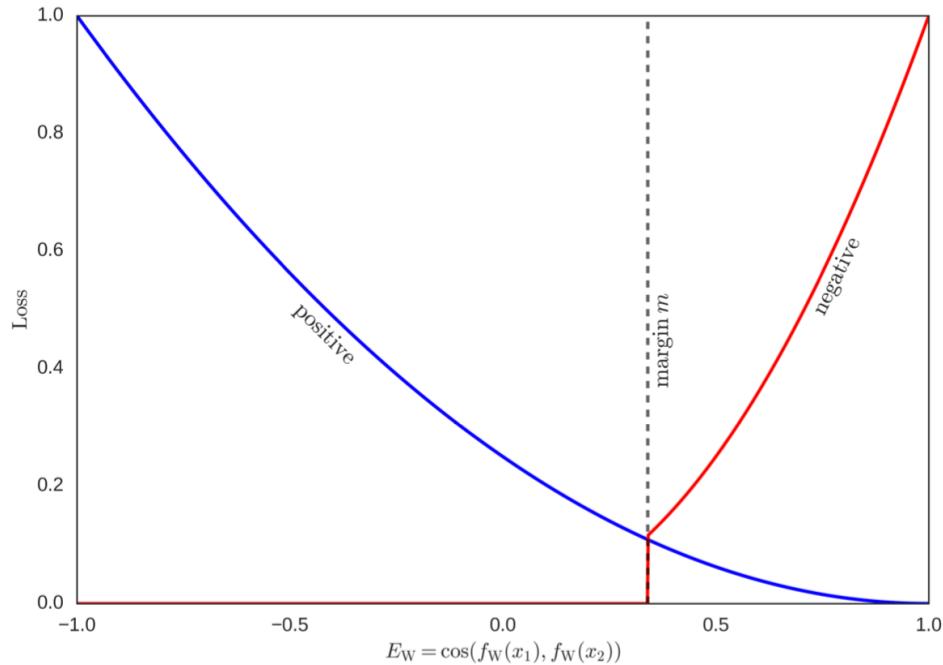
$$\text{- } L_W(X) = \sum_{i=1}^N L_W^{(i)}(x_1^{(i)}, x_2^{(i)}, y^{(i)})$$

Hàm mất mát được tính bởi

$$\text{- } L_W^{(i)} = y^{(i)}L(x_1^{(i)}, x_2^{(i)})_+ + (1 - y^{(i)})L(x_1^{(i)}, x_2^{(i)})_-$$

Các hàm mất cho các trường hợp tương tự và khác nhau:

$$\begin{aligned} \text{- } L(x_1, x_2)_+ &= \frac{1}{4}(1 - E_W)^2 && \text{Trường hợp tương tự} \\ \text{- } L(x_1, x_2)_- &= \begin{cases} E_W & \text{nếu } E_W < m \\ 0 & \text{khác} \end{cases} && \text{Trường hợp không tương tự} \end{aligned}$$



Hình 35: Contrastive loss function

2.9 Adam Optimal Function

Adam là một phương pháp thích ứng learning rate, nó tính toán learning rate cho các tham số khác nhau. Nó sử dụng moment thứ nhất và thứ hai của gradient để thích ứng learning rate cho từng tham số của mạng neural. Moment thứ n của một biến ngẫu nhiên được định nghĩa là giá trị mong đợi của biến đó với lũy thừa của n

- $m_n = E[X^n]$

Để ước tính các moment, Adam sử dụng các đường trung bình di chuyển theo cấp số nhân, được tính trên gradient được đánh giá trên mini-batch hiện tại:

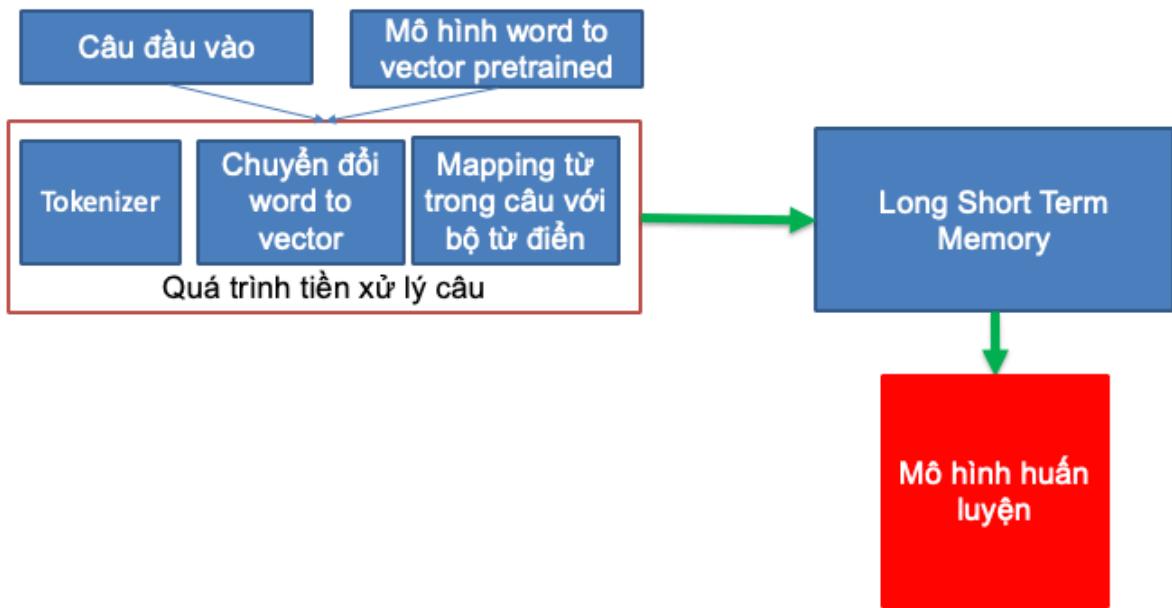
- $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$
- $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$

Với:

- m và v là đường trung bình động
- g là gradient của mini-batch hiện tại
- β là siêu tham số của giải thuật

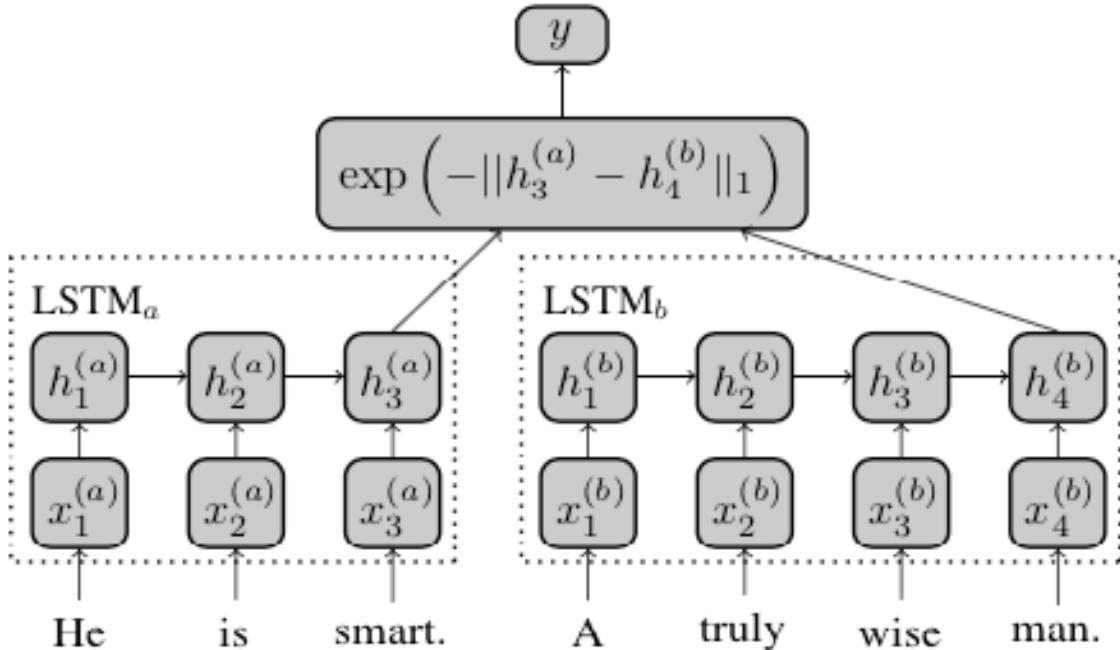
CHƯƠNG 3 – MÔ HÌNH GIẢI QUYẾT BÀI TOÁN

3.1 Mô hình huấn luyện



Hình 36: Mô hình huấn luyện

Mô hình huấn luyện được chia thành các bước: Tách từ trong câu => Áp dụng mô hình word2vec => chuyển đổi từ thành vector dạng số => áp dụng LSTM để huấn luyện.



Hình 37: Mô hình Manhattan LSTM (MaLSTM)

Mô hình LSTM hoạt động như sau: Đầu vào sẽ chuyển các từ của 2 câu, câu 1 ứng với mạng bên tay trái, và câu thứ 2 ứng với mạng bên tay phải. Mỗi bên sẽ trải qua quá trình học với 4 LSTM được xếp chồng lên nhau để học, đầu ra của mỗi tầng sẽ là đầu vào của tầng kế tiếp. Kết quả đầu ra ở mỗi tầng sẽ là tổng hợp kết quả của quá trình lan truyền thẳng và lan truyền ngược. Mạng sử dụng ở hai bên đều giống nhau và chia sẻ hyper parameter dung chung. Ở tầng cuối cùng của LSTM sẽ được đẩy vào một mạng truyền thẳng, chuẩn hoá vector thành 128. Sau khi đã đưa mạng truyền thẳng thì sẽ truyền vào không gian nhúng để tính toán, ở đó dựa vào nhãn của đầu vào quá trình train thì sẽ tính toán Entropy function dựa trên cosin.

Đặt $f_W(x_1)$ và $f_W(x_2)$ là các hình chiếu của x_1 và x_2 trong không gian nhúng được tính bởi hàm mạng f_W , năng lượng của mô hình E_W được tính bởi

$$- E_W(x_1, x_2) = \frac{\langle f_W(x_1), f_W(x_2) \rangle}{\|f_W(x_1)\| \|f_W(x_2)\|}$$

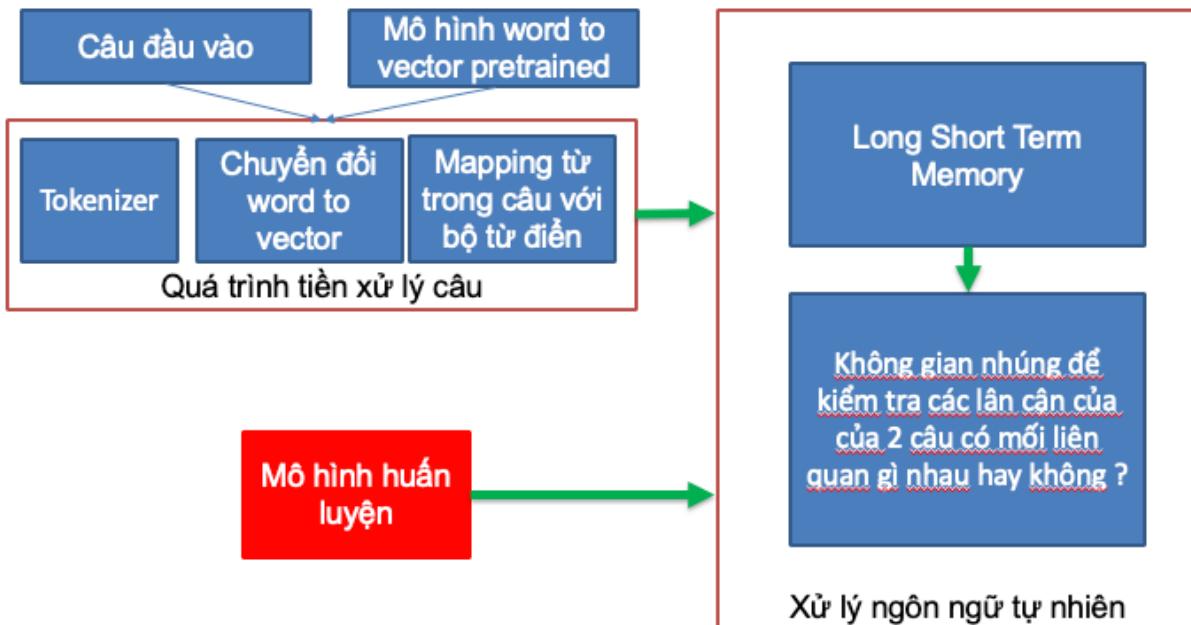
Những câu có ý nghĩa tương tự khoảng cách giữa x_1 và x_2

- Tương tự nhau thì khoảng cách đạt giá trị nhỏ nhất.
- Không tương tự nhau thì khoảng cách đạt giá trị lớn nhất.

CHƯƠNG 4 – KẾT QUẢ THỰC NGHIỆM

4.1 Phương pháp đánh giá

Mô hình đánh giá được xây dựng như sau



Hình 38: Mô hình kiểm tra

Kết quả

- Độ chính xác: 82, 16 %
- Contrastive Loss : 0.051
- Ví dụ về kết quả chạy:
 - a woman in a long green shirt, a yellow backpack, and a bottle of water is walking across a street as a couple follows close behind.
 - a woman with a backpack is walking across the street.

Kết quả: 1

- two guys are flying a kite.
- two men in white roll what appears to be bread or cakes.

Kết quả: 0

- So sánh với kết quả bài toán
 - Kết quả thực nghiệm từ dự án của bài báo:
 - Training time: (8 core cpu) = 1 complete epoch: 8min 10secs
(training requires atleast 50 epochs)
 - Contrastive Loss : 0.0477
 - Evaluation performance: similarity measure for 100,000 pairs
(8core cpu) = 2min 10secs - Accuracy 81%

CHƯƠNG 5 – KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Các hạn chế

Dựa trên những đánh giá từ kết quả thu được, có thể thấy những mặt hạn chế của ứng dụng như sau:

- Bộ cơ sở dữ liệu còn ít.
- Chưa xây dựng được hệ thống cho tiếng Việt do thiếu dữ liệu.
- Kết quả vẫn chưa đạt độ chính xác cao.

5.2 Những điều đã đạt được

Tuy ứng dụng còn nhiều mặt hạn chế, nhưng thông qua quá trình nghiên cứu và áp dụng, nhóm chúng em cơ bản đã thực hiện được một số điểm:

- Thu thập bộ cơ sở dữ liệu tiếng Anh.
- Mô phỏng lại hệ thống đánh giá tương đồng giữa 2 câu trong văn bản.
- Áp dụng các mô hình Deep Learning như LSTM cho bài toán đánh giá độ tương đồng.

5.3 Hướng phát triển

Những kết quả thu được chính là nền tảng quý báu, từ đó có thể phát triển thêm một số hướng cho đồ án như:

- Xây dựng bộ cơ sở dữ liệu tiếng Việt.
- Cải thiện các mô hình đã áp dụng để tăng tính chính xác cho hệ thống.
- Nghiên cứu, áp dụng kết hợp các mô hình đã có, cũng như các mô hình mới để cải thiện hiệu suất của hệ thống.

TÀI LIỆU THAM KHẢO

Tiếng Anh

1. Siamese Recurrent Architectures for Learning Sentence Similarity - Jonas Mueller, Aditya Thyagarajan, Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16).Morphological Transformations -
2. Learning Text Similarity with Siamese Recurrent Networks - Paul Neculoiu, Maarten Versteegh and Mihai Rotaru, Proceedings of the 1st Workshop on Representation Learning for NLP, pages 148–157, Berlin, Germany, August 11th, 2016.
3. <https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md>
4. <https://nlp.stanford.edu/projects/snli/>