ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

PHẠM HÙNG

HƯỚNG TIẾP CẬN DỰA TRÊN HỌC MÁY CHO BÀI TOÁN TRÍCH XUẤT THÔNG TIN QUAN ĐIỂM

LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN

HÀ NỘI – 2017

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

PHẠM HÙNG

HƯỚNG TIẾP CẬN DỰA TRÊN HỌC MÁY CHO BÀI TOÁN TRÍCH XUẤT THÔNG TIN QUAN ĐIỂM

Ngành: Công nghệ thông tin
Chuyên ngành: Kỹ thuật phần mềm
Mã số: 60480103

LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN

NGƯỜI HƯỚNG DẪN KHOA HỌC: TS. NGUYỄN VĂN VINH

HÀ NỘI - 2017

LỜI CAM ĐOAN

Tôi là Phạm Hùng, học viên lớp Kỹ Thuật Phần Mềm K21 xin cam đoan báo cáo luận văn này được viết bởi tôi dưới sự hướng dẫn của thầy giáo, tiến sĩ Nguyễn Văn Vinh. Tất cả các kết quả đạt được trong luận văn này là quá trình tìm hiểu, nghiên cứu của riêng tôi. Trong toàn bộ nội dung của luận văn, những điều được trình bày là kết quả của cá nhân tôi hoặc là được tổng hợp từ nhiều nguồn tài liệu khác. Các tài liệu tham khảo đều có xuất xứ rõ ràng và được trích dẫn hợp pháp.

Tôi xin hoàn toàn chịu trách nhiệm và chịu mọi hình thức kỷ luật theo quy định cho lời cam đoan của mình.

Hà Nội, ngày ….. tháng …… năm 2017

Người cam đoan

Phạm Hùng

LỜI CẢM ƠN

Lời đầu tiên, tôi xin bày tỏ sự cảm ơn chân thành đối với thầy giáo TS. Nguyễn Văn Vinh – giáo viên hướng dẫn trực tiếp của tôi. Thầy Vinh đã giúp tôi tiếp cận những kiến thức về trí tuệ nhân tạo từ những thuật toán cơ bản đến nâng cao trong quá trình nghiên cứu và hoàn thiện luận văn thạc sĩ.

Tôi cũng xin gửi lời cảm ơn tới các thầy cô trong khoa Công nghệ thông tin, trường Đại học Công Nghệ, Đại học Quốc gia Hà Nội đã hướng dẫn, chỉ bảo và tạo điều kiện cho chúng tôi học tập và nghiên cứu tại trường trong suốt thời gian qua.

Mặc dù đã cố gắng hoàn thành luận văn nhưng chắc chắn sẽ không tránh khỏi những sai sót, tôi kính mong nhận được sự thông cảm và chỉ bảo của các thầy cô và các bạn.

Tôi xin chân thành cảm ơn.

MỤC LỤC

BẢNG CÁC TỪ VIẾT TẮT

Viết tắt	Đầy đủ	Ý nghĩa
RNN	Recurrent Neural Network	Mạng neural hồi quy
ANN	Artificial Neural Network	Mạng neural nhân tạo
NLP	Natural Language Processing	Xử lý ngôn ngữ tự nhiên
LSTM	Long short-term memory	Mạng neural cải tiến giải quyết vấn đề phụ thuộc từ quá dài
CNN	Convolutional Neural network	Mạng neural tích chập
SVM	Support Vector Machine	Máy vector hỗ trợ

DANH MỤC HÌNH VẼ

TÓM TẮT NỘI DUNG

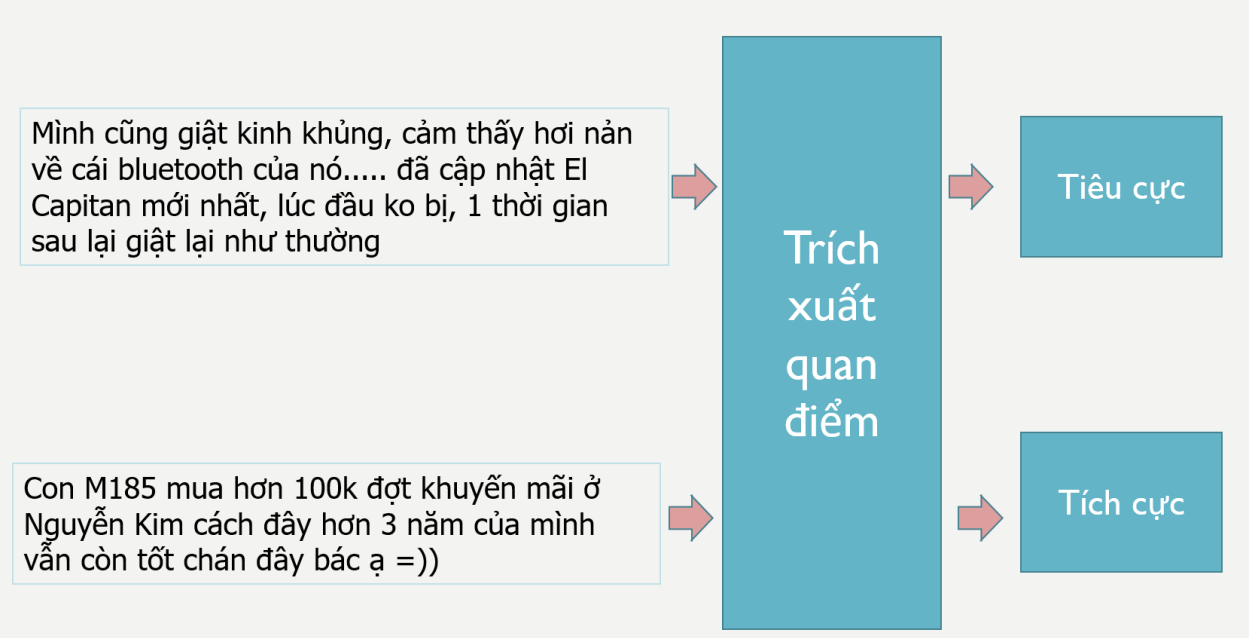
Mạng neural hồi quy RNN được áp dụng rất rộng rãi trong các bài toán xử lý ngôn ngữ tự nhiên NLP. Do mạng hồi quy RNN mô hình hóa được bản chất của dữ liệu trong NLP như đặc tính chuỗi và sự phụ thuộc lẫn nhau giữa các thành phần theo thứ tự. Ngoài ra, do năng lực tính toán của máy tính ngày càng mạnh mẽ nên đã thực hiện hóa được việc huấn luyện mạng neural hồi quy nhiều tham số vốn yêu cầu nhiều bước tính toán hơn so với mạng neural thông thường. Do đó, việc áp dụng mạng RNN có thể coi là một bước đột phá trong xử lý ngôn ngữ.

Luận văn sẽ trình bày về lý thuyết mạng neural RNN và cải tiến của nó là LSTM cùng với một số thuật toán học máy quan trọng trong quá trình xử lý dữ liệu ngôn ngữ. Cuối cùng, luận văn sẽ mô tả việc áp dụng và kết quả khi sử dụng mô hình LSTM trong bài toán trích xuất thông tin quan điểm. Thuật toán sẽ được đánh giá dựa trên hai tập dữ liệu tiếng Anh và tiếng Việt.

MỞ ĐẦU

Trong thời đại hiện nay, nhằm phục vụ cho nhu cầu cuộc sống ngày càng cao của con người, các sản phẩm và dịch vụ cũng có bước phát triển rất mạnh mẽ. Có thể kể đến từ những sản phẩm đáp ứng nhu cầu thường ngày của con người như quần áo, sách, tạp chí, đồ dùng cá nhân cho đến những nhu cầu cao hơn về thị hiếu, du lịch, thẩm mĩ. Với mỗi loại sản phẩm và dịch vụ hiện tại cũng rất phong phú về chủng loại, chất lượng, cạnh tranh về giá cả tới từ nhiều nhà cung cấp khác nhau. Do đó, việc duy trì phát triển một sản phẩm dịch vụ có được mạng lưới người sử dụng rộng rãi đòi hỏi rất nhiều công sức. Một trong những phương pháp cơ bản và hiệu quả nhất là lắng nghe ý kiến phản hồi của khách hàng về sản phẩm dịch vụ. Dựa trên những ý kiến phản hồi này, nhà cung cấp sản phẩm dịch vụ có thể đánh giá được thị hiếu của sản phẩm, hiệu quả của chiến lược marketing quảng bá sản phẩm hay điều chỉnh sản phẩm phù hợp để đạt được hiệu quả kinh doanh tốt nhất. Công việc trên có tên gọi là trích xuất thông tin quan điểm của người dùng. Đây là bài toán cơ bản nhưng có ứng dụng rất lớn trong cuộc sống.

Cùng với sự phát triển của thiết bị di động và mạng internet, người dùng có rất nhiều kênh để tương tác với nhà cung cấp dịch vụ. Có thể kể đến các kênh truyền thống như email, điện thoại, fax cho đến các hình thức mới hơn như viết phản hồi trên các trang mạng xã hội, viết bài review sản phẩm, phản hồi ngay trên trang giới thiệu sản phẩm hay trên các diễn đàn. Từ các nguồn kể trên, dữ liệu được thu thập lại dưới dạng văn bản. Từ dữ liệu dạng văn bản, luận văn sẽ trình bày phương pháp áp dụng học máy để xử lý thông tin văn bản nhằm trích xuất được thông tin quan điểm của người dùng.



Hình 0.1: Trích xuất thông tin quan điểm

Luận văn của tôi được chia thành các phần sau:

Chương 1: Trình bày tổng quan về bài toán trích xuất thông tin quan điểm và một số khái niệm liên quan. Đồng thời, tôi trình bày những thách thức của việc trích xuất thông tin quan điểm sử dụng mô hình học máy.

Chương 2: Trình bày các phương pháp và một số thuật toán sử dụng cho bài toán trích xuất thông tin quan điểm. Trong đó, tôi sẽ trình bày kỹ về mô hình mạng Recurrent Neural Network (RNN), mô hình tiên tiến đang được áp dụng cho việc xử lý thông tin dạng chuỗi như văn bản.

Chương 3: Trình bày việc áp dụng mô hình RNN cho bài toán phân tích quan điểm.

Chương 4: Kết quả một số thử nghiệm.

Chương 5: Kết luận.

CHƯƠNG 1: TỔNG QUAN VỀ BÀI TOÁN

1.1 Khái niệm quan điểm

Trong xã hội hiện đại, việc nêu và thể hiện ý kiến cá nhân là một phần của tự do ngôn luận. Với mỗi người được đào tạo khác nhau và có những cách tiếp cận khác nhau đối với một vấn đề sẽ nảy sinh ra nhiều chiều trong ý kiến, tư tưởng. Đó chính là quan điểm. Quan điểm được xây dựng chủ yếu từ ba yếu tố là thái độ, cảm xúc và ý kiến về một đối tượng. Đối tượng ở đây có thể là các cá nhân, các sự việc, sự vật hay là chất lượng dịch vụ, sản phẩm, chủ đề.

1.2 Bài toán trích xuất thông tin quan điểm

Bài toán trích xuất thông tin quan điểm dựa trên các thông tin phản hồi của người sử dụng nhằm phân loại phản hồi đó là tích cực hay tiêu cực. Thông tin phản hồi của người dùng được tổng hợp dưới dạng văn bản từ nhiều nguồn khác nhau như trên trang bán hàng, Facebook, hệ thống chợ của Google hay Apple. Dựa trên đánh giá của người dùng, kết quả của chiến lược marketing hay quảng bá sản phẩm được xác định là có hiệu quả hay không.

Bài toán trích xuất thông tin quan điểm (sentiment analysis) là một lĩnh vực nghiên cứu về các ý kiến, quan điểm, đánh giá, thái độ và cảm xúc của con người về một đối tượng. Trích xuất thông tin quan điểm thu hút được sự quan tâm lớn của cộng đồng nghiên cứu nói chung và cộng đồng xử lý ngôn ngữ tự nhiên nói riêng bởi hai yếu tố:

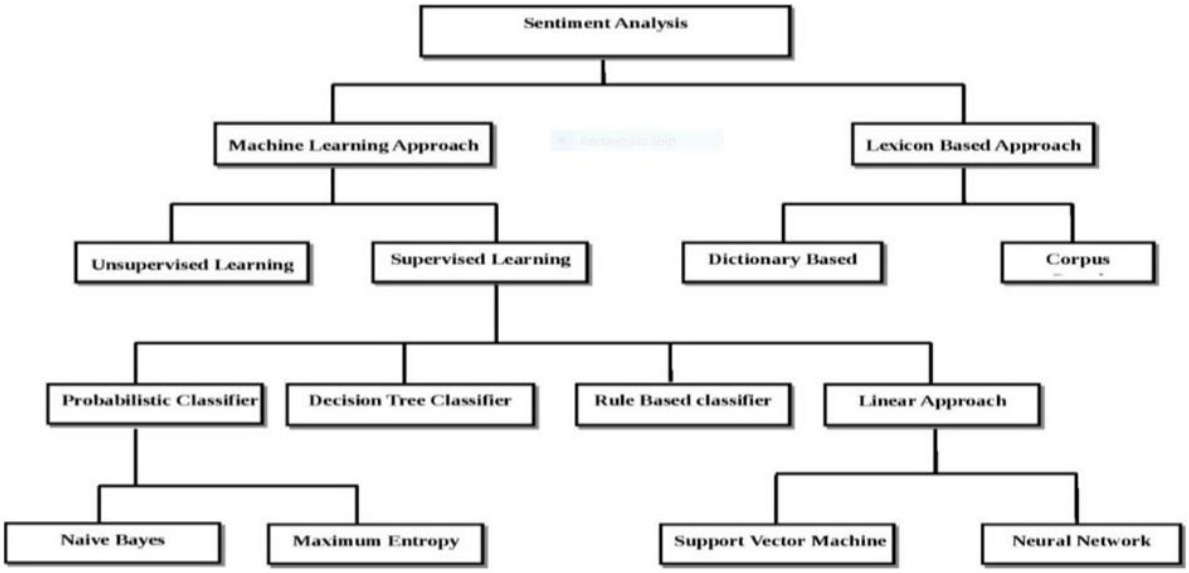
Thứ nhất, do sự bùng nổ thông tin và mạng xã hội nên con người có thể tự do chia sẻ ý kiến cảm nghĩ. Trong lịch sử loài người, đây là thời điểm lượng thông tin nói chung và thông tin về ý kiến quan điểm nói riêng phát triển rất nhanh và mạnh. Lượng thông tin chia sẻ trên mạng xã hội là khổng lồ .Theo số liệu của The Social Skinny [14], Facebook đang là mạng xã hội lớn nhất trên thế giới: cứ mỗi 60 giây sẽ có 510.000 comment được đăng lên, 293.000 trạng thái mới được cập nhật và khoảng 136.000 bức ảnh được upload. Ngoài facebook còn có rất nhiều các mạng xã hội khác như Twitter, Weibo, Tumblr, cũng như nhiều hình thức khác cho phép người dùng đưa thông tin lên internet. Nhận thấy rằng nếu có thể khai thác thông tin từ lượng dữ liệu khổng lồ này thì sẽ cho phép khai phá rất nhiều thông tin quan trọng giúp xác định và giải quyết nhiều vấn đề. Đơn cử như có thể dự đoán, định hướng xu thế của công nghệ, thời trang, tiêu dùng của xã hội.

Thứ hai, sự đa dạng và kết quả có thể thấy rõ khi áp dụng nó vào một số lĩnh vực như phân tích tâm lý người dùng, nghiên cứu thị trường. Ví dụ như trong kinh doanh, việc phân tích và nắm được các ý kiến phản hồi của người sử dụng, khách hàng sẽ giúp tổ chức, cá nhân nhận ra những điểm hạn chế của sản phẩm, dịch vụ mình cung cấp. Họ sẽ kịp thời có giải pháp khắc phục để đáp ứng được nhu cầu sử dụng của thị trường, nâng cao kết quả kinh doanh nhờ nắm bắt được thị hiếu và kênh chăm sóc khách hàng hiệu quả.

Quan điểm được chia làm chủ yếu là hai loại là tích cực (positive) và tiêu cực (negative). Ngoài ra trong một số trường hợp xét tới cả loại thứ ba là trung lập (neural).

1.3 Các hướng tiếp cận và giải quyết bài toán

Trong những năm gần đây, có rất nhiều bài báo và các công trình nghiên cứu cải tiến các thuật toán trích xuất thông tin quan điểm [6] [7] [15]. Các kỹ thuật này được phân loại theo hướng dựa trên các hướng tiếp cận dựa trên học máy hoặc dựa trên từ điển và ngữ nghĩa. Trong đó, hướng tiếp cận dựa trên học máy đang phát triển rất mạnh. Xét trên kĩ thuật học máy có giám sát có thể kể đến những thuật toán kinh điển và hiệu quả như Decision Tree, Support Vector Machine (SVM). Các thuật toán được đánh giá cao về tính đơn giản và hiệu quả trong nhiều trường hợp so với các thuật dựa trên mô hình mạng neural.

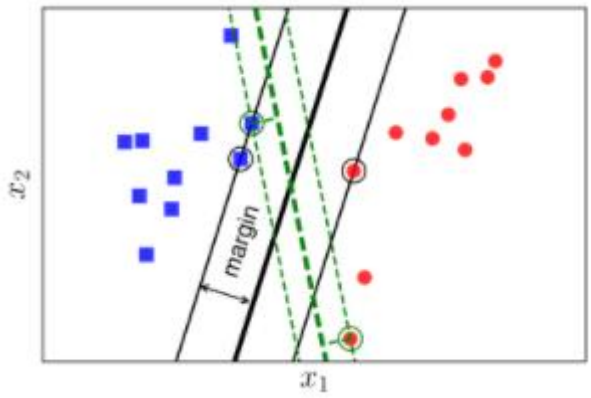


Hình 1.1 Các hướng tiếp cận giải quyết bài toán trích xuất thông tin quan điểm

Tuy nhiên, cùng với sự phát triển của khả năng tính toán các thuật toán mang hướng học sâu ngày càng phát triển hơn. Luận văn sẽ trình bày về một phương pháp dựa trên mạng neural có nhiều ưu điểm trong việc mô tả dữ liệu đầu vào, đó là mạng neural hồi quy RNN. Trước hết trong chương này sẽ đề cập tới một số thuật toán kinh điển hay sử dụng trong phân loại có thể áp dụng được đối với bài toán phân tích quan điểm.

1.3.1 Mô hình Support Vector Machine

Mô hình SVM là mô hình hết sức kinh điển trong bài toán phân loại. Tư tưởng của SVM [2] là định nghĩa ra một siêu mặt phẳng có thể phân tách các tập dữ liệu cần phân loại sao cho khoảng cách (margin) từ siêu mặt phẳng đến các tập cần phân loại là tương đương nhau và lớn nhất. Thuật toán SVM ban đầu được thiết kế để giải quyết bài toán phân lớp nhị phân với ý tưởng chính như sau:



Hình 1.2 Khoảng cách margin của 2 phân lớp là bằng nhau và lớn nhất

Trong không gian hai chiều tôi đã biết khoảng cách từ một điểm có tọa độ (x_0,y_0) tới đường thẳng có phương trình $w_1x+w_2y+b = 0$ được tính bằng:

$$h = \frac{| w_1 x_0 + w_2 y_0 + b |}{\sqrt{{w_1}^2 + {w_2}^2}}$$

Trong không gian ba chiều khoảng cách từ một điểm có tọa độ (x_0,y_0,z_0) tới một mặt phẳng có phương trình $w_1x + w_2y + w_3z + b = 0$ được tính bằng:

$$h = \frac{| w_1 x_0 + w_2 y_0 + w_3 z_0 + b|}{\sqrt{{w_1}^2 + {w_2}^2 + {w_3}^2}}$$

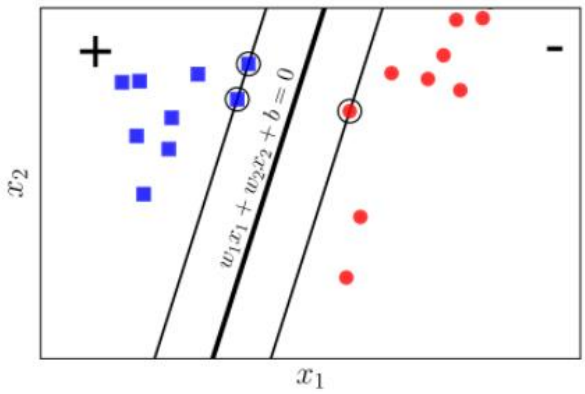
Nhận thấy nếu bỏ dấu giá trị tuyệt đối của tử số thì có thể xác định được điểm đang xét nằm về phía nào của đường thẳng hay mặt phẳng. Không làm mất tính tổng quát thì những biểu thức trong dấu giá trị tuyệt đối nếu mang dấu dương thì nằm cùng một phía dương còn những điểm làm cho biểu thức trong dấu giá trị tuyệt đối mang dấu âm thì nằm về phía âm. Những điểm nằm trên đường thẳng/ mặt phẳng sẽ làm cho giá trị của tử số bằng 0 hay khoảng cách bằng 0. Tổng quát trên không gian nhiều chiều thì sẽ phức tạp hơn so với việc biểu diễn bởi không gian 2 chiều (đường thẳng) hay không gian 3 chiều (mặt phẳng). Khái niệm này được gọi là siêu mặt phẳng có công thức $\mathbf{w^T x + b = 0}$. Khoảng cách được tính bằng:

$$h = \frac{|\, w^T x_0 + \mathrm{b}|}{\sqrt{\sum_{i=1}^{d} {w_i}^2}}$$

d là số chiều của không gian.

Chất lượng của siêu phẳng được đánh giá bởi khoảng cách h giữa hai lớp, khoảng cách càng lớn thì siêu phẳng quyết định càng tốt và chất lượng phân lớp càng cao.

Giả sử rằng các cặp dữ liệu của training set là (x_1, y_1), (x_2, y_2),…, (x_n, y_n) trong đó x_i là vector đầu vào của một điểm dữ liệu và y_i là nhãn của điểm dữ liệu đó. Giả sử nhãn của điểm dữ liệu có 2 giá trị là 1 và -1.



Hình 1.3 SVM nhị phân

Khi đó khoảng cách từ điểm đến mặt phân chia $w_1 x_1 + w_2\, x_2 + \mathrm{b} = 0$ là

$$h = \frac{y_n(\, w^T x_n + \mathrm{b})}{\sqrt{\sum_{i=1}^{d} {w_i}^2}}$$

Margin được tính là khoảng cách gần nhất của 1 điểm tới mặt phân chia

$$margin = \min_n \frac{y_n(\, w^T x_n + \mathrm{b})}{\sqrt{\sum_{i=1}^{d} {w_i}^2}}$$

Bài toán tối ưu trong SVM là bài toán tìm w và b sao cho margin này đạt giá trị lớn nhất:

$$(w, b) = \underset{w,b}{\operatorname{argmax}} \left\{ \frac{1}{\sqrt{\sum_{i=1}^{d} {w_i}^2}} \min_n y_n(\, w^T x_n + \mathrm{b}) \right\}$$

Đối với bài toán phân lớp với số phân lớp d > 2 thì tôi sử dụng chiến lược one-vs-rest bằng cách chuyển về bài toán phân lớp nhị phân giữa 1 lớp và (d-1) lớp còn lại. Tức là tôi sẽ phải thực hiện bài toán SVM nhị phân d lần giữa phân lớp thứ i và (d-1) phân lớp còn lại.

1.3.2 K-nearest neighbors

Thuật toán K-Nearest neighbors (KNN) là thuật toán phân loại dựa trên ý tưởng "Hãy cho tôi biết bạn của bạn là ai, tôi sẽ cho biết bạn là người như thế nào". Câu danh ngôn rất trùng hợp với cách thực hoạt động của thuật toán KNN. Bản chất KNN không học gì từ dữ liệu training, mọi tính toán được thực hiện khi nó cần dự đoán kết quả của dữ liệu mới. Mọi điểm trong dữ liệu training được lưu trữ trong bộ nhớ nên đây cũng là điểm hạn chế của thuật toán khi làm việc với bộ dữ liệu training lớn.

Các bước thực hiện của thuật toán như sau: thực hiện cấu hình tham số K – số điểm lân cận; đánh giá 1 điểm mới của tập test bằng cách xét K lân cận của nó; phân lớp cho điểm mới dựa trên nhãn của đa số mà K điểm trong tập train gần nhất của nó được gán.

Khái niệm thế nào là lân cận của 1 điểm thường được tính toán bằng khoảng cách vector theo norm. Ngoài ra đối với K điểm lân cận, tôi có thể đánh trọng số lớn hơn cho các điểm gần điểm cần xét hơn. Hay nói cách khác là tin cậy các điểm gần điểm cần xét hơn.

Sử dụng KNN để phân loại thường để sử dụng khi bài toán còn đơn giản, thuật toán chủ yếu thực hiện tính toán ở khâu test. Đây cũng là một trong số những thuật toán phân loại được sử dụng phổ biến nhất.

CHƯƠNG 2: MẠNG NEURAL VÀ RNN

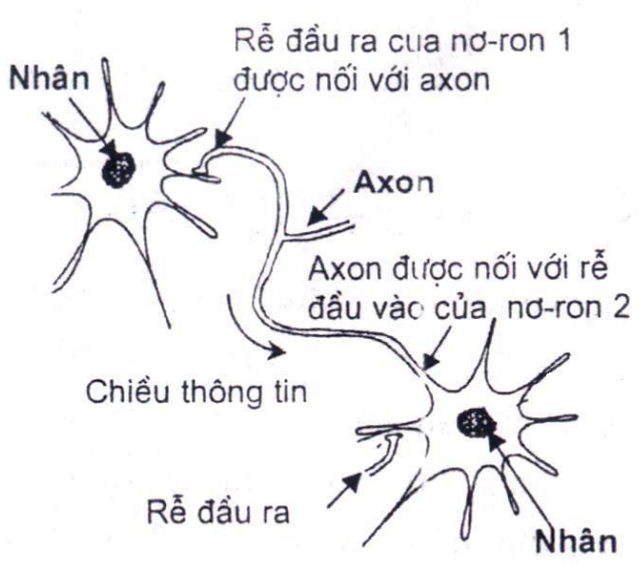
2.1 Mạng neural nhân tạo ANN

Mạng neural nhân tạo [1] là thuật toán mô phỏng lại cách thức hoạt động của hệ thống thần kinh của sinh vật trong việc học, nhận biết hay phân loại. Thuật toán đã được sử dụng rộng rãi từ những năm 1980 cho đến nay vẫn được áp dụng rộng rãi trong nhiều ngành khoa học. Mạng neural mô phỏng quá trình xử lý thông tin, mô hình được học bởi kinh nghiệm, lưu những kinh nghiệm hiểu biết và sử dụng trong các tình huống phù hợp.

2.1.1 Mạng nơ-ron sinh học

Hệ thống thần kinh là tổ chức vật chất cao cấp và có cấu tạo vô cùng phức tạp. Hệ thần kinh được cấu tạo bởi nhiều yếu tố trong đó nơ-ron là khái niệm cơ bản nhất. Trong bộ não người có khoảng 10^{11} - 10^{12} tế bào thần kinh được gọi là các nơ-ron và mỗi nơ-ron lại liên kết với khoảng 10^4 nơ ron khác thông qua các khớp nối thần kinh synapse.

Cấu tạo của mỗi nơ-ron gồm các thành phần cơ bản như thân nơ-ron và liên kết giữa các nơ-ron. Thân nơ-ron được giới hạn trong lớp màng và trong cùng là nhân. Nơi đo là nơi tiếp nhận tổng hợp và phát ra các xung thần kinh hay các tín hiệu điện sinh. Tại thân nơ-ron có rất nhiều đường rẽ nhánh gọi là rễ. Rễ được chia làm hai loại là rễ đầu vào nhân thông tin từ các nơ-ron khác qua axon và rễ đầu ra đưa thông tin qua axon tới các nơ-ron khác. Hình 2.1 mô tả thông tin được truyền từ nơ-ron 1 qua axon đến nơ-ron 2.



Hình 2.1 Mô hình mạng nơ ron sinh học

Quá trình hoạt động của nơ-ron là một quá trình điện hóa tự nhiên. Khi có tác động từ bên ngoài vào mạng nơ-ron sẽ phản ứng như sau: đầu vào của nơ-ron lớp đầu tiên sẽ xuất hiện một tín hiệu vượt quá mức cân bằng của nó và nó sẽ ở trạng thái kích thích. Trong bản thân nơ-ron sẽ xảy ra hàng loạt những phản ứng tạo thành thế năng. Thế năng được chuyển vào mạng thông qua axon để tới các nơ-ron tiếp theo. Cứ như vậy thế năng được truyền từ nơ-ron này đến nơ-ron khác trong đó nó sẽ có khả năng kích thích hoặc kìm hãm tự nhiên các neural khác trong mạng.

Một tính chất cơ bản của mạng neural sinh học là đáp ứng các kích thích, tác động từ bên ngoài và có khả năng thay đổi theo thời gian. Qua các lớp nơ-ron thì thế năng kích thích có thể được tăng lên, giảm đi hoặc thậm chí là biến mất. Chính sự liên kết chặt chẽ với nhau của các nơ-ron đã tạo ra mạng lưới đáp ứng, thay đổi không ngừng theo thời gian. Sự thay đổi trạng thái của một neural dẫn thời sự thay đổi trạng thái của các nơ-ron khác và dẫn đến sự thay đổi của toàn bộ mạng.

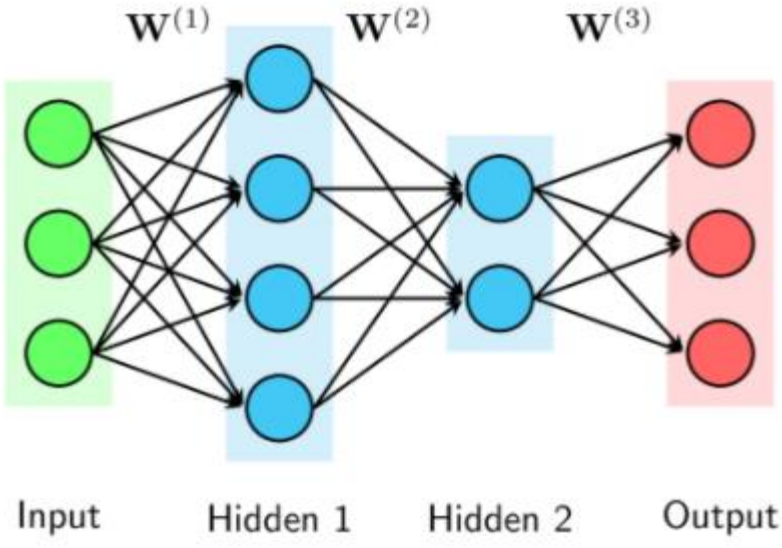
Các nhà khoa học đã tìm hiểu và lấy nguyên lý cấu trúc của mạng nơ-ron sinh học để xây dựng thành mô hình mạng neural nhân tạo.

2.1.2 Kiến trúc tổng quát của mạng neural nhân tạo

Mạng neural nhân tạo (Artificial Neural Network) gọi tắt là ANN là một mô hình xử lý thông tin phỏng theo các thức xử lý thông tin của hệ thống nơ-ron sinh học[1]. Nó

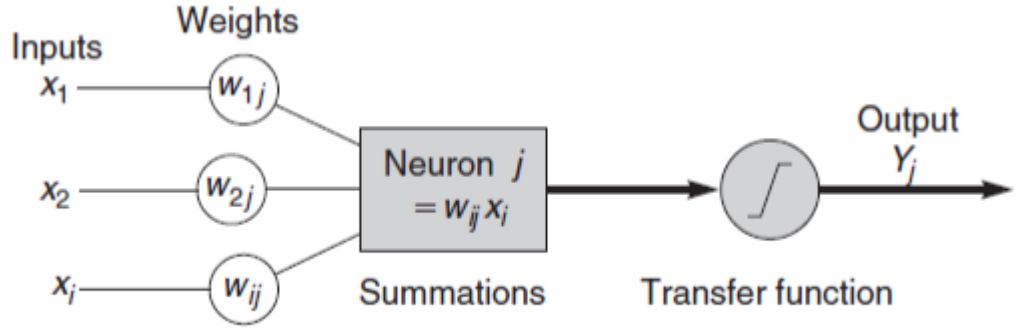
được tạo lên từ một số lượng lớn các phần tử gọi là neural kết nối với nhau thông qua các liên kết gọi là trọng số liên kết. Mạng neural nhân tạo thường được mô phỏng và huấn luyện từ tập mẫu. Qua quá trình huấn luyện, các trọng số liên kết sẽ được cập nhật sao cho giá trị làm lỗi là nhỏ nhất.

Các thành phần của mạng ANN bao gồm các lớp đầu vào (input layer), lớp ẩn (hidden layer) và các lớp đầu ra (output layer). Hình 2.2 là ví dụ về một ANN có 2 lớp ẩn.



Hình 2.2 Mạng neural 2 lớp ẩn

Kiến trúc chung của một ANN gồm 3 thành phần đó là input layer, hidden layer và output layer. Trong đó, lớp ẩn (hidden layer) gồm các nơ-ron, nhận dữ liệu input từ các nơ-ron ở lớp trước đó và chuyển đổi các input này cho các lớp xử lý tiếp theo. Quá trình xử lý thông tin của một ANN như sau:



Hình 2.3 Quá trình xử lý thông tin của neural j trong mạng ANN

Trong đó, mỗi input tương ứng với 1 thuộc tính của dữ liệu. Ví dụ như trong ứng dụng của ngân hàng xem xét có chấp nhận cho khách hàng vay tiền hay không thì mỗi input là một thuộc tính của khách hàng như thu nhập, nghề nghiệp, tuổi, số con…Output là một giải pháp cho một vấn đề, ví dụ như với bài toán xem xét chấp nhận cho khách hàng vay tiền hay không thì output là yes - cho vay hoặc no - không cho vay. Trọng số liên kết (Connection Weights) là thành phần rất quan trọng của một ANN, nó thể hiện mức độ quan trọng hay có thể hiểu là độ mạnh của dữ liệu đầu vào đối với quá trình xử lý thông tin, chuyển đổi dữ liệu từ layer này sang layer khác. Quá trình học (Learning Processing) của ANN thực ra là quá trình điều chỉnh các trọng số (Weight) của các input data để có được kết quả mong muốn. Hàm tổng (Summation Function) cho phép tính tổng trọng số của tất cả các input được đưa vào mỗi nơ-ron. Hàm tổng của một nơ-ron đối với n input được tính theo công thức sau:

$$Y = \sum_{i=1}^{n} X_i W_i$$

Kết quả trên cho biết khả năng kích hoạt của nơ-ron đó. Các nơ-ron này có thể sinh ra một output hoặc không trong ANN, hay nói cách khác rằng có thể output của 1 nơ-ron có thể được chuyển đến layer tiếp trong mạng nơ-ron hoặc không là do ảnh hưởng bởi hàm chuyển đổi (Transfer Function). Việc lựa chọn Transfer Function có tác động lớn đến kết quả của ANN. Vì kết quả xử lý tại các nơ-ron là hàm tính tổng nên đôi khi rất lớn, nên transfer function được sử dụng để xử lý output này trước khi chuyển đến layer tiếp theo. Hàm chuyển đổi phi tuyến được sử dụng phổ biến trong ANN là sigmoid (logical activation) function.

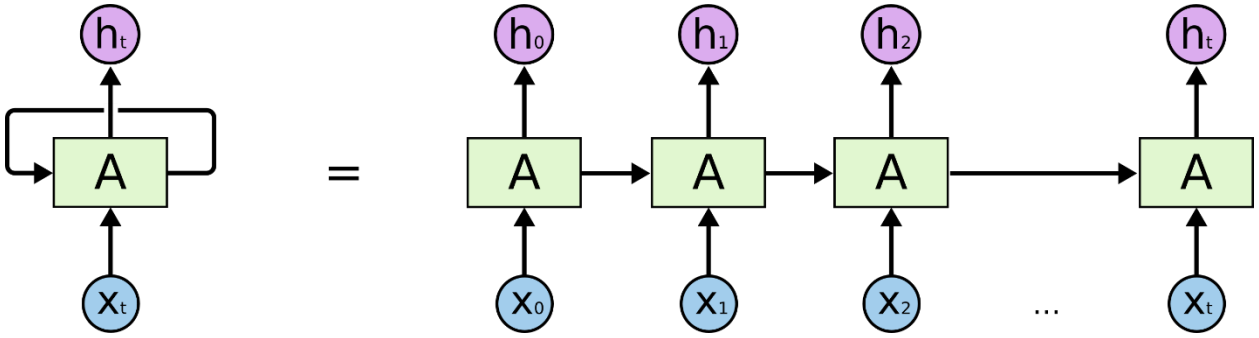
$$Y_T = 1/(1 + e^{-Y})$$

Kết quả của Sigmoid Function thuộc khoảng [0, 1] nên còn gọi là hàm chuẩn hóa (Normalized Function). Đôi khi thay vì sử dụng hàm chuyển đổi, tôi sử dụng giá trị ngưỡng (Threshold value) để kiểm soát các output của các nơ-ron tại một layer nào đó trước khi chuyển các output này đến các layer tiếp theo. Nếu output của một nơ-ron nào đó nhỏ hơn Threshold thì nó sẽ không được chuyển đến Layer tiếp theo. Ứng dụng thực tế của mạng nơ-ron thường được sử dụng trong các bài toán nhận dạng mẫu như nhận dạng chữ cái quang học (Optical character recognition), nhận dạng chữ viết tay, nhận dạng tiếng nói, nhận dang khuôn mặt.

2.2 Mạng neural hồi quy RNN

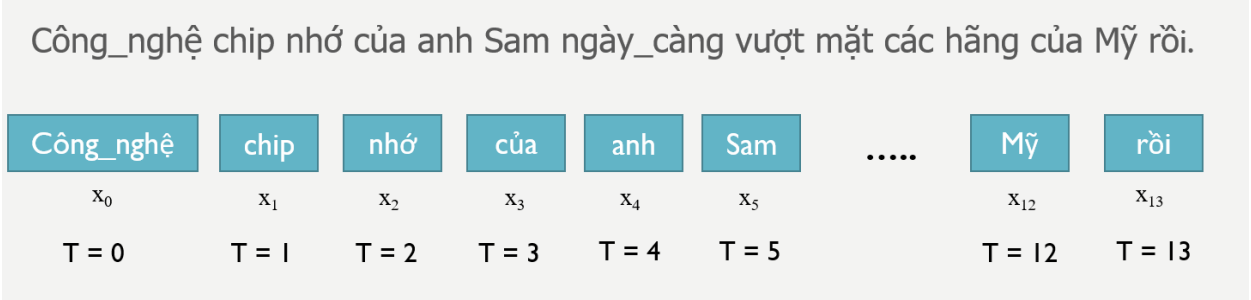
Như chúng tôi đã biết, ý kiến phản hồi được lưu dưới dạng văn bản có thể bao gồm nhiều từ, nhiều câu. Do đó, để hiểu được và trích xuất được thông tin quan điểm từ một ý kiến phản hồi, tôi phải căn cứ trên ngữ cảnh của toàn bộ những câu đã được đọc trước đó. Điều này cũng dễ hiểu bởi thực tế con người không bắt đầu suy nghĩ tại mọi thời điểm mà phải căn cứ vào những gì xảy ra trước đó. Giả sử, khi đọc một bài review về sản phẩm, tôi sẽ đọc từng từ tại mỗi thời điểm và hiểu từng từ đó dựa vào sự hiểu ngữ cảnh của các từ trước đó. tôi không vứt bỏ tri thức ngữ cảnh trước đó hay lại bắt đầu sự suy luận tại mọi thời điểm mà sự hiểu văn bản phải được duy trì nhất quán.

Các mạng ANN không thể làm được điều này vì bản chất nó không mô phỏng khía cạnh thời gian. Giả sử bạn muốn phân loại sự kiện nào sẽ xảy ra ở một thời điểm trong bộ phim. Mạng ANN khó có thể được vận dụng để dự đoán được sự kiện xảy ra ở thời điểm cần xét mà không căn cứ vào những sự kiện trước trong phim. Mạng ANN cho các neural thành phần của lớp đầu vào, lớp ẩn và lớp đầu ra là độc lập về mặt thời gian. Trong khi đó, tính chất thời gian trước sau lại là đặc trưng của ngôn ngữ văn bản hay xử lý ngôn ngữ tự nhiên.



Hình 2.4 Mô hình mạng RNN

Mạng neural hồi quy RNN [9] được mô hình để giải quyết vấn đề mô phỏng về mặt thời gian của dữ liệu chuỗi. Do đó, mạng RNN rất phù hợp cho việc mô hình hóa xử lý ngôn ngữ. Trong đó, mỗi từ trong chuỗi đầu vào sẽ được liên kết với một bước thời gian cụ thể. Trong thực tế, số bước thời gian sẽ bằng với độ dài tối đa của chuỗi. Hình 2.4 là mô tả cơ bản của mạng RNN. Hàm A nhận đầu vào x_t tại thời điểm t và đầu ra là giá trị vector ẩn h_t. Nhận thấy, hàm A cho phép thông tin được lặp lại truyền từ một bước của mạng tới bước tiếp theo.



Hình 2.5 Ví dụ về cách xử lý thông tin dạng chuỗi của RNN

Hình 2.5 cho thấy cách mô hình RNN xử lý một thông tin dạng chuỗi theo thời gian. Tại từng thời điểm t, các từ sẽ lần lượt được đưa vào mô hình. Tương ứng với mỗi mốc thời gian là một thành phần vector ẩn ht. Hiểu một cách mô hình hóa, vector h_t sẽ gói gọn và tóm tắt tất cả thông tin đã được đọc trong các bước thời gian trước đó. Trong khi đó, x_t là vector đóng gọi thông tin của một từ cụ thể được đưa vào mô hình RNN tại thời điểm t. Ở đây, x_0 là vector mô tả từ "Công_nghệ" được đưa vào mô hình tại thời điểm t=0, x_1 là vector mô tả từ "chip" được đưa vào mô hình tại thời điểm t=1.

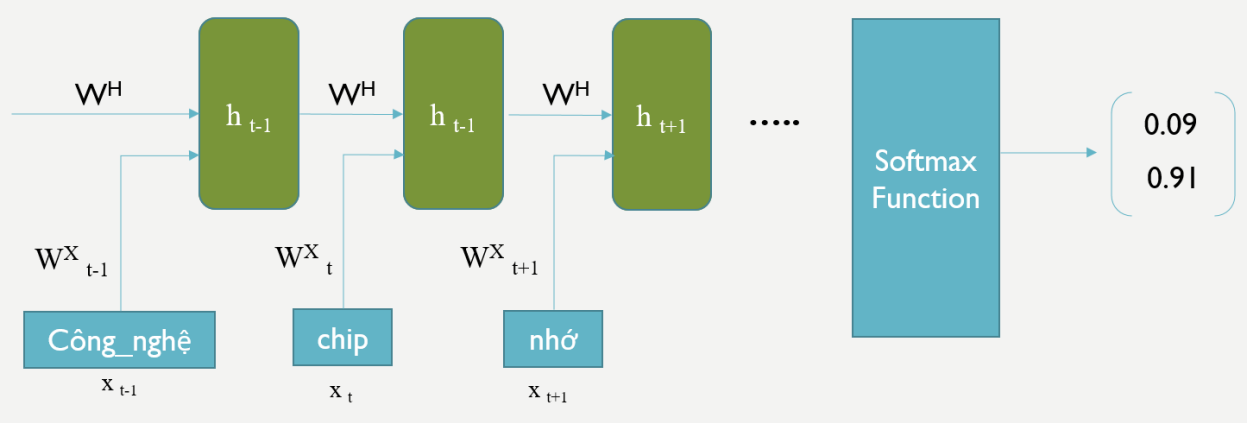
Vector trạng thái ẩn h_t là một hàm của cả từ vựng hiện tại và vector trạng thái ẩn ở bước trước. Sigma là một hàm kích hoạt thường là một hàm sigmoid hoặc tanh.

$$h_t = \sigma(W^H h_{t-1} + W^X x_t)$$

Hình 2.6 Công thức tính vector trạng thái ẩn tại thời điểm t

W^H và W^X trong công thức hình 2.6 là hai ma trận trọng số. Ma trận W^X được sử dụng để nhân với vector đầu vào x_t và ma trận trọng số W^H nhân với vector trạng thái ẩn vào thời điểm trước đó. W^H là một ma trận không thay đổi trong tất cả các bước thời gian trong khi đó W^X là ma trận có giá trị thay đổi khác nhau cho mỗi đầu vào.

Nhận thấy, giá trị của vector ẩn tại thời điểm t bị ảnh hưởng bởi giá trị của vector x_t tại thời điểm hiện tại và giá trị của vector ẩn h_{t-1} của trạng thái t-1 trước đó. Vậy giá trị h_t sẽ thay đổi như thế nào nếu hai ma trận W^H và W^X có giá trị lớn hoặc nhỏ. Giả sử W^H có giá trị lớn và W^X có giá trị nhỏ suy ra giá trị của h_t sẽ bị ảnh hưởng nhiều hơn bởi h_{t-1} mà không mấy bị ảnh hưởng bởi x_t . Nói một cách khác, vector trạng thái ẩn h_t thấy rằng từ x_t được đưa vào thời điểm t không có giá trị hay không quan trọng đối với toàn bộ ngữ cảnh tổng thể của câu cho tới thời điểm t. Do đó, h_t sẽ có giá trị xấp xỉ so với h_{t-1}.



Hình 2.7 Mô hình trích xuất quan điểm cơ bản sử dụng RNN và softmax

Ma trận trọng số W được cập nhật thông qua quá trình tối ưu hóa hàm lỗi tại bước lan truyền ngược. Vector trạng thái ẩn tại bước cuối cùng được đưa vào hàm phân loại. Bước này thường được đặt tên là full connection, Trong đó, vector trạng thái ẩn ở bước cuối thường được nhân với mà một ma trận trọng số và đưa vào hàm softmax để đưa ra tương ứng các giá trị của lớp phân loại. Thông thường đối với bài toán trích xuất thông tin quan điểm thì tôi sẽ xác định giá trị đầu ra của hàm softmax cho hai phân lớp tích cực và tiêu cực.

$$a_i = \frac{\exp(z_i)}{\sum_{j=1}^{C} \exp(z_j)}, \quad \forall i = 1, 2, \ldots, C$$

Hình 2.8 Hàm softmax

Hàm softmax thường được sử dụng tính xác suất thuộc phân lớp i trong bài toán phân loại. C là số lớp được phân loại. Hàm softmax có ưu điểm là các xác suất ai đều dương và có tổng bằng 1.

2.3 Vấn đề lưu trữ thông tin ngữ cảnh phụ thuộc lâu dài.

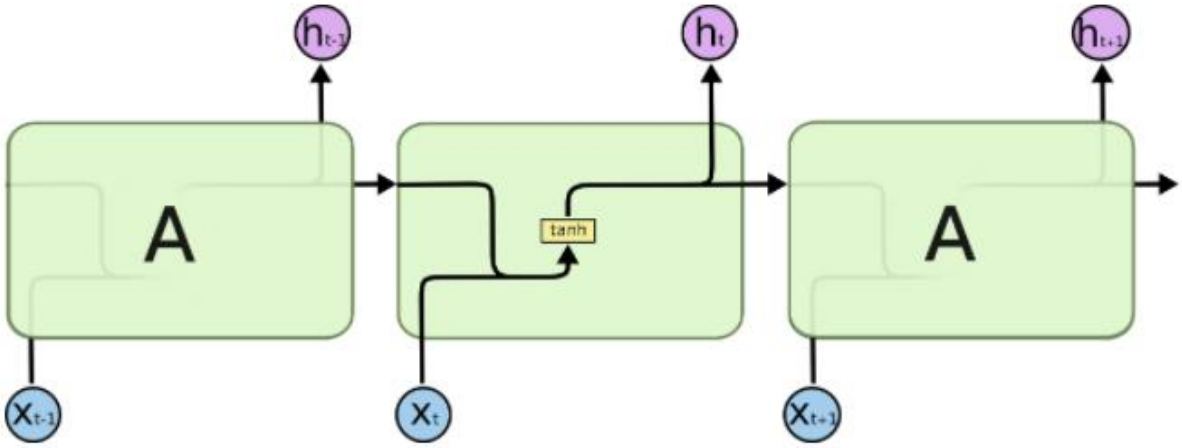
Xét một câu hỏi sau: " Số thứ nhất bằng 3. Đám mây đang bay trên bầu trời. Số thứ hai bằng 4. Tổng của hai số bằng mấy?". Ở mức độ lưu trữ thông tin cao, mạng RNN có thể lưu trữ toàn bộ các thông tin của 4 câu kể trên. Sau đó, RNN xác định ngữ cảnh câu hỏi cũng như giá trị của số thứ nhất và số thứ hai. tôi thấy rằng câu "Đám mây đang bay trên bầu trời" không có giá trị trong ngữ cảnh này. Hay nói cách khác là làm nhiễu kết quả của câu trả lời. Để trả lời câu hỏi trên, bắt buộc mạng RNN phải lưu trữ toàn bộ

các từ vào trong bộ nhớ. Trong phạm vi 4 câu cho tới 10 câu có thể khả thi, nhưng nếu đoạn văn dài và thông tin quan trọng được xuất hiện rời rạc, ngăn cách bởi nhiều câu nhiễu thì cách lưu trữ của RNN trở nên nặng nề và không hợp lý. Đây chính là vấn đề lưu trữ thông tin phụ thuộc lâu dài.

Trên lý thuyết, mạng RNN có thể phát sinh bộ nhớ đủ để xử lý vấn đề lưu trữ phụ thuộc dài. Tuy nhiên, trong thực tế thì không phải vậy. Vấn đề này đã được Hochreiter (1991) đưa ra như thách thức của mạng RNN. Và mạng Long short-term memory (LSTM) được phát biểu năm 1997 đã giải quyết được vấn đề này.

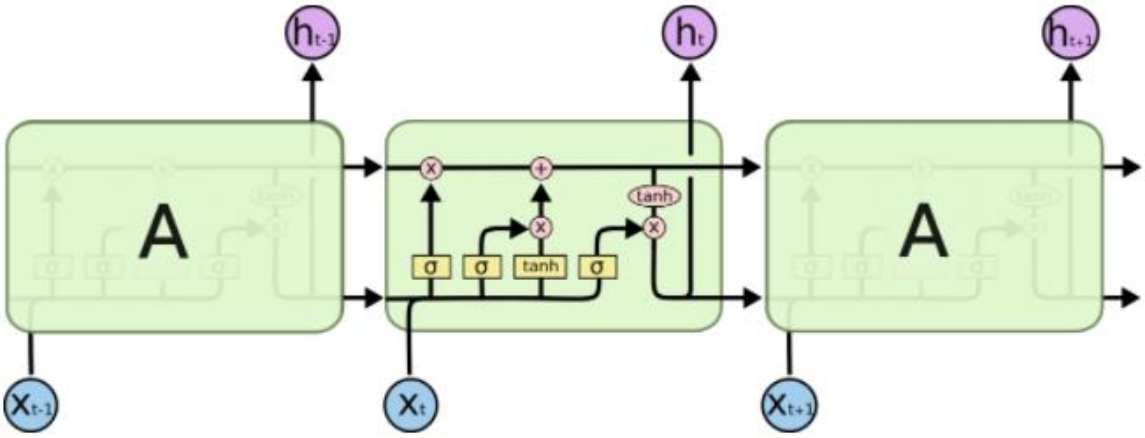
2.4. Mạng Long short-term memory

Long short term memory là cải tiến của mạng RNN nhằm giải quyết vấn đề học, lưu trữ thông tin ngữ cảnh phụ thuộc dài. tôi cùng xem xét cách LSTM [9] cải tiến hơn so với mạng RNN. Trong mô hình RNN, tại thời điểm t thì giá trị của vector ẩn ht chỉ được tính bằng một hàm tanh



Hình 2.9 Module xử lý tính h_t của RNN

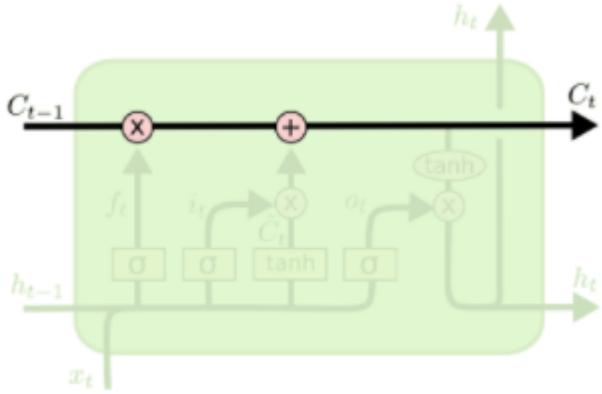
LSTM cũng có cấu trúc mắt xích tương tự, nhưng các module lặp có cấu trúc khác hẳn. Thay vì chỉ có một layer neural network, thì LSTM có tới bốn layer, tương tác với nhau theo một cấu trúc cụ thể. Christopher Olah [10] đã có cách giải thích rất cụ thể về cách hoạt động của RNN.



Hình 2.10 Module lặp của mạng LSTM

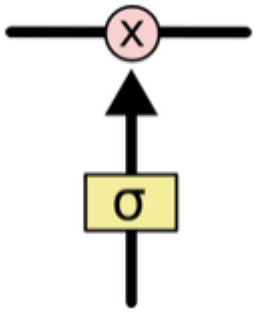
Các ký hiệu sử dụng trong mạng LSTM gồm có: hình chữ nhật là các lớp ẩn của mạng nơ-ron, hình tròn biểu diễn toán tử Pointwise, đường kẻ gộp lại với nhau biểu thị phép nối các toán hạng, và đường rẽ nhánh biểu thị cho sự sao chép từ vị trí này sang vị trí khác. Mô hình thiết kế của LSTM là một bảng mạch 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 thiết kế bảng mạch.

Mấu chốt của LSTM là cell state (trạng thái nhớ), đường kẻ ngang chạy dọc ở trên cùng của hình 2.11. Cell state giống như băng chuyền, chạy xuyên thẳng toàn bộ mắc 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 2.11 Cell state của LSTM giống như một băng chuyền

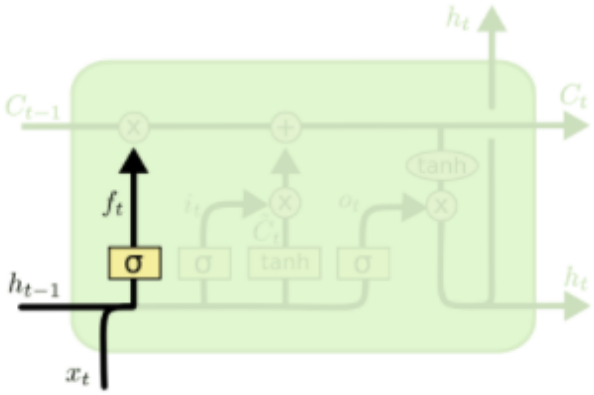
LSTM có khả năng thêm hoặc bớt thông tin vào cell state, được quy định một cách cẩn thận bởi các cấu trúc gọi là cổng (gate). Các cổng này là một cách (tuỳ chọn) để định nghĩa thông tin băng qua. Chúng được tạo bởi hàm sigmoid và một toán tử nhân pointwise.



Hình 2.12 Cổng trạng thái LSTM

Hàm kích hoạt Sigmoid có giá trị từ 0 – 1, mô tả độ lớn thông tin được phép truyền qua tại mỗi lớp mạng. Nếu tôi thu được zero điều này có nghĩa là "không cho bất kỳ cái gì đi qua", ngược lại nếu thu được giá trị là một thì có nghĩa là "cho phép mọi thứ đi qua". Một LSTM có ba cổng như vậy để bảo vệ và điều khiển cell state.

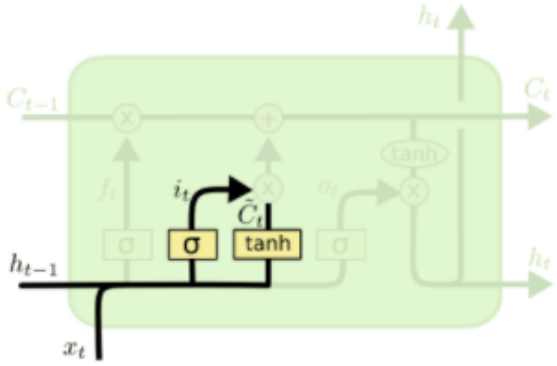
Quá trình hoạt động của LSTM được thông qua các bước cơ bản sau. Bước đầu tiên của mô hình LSTM là quyết định xem thông tin nào chúng tôi cần loại bỏ khỏi cell state. Tiến trình này được thực hiện thông qua một sigmoid layer gọi là "forget gate layer" – cổng chặn. Đầu vào là h_{t-1} và x_t , đầu ra là một giá trị nằm trong khoảng [0, 1] cho cell state C_{t-1}. 1 tương đương với "giữ lại thông tin", 0 tương đương với "loại bỏ thông tin".



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Hình 2.13 Cổng chặn f_t

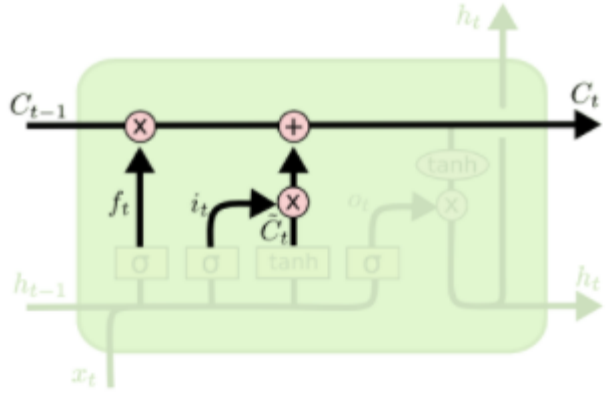
Bước tiếp theo, cần quyết định thông tin nào cần được lưu lại tại cell state. tôi có hai phần là single sigmoid layer được gọi là “input gate layer”- cổng vào quyết định các giá trị chúng tôi sẽ cập nhật. Tiếp theo, một tanh layer tạo ra một vector ứng viên mới $\tilde{Ct}$ được thêm vào trong cell state.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Hình 2.14 Cổng vào i_t và tanh $\tilde{Ct}$

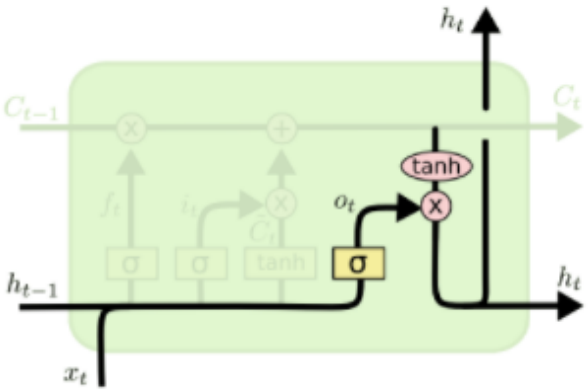
Ở bước tiếp theo, sẽ kết hợp hai thành phần này lại để cập nhật vào cell state. Lúc cập nhật vào cell state cũ, C_{t-1}, vào cell state mới C_t . tôi sẽ đưa state cũ hàm f_t , để quên đi những gì trước đó. Sau đó, tôi sẽ thêm $i_t * \tilde{Ct}$. Đây là giá trị ứng viên mới, co giãn (scale) số lượng giá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



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Hình 2.15 Giá trị state C_t

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 single sigmoid layer để quyết định xem phần nào của cell state chúng tôi dự định sẽ output. Sau đó, tôi sẽ đẩy cell state qua tanh (đẩy giá trị vào khoảng -1 và 1) và nhân với một “output sigmoid gate” cổng ra, để giữ lại những phần tôi muốn output ra ngoài.



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Hình 2.16 Giá trị cổng ra và vector trạng thái ẩn h_t

Mạng LSTM theo các công thức kể trên mà được lặp lại qua từng thời điểm t. Thông tin của cell state được điều khiển bởi cấu trúc các cổng chặn f_t, cổng vào i_t và cổng ra o_t . Trong đó cổng chặn f_t chính là tư tưởng chủ đạo của mạng LSTM khi cho phép điều khiển lượng thông tin đầu vào h_{t-1} từ các thời điểm trước.

CHƯƠNG 3: RNN CHO BÀI TOÁN TRÍCH XUẤT QUAN ĐIỂM

3.1 Bài toán trích xuất thông tin quan điểm sử dụng RNN

3.2 Một số phương pháp vector hóa từ

Để hiểu cách mạng neural hay các mô hình mạng học sâu được áp dụng, đầu tiên tôi phải hiểu cách mà dữ liệu được đưa vào mô hình. Mạng neural tích chập Convolutional Neural Network (CNN) sử dụng đầu vào là mảng các giá trị của pixel, các mô hình hồi quy logistic thì dùng các định lượng đặc trưng. Nhận thấy, các đầu vào của các mô hình cần phải là các giá trị vô hướng hoặc ma trận các giá trị vô hướng. Tuy nhiên, khi suy nghĩ một quá trình xử lý ngôn ngữ tự nhiên NLP thì đầu vào thường là một từ, một câu hay một văn bản. Do đó, thay vì có một đầu vào là từ, câu hay chuỗi thì tôi cần chuyển đổi từng từ trong câu thành một vector. Bước xử lý này trong NLP gọi là vector hóa dữ liệu. Trong phần này, luận văn sẽ nêu một số cách vector hóa được sử dụng từ đơn giản đến nâng cao.

3.2.1 Bag of Words

Mô hình Bag of Words là mô hình thường dùng trong các tác vụ phân lớp văn bản. Thông tin sẽ được biểu diễn thành tập các từ kèm với tần suất xuất hiện của mỗi từ này trong văn bản. Cơ bản là thực hiện bằng cách đếm số lần xuất hiện của mỗi từ trong văn bản.

Ví dụ, với hai câu sau:

(1) Nam thích xem phim. Lan cũng rất thích xem phim
(2) Nam còn thích chơi đá bóng

Dựa trên hai câu trên thì tập từ điển được xây dựng là

[“Nam”, “thích”, “xem”, “phim”, “Lan”, “cũng”, “rất”, “còn”, “chơi”, “đá_bóng”]

Dự trên tập từ điển xây dựng được, tôi vector hóa 2 câu ban đầu được kết quả như sau:

(1) [1,2,2,2,1,1,1,0,0,0]
(2) [1,1,0,0,0,0,0,1,1,1]

Nhận xét rằng với cách mô hình hóa bằng Bag of words thì sẽ không quan tâm đến thứ tự xuất hiện của từ mà chỉ quan tâm đến tần suất xuất hiện. Do đó, hai câu như "anh yêu em" và "em yêu anh" được vector hóa là như nhau.

3.2.2 TF-IDF

TF-TDF là thuật ngữ viết tắt của Term Frequency – Inverse Document Frequency. TF-IDF là trọng số của một từ trong văn bản thu được thông qua thống kê thể hiện mức độ quan trọng của từ này trong một văn bản. Qua nghiên cứu, mô hình Bag of Words có đặc điểm là các từ quan trọng trong văn bản thường xuất hiện ít. Mà nội dung văn bản lại cần trọng số đóng góp của các từ này càng nhiều trong các thành phần vector. Mô hình TF-IDF là một cách để làm nổi bật các từ chỉ xuất hiện ở một vài văn bản. Bên cạnh đó là các từ xuất hiện càng nhiều ở các văn bản thì tôi càng giảm giá trị của các từ này.

Các từ hiếm, quan trọng thường có đặc điểm sau:

- Xuất hiện nhiều trong một văn bản
- Xuất hiện ít trong cả tập ngữ liệu

$$\mathrm{TF(t,d)} = \frac{\text{Số lần từ t xuất hiện trong văn bản d}}{\text{Tổng số từ trong văn bản d}}$$

$$IDF(t,D) = \log \frac{Tổng\ số\ văn\ bản\ trong\ tập\ mẫu\ D}{\text{Số văn bản có chứa từ t}}$$

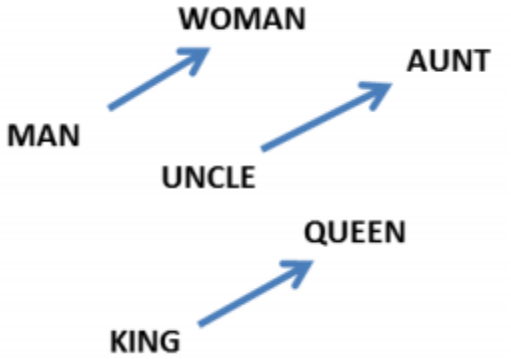
$$TF\_IDF(t,d,D) = \mathrm{TF(t,d)} * IDF(t,D)$$

Nhận thấy hàm IDF(t,D) đảm bảo tính chất nêu trên của từ quan trọng. Một từ mà xuất hiện ở nhiều văn bản thì mẫu của hàm log lớn dẫn đến log tiến về 0 tương đương với từ này kém giá trị. Và ngược lại, số từ sử dụng trong các văn bản càng ít thì log sẽ tiến về giá trị lớn hơn. Sử dụng phương pháp TF-IDF tôi sẽ mô tả được vector của tập ngữ liệu kích thước bằng số lượng văn bản x số lượng từ trong ngữ liệu. Mô hình TF-IDF đã cải tiến hơn mô hình Bag of Words ở góc độ nhấn mạnh được các từ quan trọng.

3.2.3 Word2vec

Giới thiệu

Trong khi TF-IDF vẫn đặc trưng cho kiểu mô hình dựa trên Bag of Words sử dụng phép đếm và xác suất thì Word2vec được ra đời với nhiều cải tiến đáng kể. Word2vec là phương pháp biểu diễn một từ dưới dạng một phân bố quan hệ với các từ còn lại. Mỗi từ được biểu diễn bằng một vector có các phần tử mang giá trị là phân bố quan hệ của từ này đối với các từ khác trong từ điển. Năm 2013, Google đã khởi dựng dự án word2vec của riêng mình với dữ liệu được sử dụng từ Google News [7] [10]. Bộ dữ liệu được coi là đồ sộ nhất cho tới bây giờ với 100 tỷ từ.



Hình 3.1 Phân bố quan hệ giữa từ trong word2vec

Ví dụ bài toán kinh điển King + Man – Woman = ?. Việc nhúng các từ trong không gian vector cho thấy sự tương tự giữa các từ. Giả sử như tại hình 3.1 là một sự khác biệt về mặt giới tính giữa các cặp từ (“man”,”woman”), (“uncle”,”aunt”), (“king”,”queen”)

$$W(\text{“woman”}) – W(\text{“man”}) \approx W(\text{“aunt”}) – W(\text{“uncle”})$$

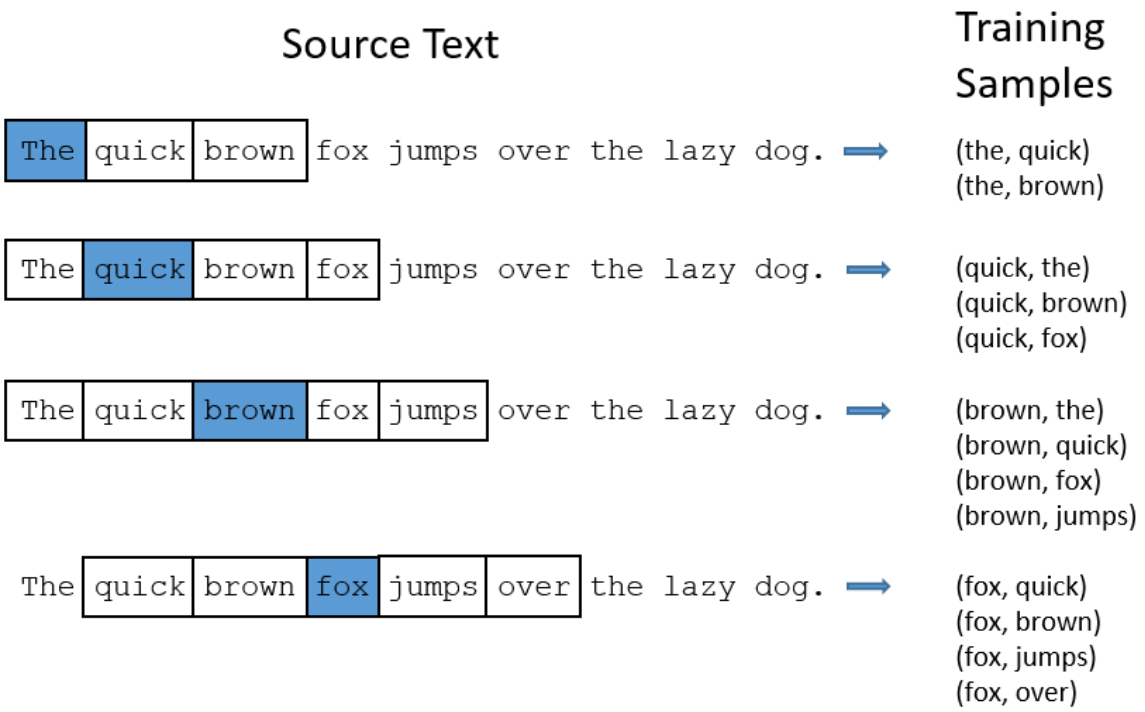
$$W(\text{“woman”}) – W(\text{“man”}) \approx W(\text{“queen”}) – W(\text{“king”})$$

Từ đó, kết quả của King + Man – Woman = Queen.

Để xây dựng được vector mô tả phân bố quan hệ với tập từ điển, bản chất mô hình Word2vec sử dụng một mạng neural đơn giản với một lớp ẩn. Sau khi được huấn luyện trên toàn bộ tập văn bản, toàn bộ lớp ẩn sẽ có giá trị mô hình hóa quan hệ của từ trong tập văn bản được huấn luyện ở mức trừu tượng. Trong ngữ cảnh, từ sẽ được huấn luyện việc sử dụng thuật toán Continuous Bag of Words (CBOW) và skip gram. Bản chất của CBOW là sử dụng ngữ cảnh để đoán từ và bản chất của skip gram là dùng từ để dự đoán

ngữ cảnh. Một trong hai cách sẽ được áp dụng để huấn luyện cho mô hình word2vec, trong đó cách sử dụng mô hình skip gram thường được sử dụng do việc đáp ứng tốt với tập dữ liệu lớn.

Khi sử dụng mô hình skip gram thì đầu vào là một từ trong câu, thuật toán sẽ nhìn vào những từ xung quanh nó. Giá trị số từ xung quanh nó được xét gọi là "window size". Một window size bằng 5 có nghĩa sẽ xét 5 từ trước nó và 5 từ sau nó. Xác suất đầu ra sẽ liên quan tới khả năng tìm thấy các từ xung quanh từ hiện tại đang xét. Ví dụ nếu tôi đã huấn luyện với từ đầu vào là "bóng đá", xác suất đầu ra sẽ cao hơn đối với những từ "quả bóng" hay "cầu thủ" so với các từ không liên quan như "dưa hấu" hay "Nam Phi". tôi sẽ huấn luyện mạng neural này bằng cách cho xét từng cặp từ gồm từ được xét và từ xung quanh nó. Xét câu "The quick brown fox jumps over the lazy dog" với window size bằng 2. Từ được bôi đậm là từ đầu vào.

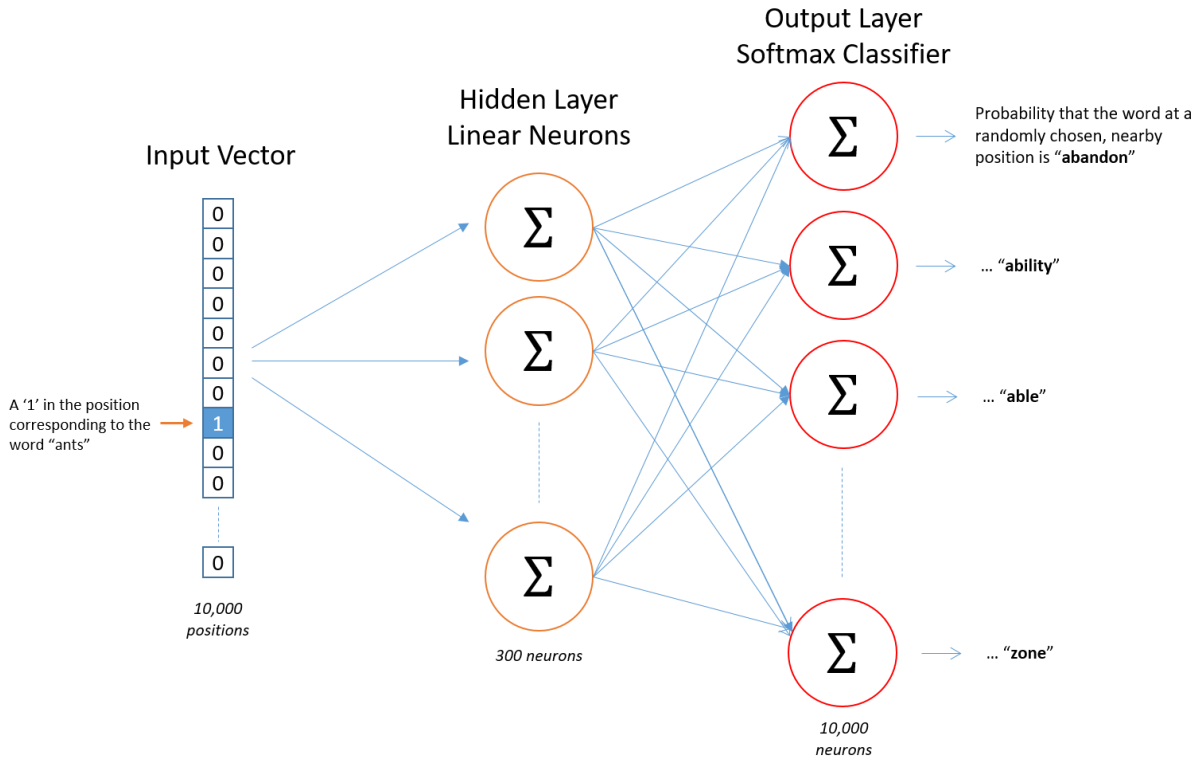


Hình 3.2 Mô hình skip-gram trong Word2vec

Mạng neural sẽ được huấn luyện từ số liệu thống kê số lần cặp ghép xuất hiện. Do vậy, mạng sẽ nhận được nhiều cặp mẫu huấn luyện ("bóng đá", "cầu thủ") hơn là ("bóng đá", "dưa hấu"). Khi quá trình huấn luyện kết thúc, nếu đưa ra từ "bóng đá" như đầu vào thì sẽ có một giá trị xác suất cao hơn cho "cầu thủ" và "quả bóng" so với "dưa hấu" và "Nam Phi".

Chi tiết cách thực hiện

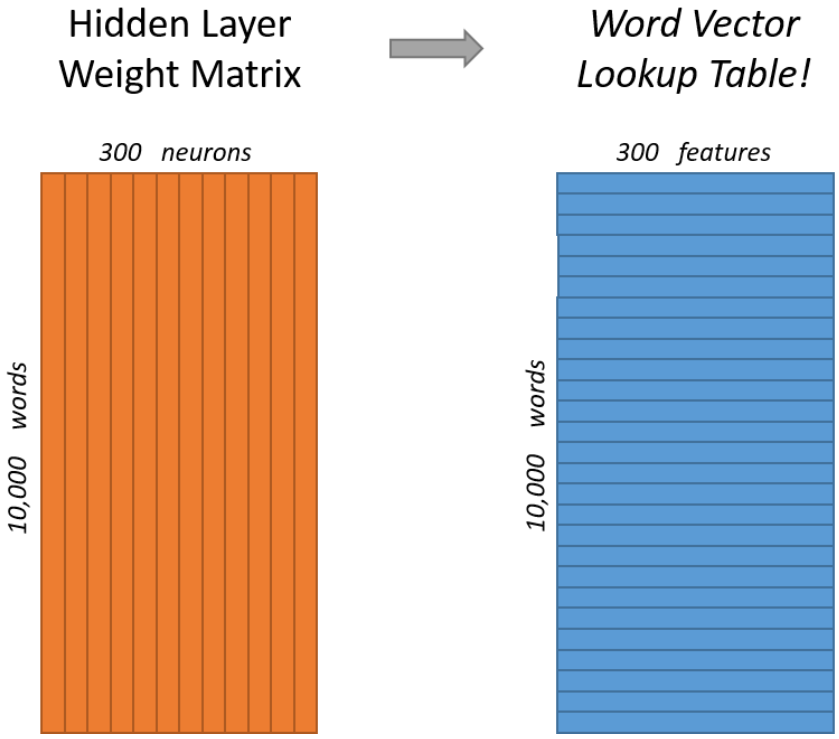
Trước hết, tôi phải đưa từ vào mạng neural một lớp ẩn kể trên. Để có thể huấn luyện được, từ được vector hóa để cho vào mạng. tôi có thể xây dựng kho từ điển từ tập dữ liệu văn bản sau đó sử dụng one-hot-vector để diễn tả từng từ trong kho từ điển. Giả sử, tôi có từ điển gồm 10.000 từ riêng biệt. Vector one-hot sẽ gồm 10.000 thành phần đại diện cho mỗi từ trong từ điển. Vector one-hot có dạng bao gồm toàn bộ giá trị bằng 0, chỉ có chỉ số tương ứng với vị trí của từ trong từ điển có giá trị bằng 1. Ví dụ từ "ants" sẽ biểu diễn bằng vector 10.000 phần tử. gồm toán số 0, duy nhất số 1 tại vị trí tương ứng với từ "ants" trong từ điển [11]



Hình 3.3 Mô hình mạng neural 1 lớp ẩn của Word2vec

Lớp ẩn giả sử gồm 300 neuron, thường không sử dụng hàm activation, nhưng đầu ra thì sử dụng hàm softmax. Đầu ra sẽ là vector cũng là một vector có độ lớn 10.000 và giá trị tương ứng với mỗi vị trí là xác suất xuất hiện gần từ đã chọn của từ gần vị trí đó. Kích thước 300 neuron ở lớp ẩn là một hyperparameter của mô hình, nó được gọi là số chiều hay số đặc trưng của word2vec. Con số 300 được Google sử dụng trong mô hình

huấn luyện từ tập ngữ liệu Google News [12]. Giá trị hyperparameter có thể được thay đổi sao cho phù hợp với mô hình, dữ liệu của người nghiên cứu.



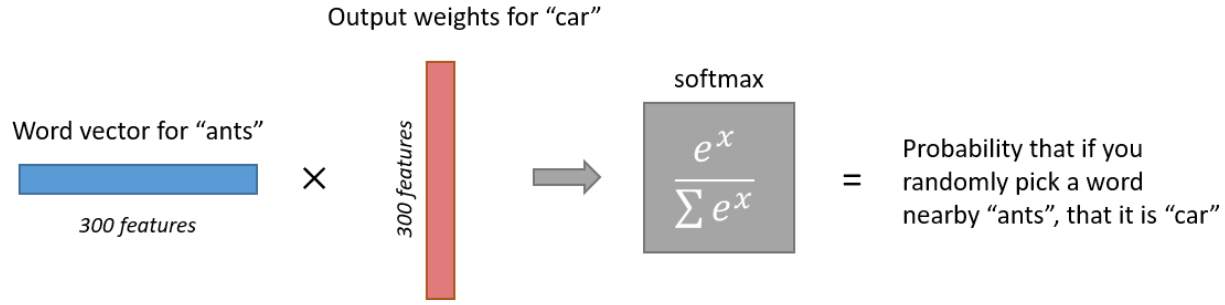
Hình 3.4 Ma trận trọng số của lớp ẩn của mô hình word2vec

Vậy mục đích cuối cùng của việc huấn luyện trên toàn tập ngữ liệu là tìm ra ma trận trọng số tại lớp ẩn. Nhận thấy đầu vào của mô hình là 1 từ được biểu diễn dưới dạng one-hot vector tức là một vector có các giá trị toàn bằng 0, chỉ có một vị trí bằng 1 tương ứng với vị trí của từ đầu vào theo thứ tự từ điển. Việc nhân vector one-hot đầu vào với ma trận trọng số bản chất là việc tìm kiếm trên ma trận trọng số một vector đặc trưng có chiều dài bằng số chiều bằng số chiều của ma trận trọng số.

$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

Hình 3.5 Lớp ẩn của mô hình hoạt động như một bảng tra cứu

Đầu ra của mô hình Word2vec là một bộ phân loại sử dụng hàm softmax để tính xác suất. Ưu điểm của hàm softmax là luôn tạo giá trị xác suất dương và tổng tất cả các xác suất thành phần là bằng 1. Giả sử tính mối tương quan giữa từ "ants" và từ "car", hai từ này sẽ được vector hóa dựa vào ma trận trọng số của lớp ẩn đã huấn luyện. Đầu ra qua hàm softmax sẽ có ý nghĩa là xác suất từ "car" xuất hiện gần từ được chọn "ants"

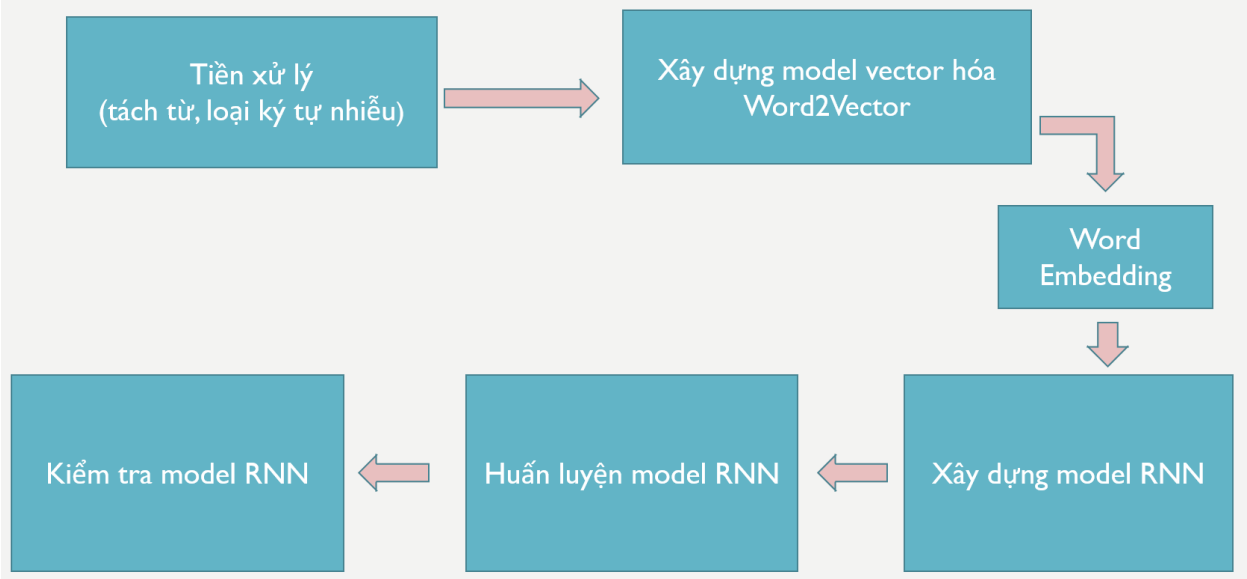


Hình 3.6 Mối tương quan giữa từ "ants" và từ "car"

3.3. Áp dụng LSTM trong bài toán trích xuất thông tin quan điểm

Bài toán trích xuất thông tin quan điểm có đầu vào là tập văn bản bao gồm các ý kiến phản hồi đã được gán nhãn. Số nhãn thường có số lượng bằng hai gồm ý kiến tích cực và tiêu cực. Ngoài ra, trong một số trường hợp số lượng nhãn bằng ba do thêm loại ý kiến trung tính (trung tính là phản hồi không mang tính tiêu cực hay tích cực).

Việc giải bài toán trích xuất thông tin quan điểm sẽ bao gồm việc giải quyết một chuỗi các bài toán nhỏ hơn. Chuỗi các bài toán nhỏ hơn này được gọi là pipeline của mô hình học máy.



Hình 3.7 Pipeline của bài toán trích xuất thông tin quan điểm sử dụng RNN

Như tôi quan sát ở hình 3.7, pipeline khi giải quyết bài toán trích xuất thông tin quan điểm sẽ bao gồm việc giải quyết các vấn đề sau

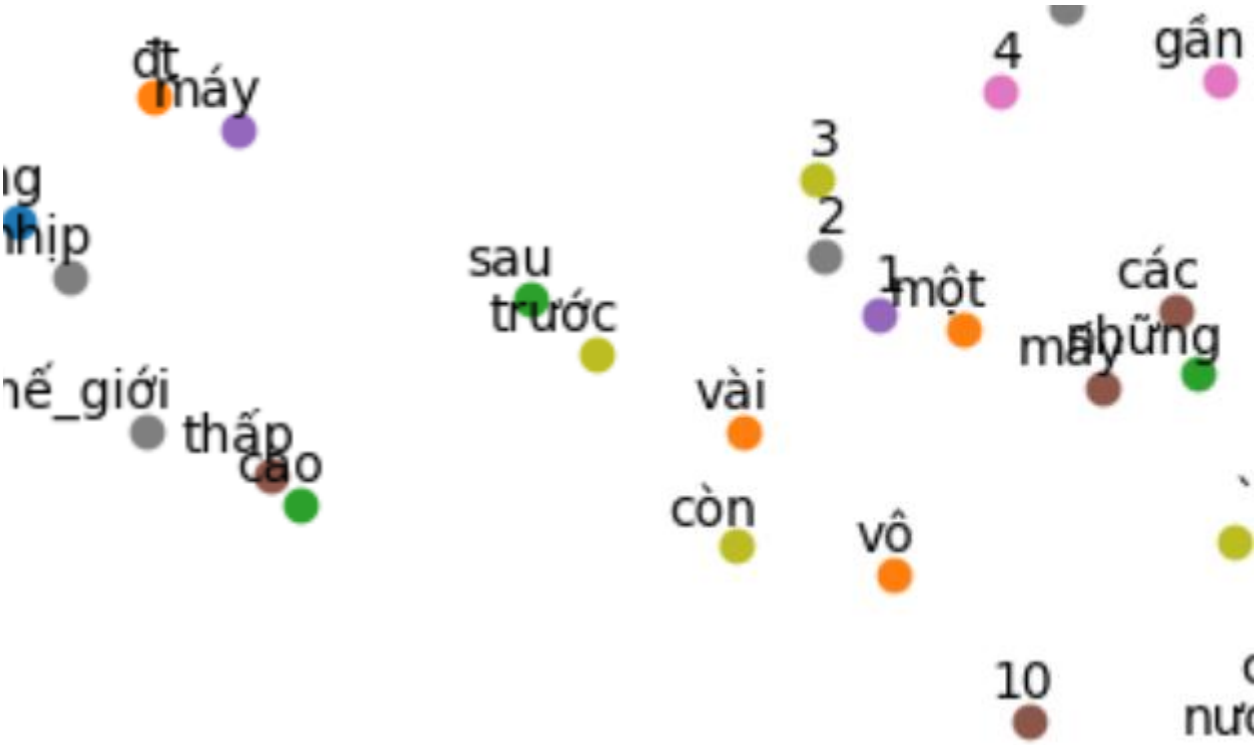
- Tiền xử lý kho ngữ liệu
- Xây dựng model vector hóa Word2vec cho tập ngữ liệu
- Word Embedding sử dụng mô hình kết quả của Word2vec để vector từng câu trong tập ngữ liệu
- Áp dụng mạng RNN để giải quyết bài toàn bao gồm các bước nhỏ: xây dựng model RNN, huấn luyện model RNN, kiểm tra model RNN

3.3.1 Tiền xử lý kho ngữ liệu

Tiền xử lý hay xử lý sơ bộ tập văn bản là bước đầu tiên phải thực hiện trong tất cả các mô hình xử lý ngôn ngữ tự nhiên. Đây là bước tưởng chừng như không quan trọng nhưng lại đem đến hiệu quả không ngờ cho các mô hình học máy. Trong phạm vi luận văn, các bước tiền xử lý kho ngữ liệu được thực hiện bao gồm việc tách từ, loại bỏ những kí tự không hợp lệ, loại bỏ những từ vô nghĩa trong văn bản – stop words. Quá trình thực hiện tách từ trong tiếng Việt sẽ đòi hỏi nhiều công sức hơn so với tiếng Anh. Do từ trong tiếng Việt có nhiều âm tiết như từ ghép trong khi tiếng Anh thì mỗi từ ngăn cách với nhau bằng khoảng trắng. Bộ tách từ Đông Du của tác giả Lưu Tuấn Anh [3] ứng dụng phương pháp pointwise vào bài toán tách từ tiếng Việt cho kết quả khá tốt.

3.3.2 Xây dựng Word2vec

Xây dựng mô hình word2vec từ tập ngữ liệu văn bản đã được tiền xử lý. Mô hình Word2vec bản chất là việc huấn luyện một mạng ANN với một lớp ẩn. Các cặp từ được tách theo skip-gram và dựa trên xác suất để tính độ tương quan giữa các từ. Mô hình Word2vec trong luận văn lựa chọn số đặc trưng là 128. Có thể kiểm tra độ tương quan giữa các từ bằng giảm số chiều đặc trưng bằng thuật toán như PCA để dễ dàng quan sát hơn.

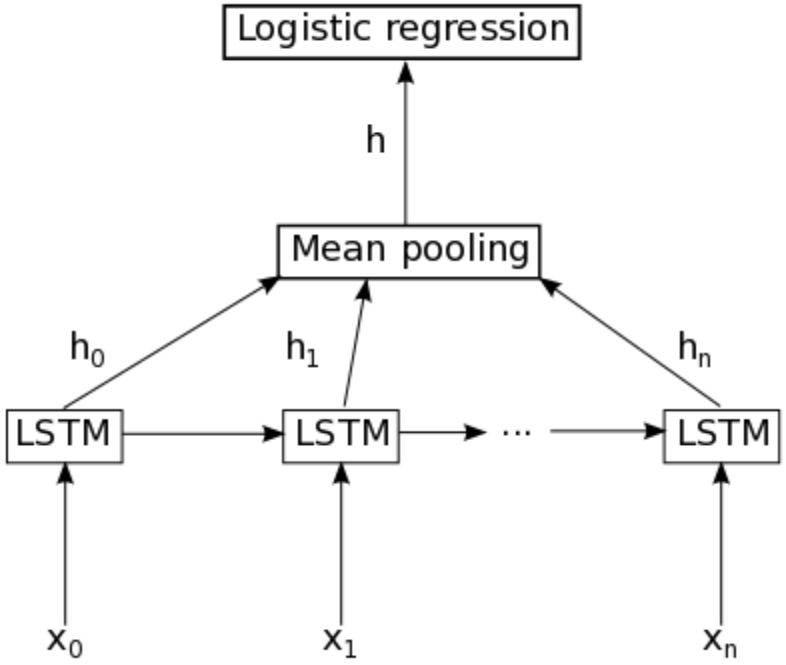


Hình 3.8 Quan sát sự tương quan giữa các từ trong word2vec

Nhận thấy rằng các cặp từ như chỉ số đếm "1", "2", "3", "4", "10", "một" hay "các", "những" hay "thấp", "cao" được đặt gần nhau – có sự tương quan với nhau về ý nghĩa.

3.3.3 Model LSTM

Với ưu điểm về lưu trữ phụ thuộc dài, model sử dụng để huấn luyện trong luận văn này là model LSTM. Mô hình mà luận văn sử dụng được mô tả trong phần 2.4 của luận văn gồm một lớp LSTM duy nhất sau đó là một lớp tổng hợp trung bình (full-connection) và một lớp hồi quy logistic. Các từ được vector hóa sử dụng mô hình Word2vec.



Hình 3.9 Mô hình LSTM sử dụng trong luận văn

Từ một chuỗi đầu vào x_0, x_1,… x_n sử dụng các cơ chế tính toán nêu trên của các cổng vào, cổng ra và cổng chặn sẽ tính được tương ứng giá trị vector trạng thái ẩn h_0, h_1, …h_n . Giá trị vector trạng thái ẩn tại các thời điểm sau đó được tính trung bình trên tất cả các dấu thời gian để được vector trạng thái h. Vector h sẽ đại diện cho câu đang xét. Cuối cùng, vector h được đưa vào một lớp hồi quy để gán nhãn, phân loại cho kết quả đầu ra.

CHƯƠNG 4: KẾT QUẢ THỰC NGHIỆM

4.1 Bộ ngữ liệu

Luận văn sử dụng hai bộ ngữ liệu một tiếng Anh và một tiếng Việt được thu thập từ đánh giá của người dùng. Các kết quả thử nghiệm bao gồm việc turning các hyper parameter trong mô hình LSTM và cuối cùng là so sánh kết quả của LSTM với các thuật toán state-of-art sử dụng cả hai bộ ngữ liệu tiếng Việt và tiếng Anh.

4.1.1 Bộ ngữ liệu tiếng Anh (Food Reviews)

Bộ ngữ liệu tiếng Anh là bộ Food Reviews lấy dữ liệu từ Amazon [17]. Dữ liệu được thu thập trong 10 năm, bao gồm 568.454 đánh giá về sản phẩm đồ ăn trên trang thương mại điện từ Amazon. Dữ liệu bao gồm cả thông tin sản phẩm, thông tin người dùng, xếp hạng ưa thích và phần dữ liệu văn bản ghi lại đánh giá của người dùng.

Dataset statistics	
Number of reviews	568,454
Number of users	256,059
Number of products	74,258
Users with > 50 reviews	260
Median no. of words per review	56
Timespan	Oct 1999 - Oct 2012

Hình 4.1 Bộ ngữ liệu tiếng Anh

Định dạng dữ liệu gồm thông tin sản phẩm (product) và thông tin đánh giá (review)

```
product/productId: B001E4KFG0
review/userId: A3SGXH7AUHU8GW
review/profileName: delmartian
review/helpfulness: 1/1
review/score: 5.0
review/time: 1303862400
review/summary: Good Quality Dog Food
review/text: I have bought several of the Vitality canned dog
food products and have found them all to be of good quality.
The product looks more like a stew than a processed meat and it
smells better. My Labrador is finicky and she appreciates this
product better than most.
```

Hình 4.2 Định dạng dữ liệu bộ Food Reviews

Bộ dữ liệu tiếng Anh được cung cấp dưới dạng file text dung lượng sau giải nén khoảng 360MB, trong đó mỗi review có định dạng như trên. Tại bước tiền xử lý, tôi chỉ quan tâm đến các trường thông tin sau

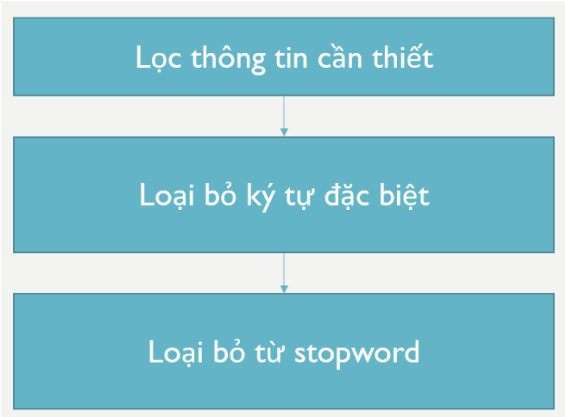
- Product/ productId: định danh sản phẩm
- Review/ score: điểm đánh giá của người dùng thang điểm 5
- Review/summary: đánh giá chung về sản phẩm
- Review/ text: phản hồi của người dùng

Nhận thấy phần đánh giá chung về sản phẩm thường rất ngắn (dưới 10 từ) và phục vụ mục đích nghiên cứu của luận văn là sử dụng ưu điểm của LSTM nên tạm thời bỏ qua thông tin đánh giá chung về sản phẩm. Nếu sử dụng các phương pháp đánh giá khác thì có thể sử dụng thêm thông tin đánh giá chung này.

	Positive	**Neural**	**Negative**
Review/score	4-5	3	0-2
Số lượng đánh giá	443.777	42.640	82.037

Hình 4.3 Phân bố loại câu trong ngữ liệu tiếng Anh

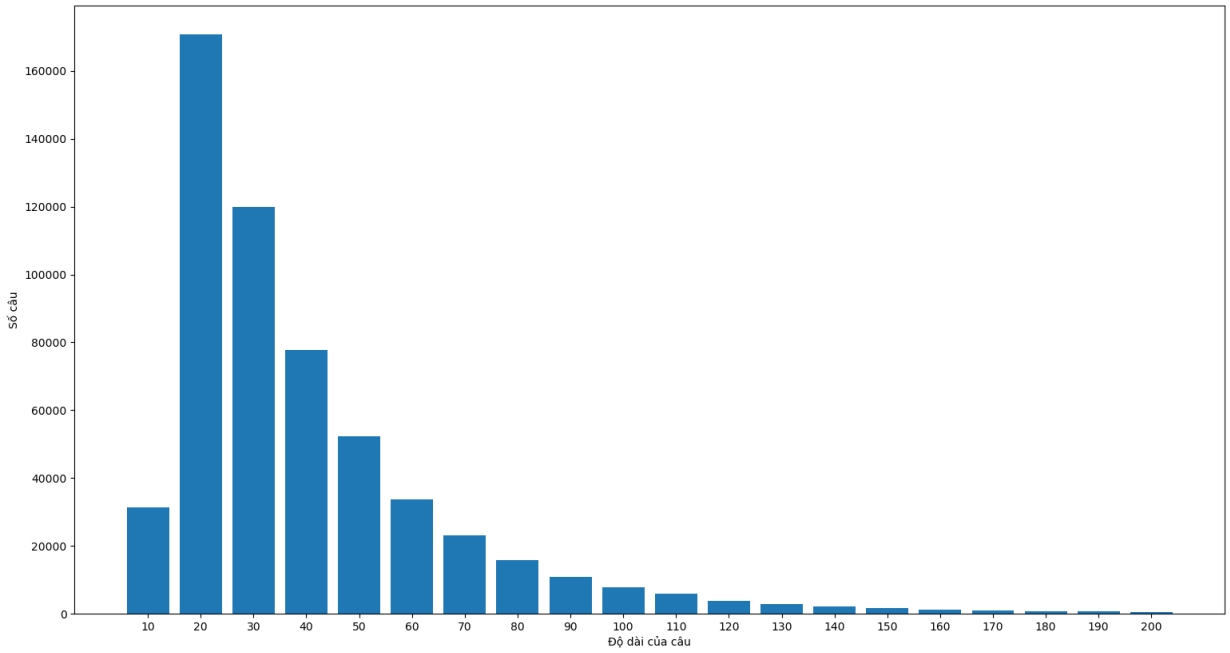
Đối với bộ dữ liệu Food Reviews tôi sử dụng một số phương pháp tiền xử lý như sau



Hình 4.4 Tiền xử lý bộ dữ liệu Food Reviews

Làm một vài khảo sát đối với tập dữ liệu này tôi có một số thông tin như sau: câu dài nhất là 1103 từ; trong đó độ dài câu gồm 13 từ có số lượng câu lớn nhất là 19166 câu. Tính được độ dài câu có mean = 35.29 và sigma = 31.76.

Ta có phân bố số lượng từ trong câu như hình sau



Hình 4.5 Phân bố số câu và độ dài câu

Nhận thấy độ dài của câu tập trung phần lớn vào khoảng 20 từ. Số lượng các câu trên 100 từ là không nhiều. Tổng số lượng từ mới được dùng trong tập ngữ liệu là 50.538 từ sau khi đã loại bỏ các stopword. Stopword là các từ không có giá trị, ít có ý nghĩa trong phân tích văn bản.

{'his', 'because', 'shan', 'own', 'themselves', 'doesn', 'our', 'ourselves', 'up', 'should', 'under', 'most', 'at', 'having', 'where', 'him', 'below', 'am', 'wouldn', 'itself', 'your', 'll', 'from', 'their', 'ain', 'more', 'they', 'have', 'out', 'nor', 'of', 'weren', 'down', 'that', 'into', 'as', 'these', 'both', 'only', 'than', 'here', 'some', 'so', 'herself', 'how', 's', 'on', 'myself', 't', 'has', 'her', 'further', 'himself', 'again', 'hers', 'doing', 'before', 'very', 'just', 'd', 'between', 'in', 'during', 'yourself', 'whom', 'which', 'or', 've', 'what', 'against', 're', 'aren', 'was', 'yours', 'for', 'm', 'don', 'didn', 'she', 'not', 'y', 'been', 'its', 'mustn', 'and', 'ours', 'after', 'them', 'shouldn', 'you', 'few', 'couldn', 'mightn', 'same', 'haven', 'ma', 'be', 'theirs', 'but', 'such', 'wasn', 'were', 'those', 'a', 'to', 'an', 'did', 'too', 'with', 'about', 'who', 'isn', 'we', 'my', 'other', 'needn', 'i', 'when', 'the', 'then', 'once', 'all', 'will', 'won', 'is', 'this', 'he', 'off', 'while', 'yourselves', 'are', 'there', 'it', 'had', 'why', 'hadn', 'hasn', 'through', 'over', 'can', 'until', 'above', 'no', 'being', 'by', 'do', 'any', 'if', 'each', 'o', 'now', 'me', 'does'}

Hình 4.6 Một số stopword trong tiếng Anh

Sau khi loại bỏ stopword, tôi có thể tiến hành build Word2vec cho tập ngữ liệu. Ở đây tôi sử dụng thuật toán skip-gram. Một tham số đáng quan tâm khác là số đặc trưng của 1 từ.

Ví dụ tôi chọn num_feature = 100. Từ love được biểu diễn bằng 1 vector 1x100. Thử kiểm tra sự tương quan của một số từ vựng

print(w2v.wv.most_similar("love",topn=10))
[('awesome', 0.7321419715881348),
('adore', 0.7253466844558716),
('great', 0.7180365324020386),
('cruncha', 0.7085840702056885),
('enjoy', 0.7057753801345825),
('wonderful', 0.7021138668060303),
('fantastic', 0.701633095741272),
('chice', 0.6959320306777954),
('like', 0.6955946683883667),
('loved', 0.6900432109832764)]

print(w2v.wv.most_similar("bad",topn=10))
[('terrible', 0.7505322694778442),
('whang', 0.7224236726760864),
('goog', 0.721787691116333),
('purhaps', 0.7203595638275146),
('worse', 0.7183746099472046),
('unspeakably', 0.7172457575798035),
('awful', 0.7153866291046143),
('privious', 0.7141600251197815),
('persay', 0.7054546475410461),
('robutussin', 0.7046265602111816)]

print(w2v.wv.most_similar("buy",topn=10))
[('buying', 0.8258473873138428),
('purchase', 0.807976484298706),
('recomened', 0.7435208559036255),
('stoe', 0.7422757744789124),
('sell', 0.7328139543533325),
('orer', 0.7265053987503052),
('woow', 0.7204826474189758),
('purchasing', 0.7186599969863892),
('wfs', 0.7183821201324463),
('betwween', 0.7177314758300781)]

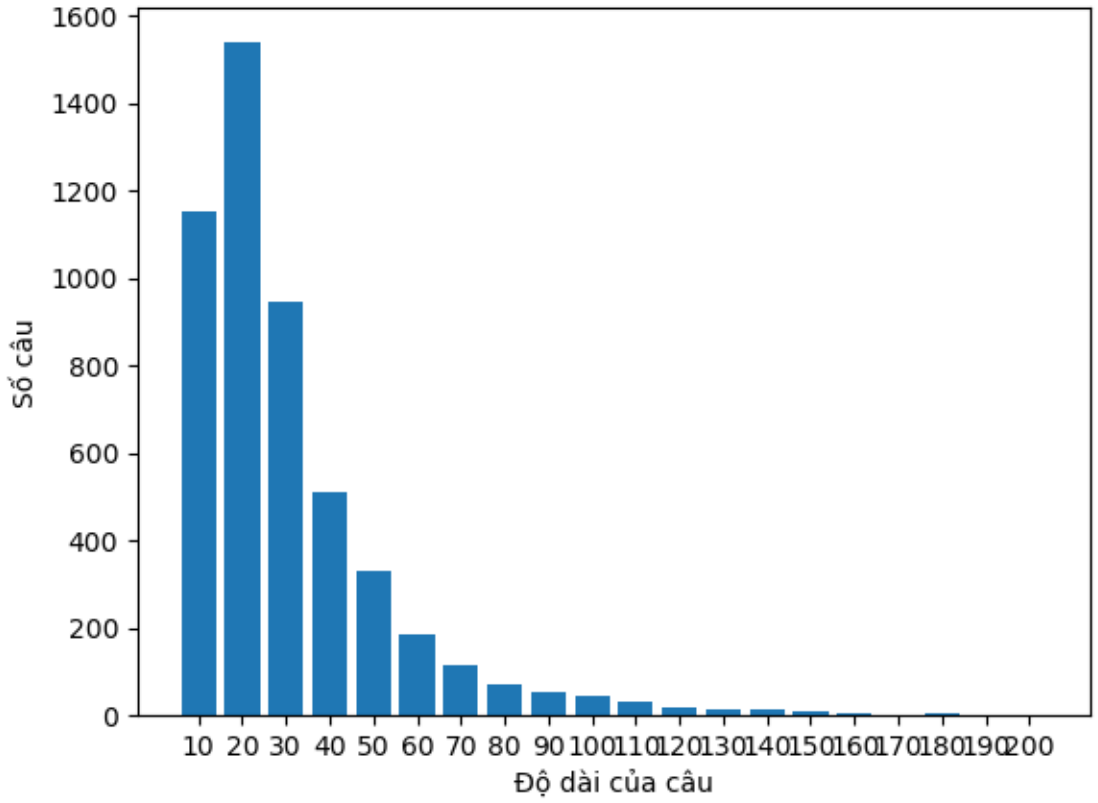
Hình 4.7 Kiểm nghiệm sự tương quan của một số từ trong word2vec bộ tiếng Anh

Nhận thấy bộ word2vec cho kết quả khá tốt khi tôi lấy ví dụ 10 phần tương quan đồng nghĩa nhất với nó. Ví dụ từ love với các từ like, adore, enjoy, wonderful ; từ bad với các từ terrible, worse, awful; từ buy với các từ purchase, buying, sell. Tham số kiểu float bên cạnh các từ thể hiện sự tương quan với từ gốc. Word2vec là một trong những cách thực hiện word embedding tốt cho xử lý ngôn ngữ tự nhiên.

4.1.2 Bộ ngữ liệu tiếng Việt

Bộ ngữ liệu tiếng Việt gồm 5.100 nhận xét về sản phẩm tin học bao gồm 1.700 nhận của tích cực, tiêu cực và trung tính mỗi loại. Tập test bao gồm 1.050 nhận xét trong đó gồm 350 nhận xét mỗi loại. Câu dài nhất là có 2.716 từ và câu ngắn nhất có 1 từ. Trung bình số từ trên câu là 28,4 từ.

Tích cực	**Trung tính**	**Tiêu cực**
1.700	1.700	1.700



Hình 4.8 Phân bố độ dài của tập mẫu tiếng Việt

Một số ví dụ trong tập ngữ liệu

Mình đang dùng với win10, chuột di mượt mà, không lag giật gì bạn à.
Đang dùng con VX nano từ 2009 đến giờ chưa hỏng
Thật không thể tin nổi, dung lượng SSD hiện tại đã lên đến 16TB
Tốc độ ghi nhanh quá. Các phần cứng khác phải cỡ khủng để xứng đôi với raid 0.☺
Mình đến bây giờ vẫn giữ một em PS1 và vẫn chạy tốt
Cấu hình tốt, đẹp, tiện

Hình 4.9 Ví dụ về đánh giá tích cực trong bộ ngữ liệu tiếng Việt

Tương tự như bộ dữ liệu tiếng Anh tôi cũng có các bước tiền xử lý dữ liệu như xử lý encoding, xử lý parser cho từ, loại bỏ ký tự đặc biệt, loại bỏ từ stopword.

bị	cứ	gì	như
bởi	của	khi	nhưng
cả	cùng	không	những
các	cũng	là	nơi
cái	đã	lại	nữa
cần	đang	lên	phải
càng	đây	lúc	qua
chỉ	để	mà	ra
chiếc	đến_nỗi	mỗi	rằng
cho	đều	một_cách	rằng
chứ	điều	này	rất
chưa	do	nên	rất
chuyện	đó	nếu	rồi
có	được	ngay	sau
có_thể	dưới	nhiều	sẽ

Hình 4.10 Một số stopword trong tiếng Việt

4.2 Cài đặt và thử nghiệm

Các thử nghiệm được cài đặt sử dụng ngôn ngữ python [16] trên môi trường python 3.6. Một số thư viện của python sử dụng trong thực nghiệm gồm:

Thư viện	
Numpy	Thư viện xử lý mảng, ma trận thực hiện các phép tính như nhân ma trận, tính ma trận chuyển vị …
Re	Thư viện về biểu thức chính quy Regular Expression
Pandas	Đọc dữ liệu lớn
Sklearn	Thư viện hỗ trợ cài đặt các thuật toán cơ bản như SVM, ANN
Gensim	Thư viện hỗ trợ cài đặt mô hình Word2vec
TensorFlow	Thư viện rất mạnh cho học máy hỗ trợ cài đặt mô hình, huấn luyện và kiểm thử mô hình
Matplotlib	Thư viện vẽ các loại đồ thị và hình

4.2.1 Bước tiền xử lý

Tiền xử lý là bước quan trọng không kém so với các bước xây dựng mô hình toán. Theo Andrew Ng [8] tiền xử lý tốt mang lại kết quả tốt không ngờ cho toàn mô hình. Tại bước tiền xử lý, tôi chủ yếu thực hiện việc loại bỏ những ký tự HTML, những ký tự không phải là chữ cái. Hàm loại bỏ các ký tự nhiễu đầu vào là một phản hồi khách hàng và đầu ra là phản hồi đã được làm mịn. Mã python của hàm loại bỏ ký tự nhiễu có dạng:

```
def clean_sentence(sentence):
    # Remove HTML
    review_text = BeautifulSoup(sentence).text

    # Remove non-letters
    letters_only = re.sub("[^a-zA-Z]", " ", review_text)
    return letters_only
```

Tiếp đó, tôi thực hiện loại bỏ những từ stopword trong phản hồi

```
def review_to_words(review):
    """
    Function to convert a raw review to a string of words
    :param review
    :return: meaningful_words
    """
    # 1. Convert to lower case, split into individual words
    words = review.lower().split()
    #
    # 2. In Python, searching a set is much faster than searching
    #   a list, so convert the stop words to a set
```

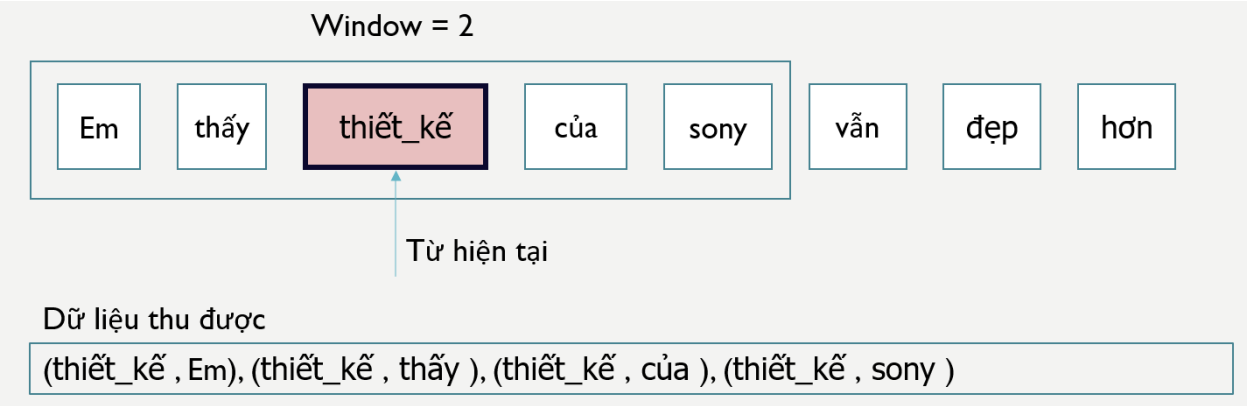
```
    stops = set(stopwords.words("english"))
    #
    # 3. Remove stop words
    meaningful_words = [w for w in words if not w in stops]
    #
    # 4. Join the words back into one string separated by space,
    # and return the result.
    return " ".join(meaningful_words)
```

Đối với bộ ngữ liệu tiếng Việt cần thêm bước tách từ, ở đây có thể dùng một số công cụ tách từ có sẵn như Đông Du [3] của tác giả Lưu Tuấn Anh.

4.2.2 Xây dựng model Word2vec

Từ mảng các phản hồi đã được tiền xử lý, thực hiện xây dựng mô hình Word2vec. Mô hình Word2vec xây dựng một từ điển các từ và giá trị ánh xạ vector cho từ đó.

Khi đưa một câu vào, dựa trên giá trị window tôi sẽ tách được các cặp từ mô tả sự xuất hiện của từ hiện tại với từ xung quanh. Giả sử đối với câu "Em thấy thiết kế của sony vẫn đẹp hơn", hình dưới đây mô tả việc lấy các cặp từ để đưa vào huấn luyện khi từ hiện tại là "thiết kế".



Hình 4.11 Cách lấy cặp từ đưa vào huấn luyện Word2vec

Bản chất huấn luyện Word2vec sẽ dựa vào tần suất xuất hiện của các cặp từ để dự đoán từ tiếp theo trong câu. Từ đó, tính toán tối ưu hàm mất mát và cập nhật các tham số feature của từ. Xây dựng model word2vec sử dụng thư viện Gensim như sau.

```
from gensim.models import Word2vec
model = Word2vec(doc, size=100, window=10, min_count=3, workers=4, sg=1);
model.save("food.w2v")
```

- min_count: giá trị ngưỡng của từ. Những từ có tần suất xuất hiện lớn hơn min_count mới được đưa vào mô hình word2vec

- Window: giá trị của cửa sổ từ. Tại vị trí hiện tại của từ đang xét sẽ ghi nhận giá trị window từ đứng trước và đứng sau từ hiện tại.
- Size: số lượng feature mong muốn
- Sg: sử dụng thuật toán CBOW hoặc skip-model để huấn luyện

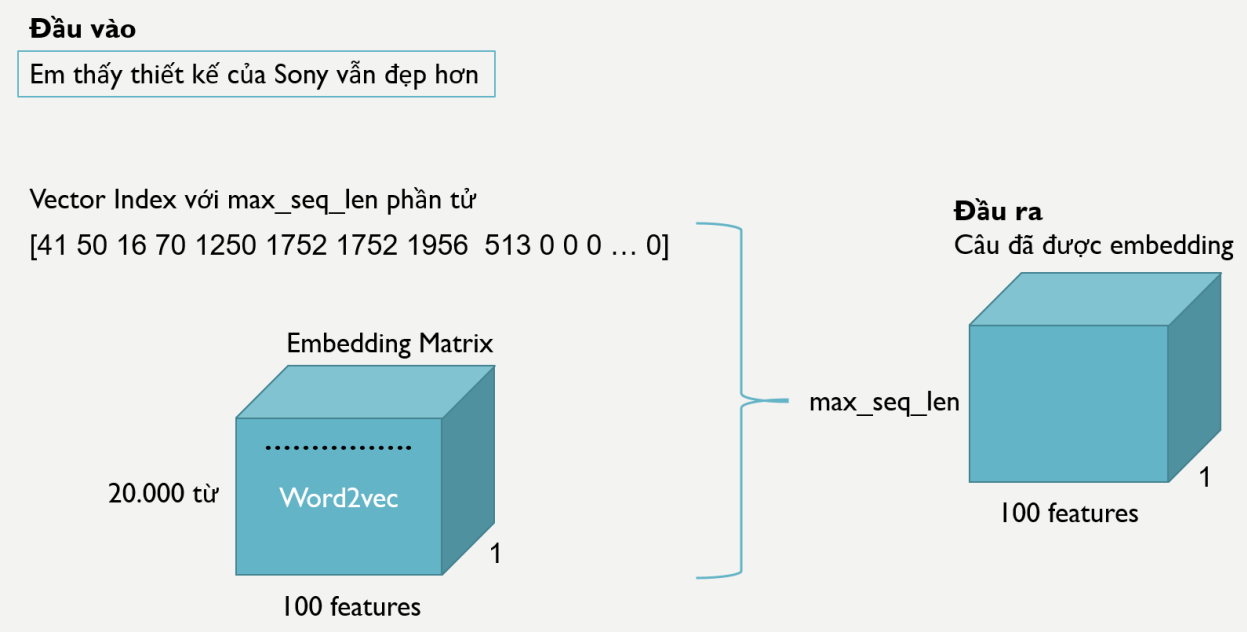
Kết quả là mỗi từ được biểu diễn dưới dạng vector 100x1. Ví dụ từ good

```
w2v.wv['good']
array([-0.21943139, -0.33590445, -0.0489771 , -0.14578219, -0.17717394,
       0.04256329,  0.02610714, -0.03540136, -0.10647894, -0.10235822,
       0.19485623, -0.35626093, -0.00579968, -0.19745331,  0.01853223,
       0.08233038, -0.06455436,  0.04178619, -0.25828445, -0.00862685,
       0.31168512,  0.00802558,  0.24427734, -0.33647063,  0.00961189,
       0.0858779 ,  0.07523053,  0.18785904, -0.15984604,  0.04393168,
       0.30591741, -0.04175724, -0.30127776,  0.18953446,  0.1991684 ,
       0.13903525,  0.02654658,  0.00877954,  0.05743998, -0.15498054,
       0.24042086, -0.14992148,  0.15259801, -0.01032948, -0.35611239,
      -0.15097243,  0.05192766,  0.09714656,  0.0888728 , -0.26703352,
      -0.12232982,  0.09373455,  0.09741747, -0.25320995, -0.03402151,
      -0.02187909,  0.04218853,  0.03189047,  0.14396758,  0.05118875,
      -0.3612909 ,  0.12412404, -0.39866322,  0.14960717, -0.12257327,
       0.1179563 ,  0.11216327,  0.07519023,  0.11648606,  0.18818906,
       0.28355086,  0.02805633,  0.06429619, -0.12048437,  0.01799544,
      -0.31031793, -0.10182056,  0.31299064, -0.09184895,  0.01041629,
       0.18477698, -0.04363374,  0.37875053,  0.22910933,  0.27517578,
      -0.25513521, -0.06690233, -0.07902425,  0.05865611, -0.04321217,
      -0.03790821, -0.0811172 , -0.03884944, -0.05603766,  0.35733798,
      -0.39968881, -0.09622443, -0.08815863, -0.20409873, -0.0056514 ], dtype=float32)
```

4.2.3 Word Embedding

Word Embedding là quá trình đưa các từ trong câu về dạng để mô hình toán có thể hiểu được. Cụ thể là từ dạng text, các từ sẽ được chuyển về dạng vector đặc trưng để đưa vào mô hình LSTM. Trước khi đưa về dạng vector các câu cần được chuẩn hóa về độ dài. Chọn max_seq_len là độ dài của câu, khi đó tất cả các câu trong tập huấn luyện đều được cắt hoặc nối để có độ dài max_seq_len.

Khi một câu được đưa vào, trước tiên nó sẽ được embedding theo số index tương ứng của nó trong từ điển. Sau đó, dựa trên từ điển và kết quả word2vec thu được tôi embedding toàn bộ câu dưới dạng ma trận như hình dưới đây.



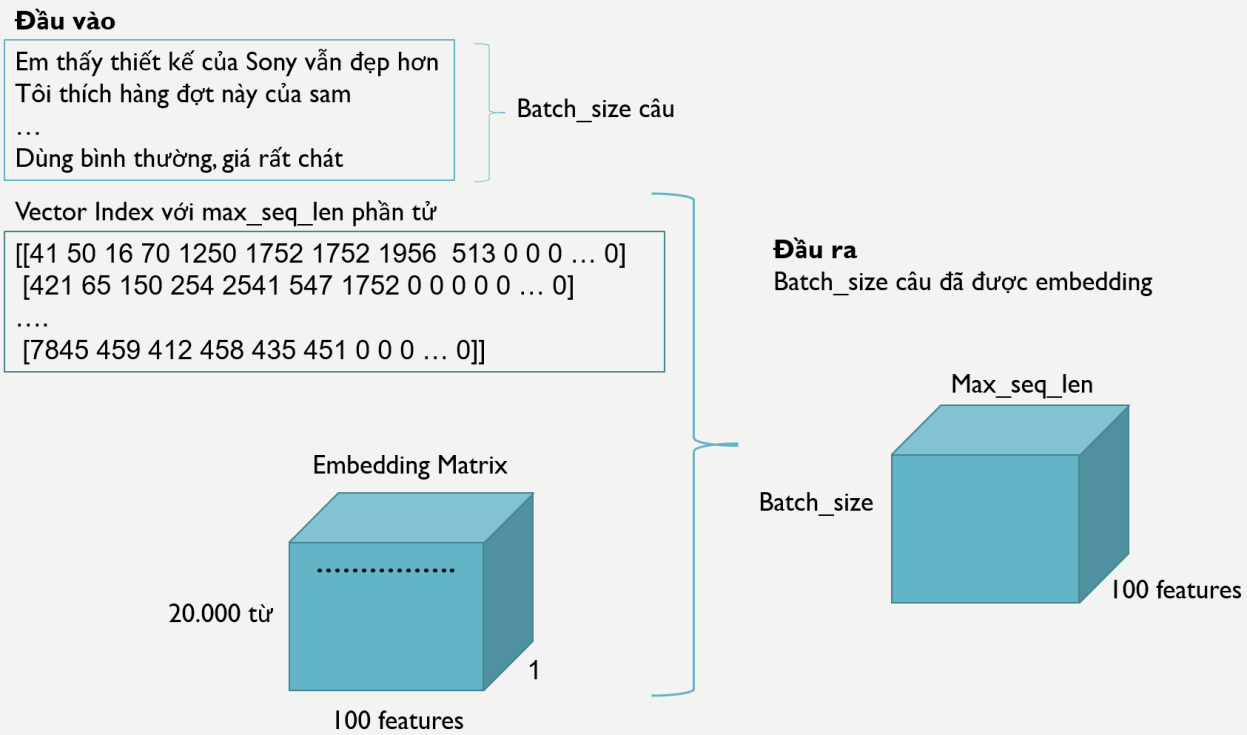
Hình 4.12 Quá trình word embedding của 1 câu

Tương ứng nhãn của câu cũng được embedding theo bảng sau

Tích cực	[1,0,0]
Trung tính	[0,1,0]
Tiêu cực	[0,0,1]

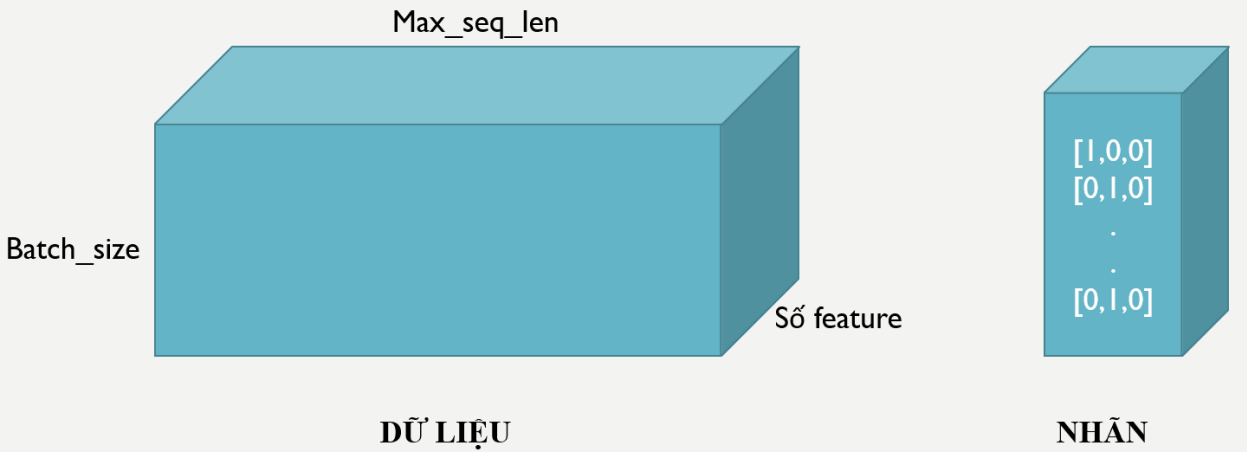
4.2.4 Huấn luyện mô hình LSTM

Huấn luyện mô hình tôi sẽ đưa vào mô hình batch_size số câu trong một lượt huấn luyện. Cách đưa vào batch_size chứ không đưa toàn bộ mô hình dựa trên tư tưởng của thuật toán Mini-batch Gradient Decent. Thuật toán sẽ lấy ngẫu nhiên và không lặp lại batch_size bộ dữ liệu từ tập huấn luyện. Mô tả quá trình word embedding với batch_size câu như sau.



Hình 4.13 Đưa batch_size câu vào mô hình huấn luyện

Bài toán học có giám sát này dữ liệu và nhãn được đưa về dạng như sau.



Hình 4.14 Dữ liệu và nhãn sau khi word embedding

Để xây dựng mô hình LSTM tôi sử dụng thư viện TensorFlow [18], một mã nguồn mở rất mạnh trong học máy hiện đang được nhiều hãng lớn như Google sử dụng trong các sản phẩm thương mại. Trước tiên, tôi cần tạo TensorFlow graph. Để xây dựng TensorFlow graph, tôi định nghĩa một số siêu tham số (hyperparameter) như batch_size, số lượng LSTM units, số lượng vòng lặp khi train.

```
vocab_size = 20000
```

```
batch_size = 512
lstm_units = 64
iterations = 100000
```

Đối với TensorFlow graph, tôi định nghĩa 2 placeholders dữ liệu và nhãn dựa trên số chiều của ma trận tương ứng.

```
import TensorFlow as tf
tf.reset_default_graph()

labels = tf.placeholder(tf.float32, [batch_size, numClasses])
input_data = tf.placeholder(tf.int32, [batch_size, max_seq_len])
data = tf.Variable(tf.zeros([batch_size, max_seq_len, num_feature]),dtype=tf.float32)
data = tf.nn.embedding_lookup(wordVectors,input_data)
```

Sử dụng hàm embedding_lookup cho việc embedding batch_size câu đầu vào. Số chiều của data sẽ là (batch_size x max_seq_len x num_feature). tôi đưa data vào mô hình LSTM bằng việc sử dụng hàm tf.nn.rnn_cell.BasicLSTMCell. Hàm BasicLSTMCell đầu vào là 1 siêu tham số lstm_units là số lượng units trong layer của LSTM. Tham số này phải được tinh chỉnh phù hợp đối với mỗi tập dữ liệu để đạt kết quả tốt nhất. Ngoài ra, khi huấn luyện mô hình mạng neural, tôi nên dropout bớt các tham số để tránh mô hình bị overfitting.

```
lstmCell = tf.contrib.rnn.BasicLSTMCell(lstm_units)
lstmCell = tf.contrib.rnn.DropoutWrapper(cell=lstmCell, output_keep_prob=0.75)
value, _ = tf.nn.dynamic_rnn(lstmCell, data, dtype=tf.float32)
```

Việc mô hình hóa LSTM tôi có nhiều cách để xây dựng. tôi có thế xếp chồng nhiều lớp LSTM lên nhau, khi đó vector ẩn cuối cùng của lớp LSTM thứ nhất sẽ là đầu vào của lớp LSTM thứ 2. Việc xếp chồng nhiều lớp LSTM lên nhau được coi là cách rất tốt để lưu giữ phụ thuộc ngữ cảnh xa lâu dài. Tuy nhiên vì thế số lượng tham số sẽ tăng gấp số lớp lần, đồng thời cũng tăng thời gian huấn luyện, cần thêm dữ liệu và dễ bị overfitting. Trong khuôn khổ của các tập dữ liệu thu thập được trong luận văn, tôi sẽ không xếp chồng các lớp LSTM vì những thử nghiệm với nhiều lớp LSTM không hiệu quả và gây overfitting. Đầu ra của mô hình LSTM là một vector ẩn cuối cùng, vector này được thay đổi để tương ứng với dạng vector kết quả đầu ra bằng cách nhân với ma trận trọng số.

```
weight = tf.Variable(tf.truncated_normal([lstm_units, numClasses]))
bias = tf.Variable(tf.constant(0.1, shape=[numClasses]))
value = tf.transpose(value, [1, 0, 2])
last = tf.gather(value, int(value.get_shape()[0]) - 1)
prediction = (tf.matmul(last, weight) + bias)
```

Tính toán độ chính xác (accuracy) dựa trên kết quả dự đoán của mô hình và nhãn. Kết quả dự đoán mô hình càng giống với kết quả nhãn thực tế thì mô hình càng có độ chính xác cao.

```
correctPred = tf.equal(tf.argmax(prediction,1), tf.argmax(labels,1))
accuracy = tf.reduce_mean(tf.cast(correctPred, tf.float32))
```

Kết quả dự đoán của mô hình không phải luôn luôn giống nhãn, đó gọi la lỗi. Để huấn luyện mô hình tôi cần tối thiểu hóa giá trị lỗi này. Định nghĩa một hàm tính lỗi cross entropy và một layer softmax sử dụng thuật toán tối ưu Adam với learning_rate được lựa chọn như một siêu tham số.

```
loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=prediction,
labels=labels))
optimizer = tf.train.AdamOptimizer(learning_rate=0.0001).minimize(loss)
```

Lưu trữ độ chính xác và giá trị hàm lỗi qua từng vòng lặp khi huấn luyện sửa dụng tensorboard.

```
sess = tf.InteractiveSession()
saver = tf.train.Saver()
tf.summary.scalar('Loss', loss)
tf.summary.scalar('Accuracy', accuracy)

logdir = "tensorboard/" +"dict="+str(vocab_size) + "_maxSeq=" + str(maxSeqLength) +
"_batch=" + str(batchSize) + "_dimens=" + str(numDimensions) + "/"
writer = tf.summary.FileWriter(logdir, sess.graph)
merged = tf.summary.merge_all()
```

Thực hiện các thử nghiệm với mô hình LSTM có rất nhiều loại tham số cần turning thay đổi đối với mỗi tập dữ liệu. Ví dụ như lựa chọn giá trị learning_rate, lựa chọn hàm tối ưu, số lượng units LSTM, kích thước từ điển, số lượng đặc trưng của từ, số vòng lặp thực hiện huấn luyện LSTM … Dựa trên rất nhiều thử nghiệm, tôi sẽ rút ra được một số tham số ảnh hưởng nhiều hay ít đến kết quả thực hiện huấn luyện. Từ đó, tôi có thể rút ra được nhiều kết luận bổ ích của thực nghiệm.

4.2.5 Cài đặt một số phương pháp học có giám sát kinh điển

Việc cài đặt một số thuật toán như SVM, KNN có vai trò so sánh kết quả đối với thuật toán LSTM mà tôi đã xây dựng. Để cài đặt các thuật toán này, tôi có thể sử dụng thư viện sklearn [20] rất dễ dàng sau khi dữ liệu đã được word embedding.

4.3 Kết quả trích xuất thông tin quan điểm

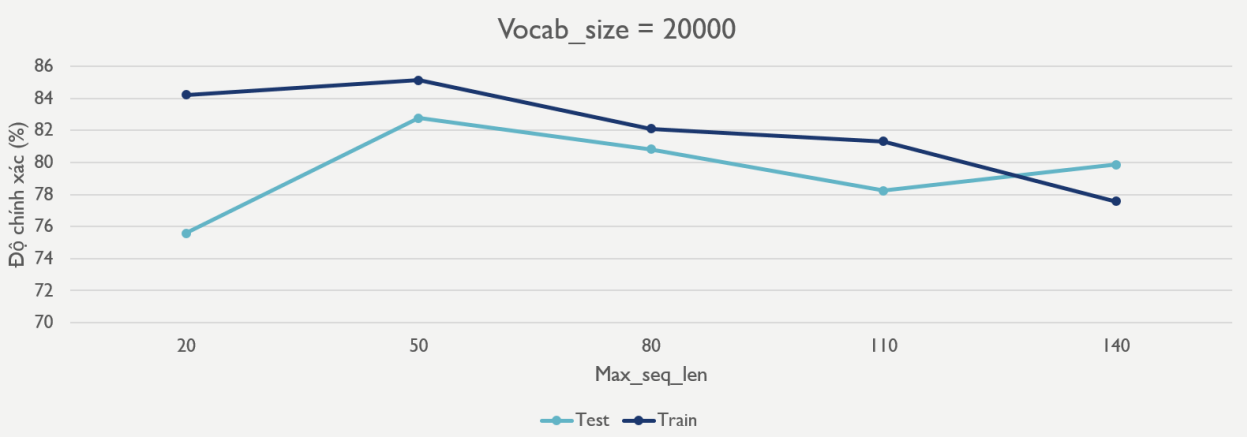
4.3.1 Một số thử nghiệm và kết quả trên bộ ngữ liệu tiếng Anh

Việc huấn luyện mô hình LSTM cho kết quả đầu ra phụ thuộc vào nhiều yếu tố như các siêu tham số. Khi thay đổi các tham số để tối ưu cho mô hình, tôi sẽ phải làm rất nhiều các thử nghiệm. Để đánh giá được một hay vài tham số có ý nghĩa hơn so với các tham số khác tôi sẽ thực hiện tinh chỉnh và căn cứ vào đường học (Learning Curve) để đánh giá. Những thử nghiệm trong luận văn, tôi đã lựa chọn những tham số có ý nghĩa về mặt ngôn ngữ để đánh giá. Chi tiết tôi chia bộ dữ liệu tiếng Anh làm 2 tập train và test theo tỉ lệ 60/40 và thực hiện các thử nghiệm như sau.

Thử nghiệm 1: Giữ số lượng từ vựng bằng 20000 (vocab_size = 20000)

Số lượng từ của tập ngữ liệu được tính toán ở trên là 50.538, tuy nhiên tôi thử chọn 20.000 từ được sử dụng nhiều nhất để làm từ điển. Thay đổi độ dài cho phép của câu đầu vào (max_seq_len). Max_seq_len có tác dụng truncate chuỗi các câu đầu vào thành câu có độ dài là max_seq_len, trong đó những câu có độ dài nhỏ hơn được điền tiếp 1 số ký tự đặc biệt và câu có độ dài lớn hơn thì được cắt đi chỉ còn độ dài max_seq_len

Max_seq_len	Độ chính xác (Train)	Độ chính xác (Test)
25	84.23 %	75.57 %
50	85.12 %	82.76 %
80	82.11 %	80.82 %
110	81.31 %	78.23 %
140	77.57 %	79.85 %

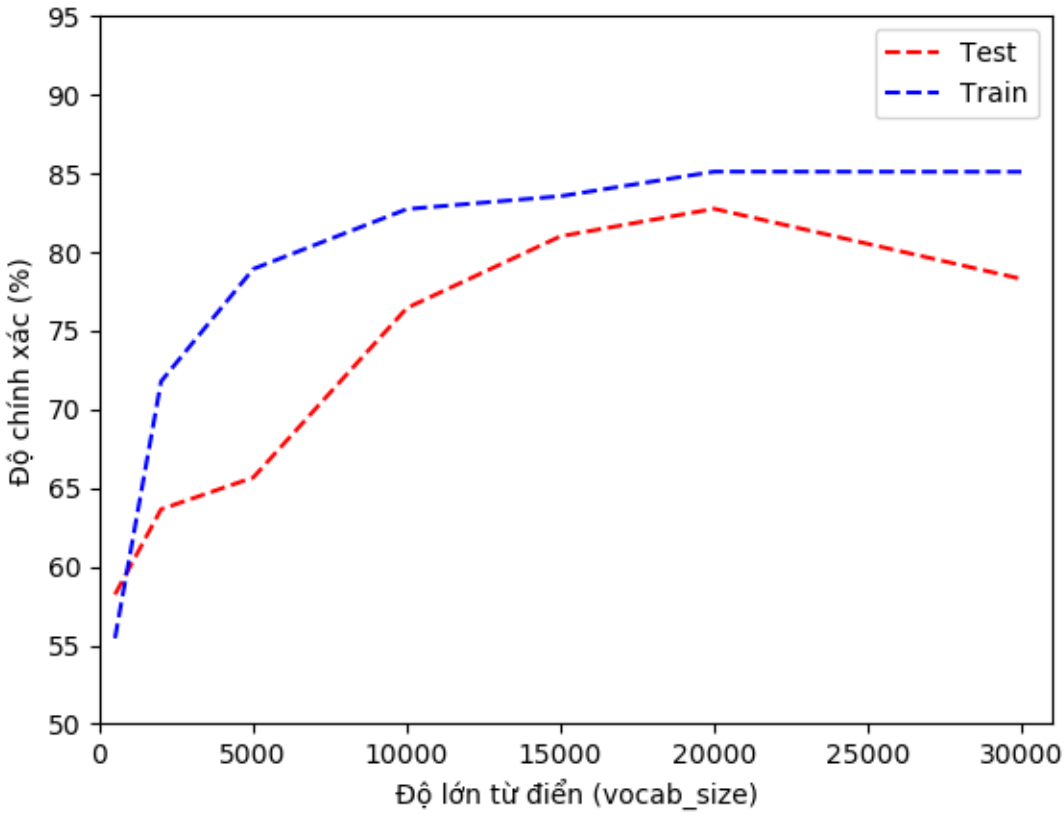


Hình 4.15 Kết quả thử nghiệm với số lượng từ vựng 20.000

Nhận xét, số lượng từ vựng không đổi thì max_seq_len cho kết quả tốt nhất với độ dài bằng 50 từ. Với số từ bằng 50 tương ứng với trên 80% câu trong tập mẫu do đó tôi thấy giá trị này đại diện khá tốt cho độ dài của câu.

Thử nghiệm 2: Giữ độ dài từ mỗi câu là 50 từ

Giữ max_seq_len = 50, thay đổi độ lớn của từ điển. Thay đổi độ lớn của từ điển ảnh hưởng khá lớn đến kết quả bởi nếu số lượng từ nhỏ sẽ có quá nhiều từ trong tập mẫu sẽ không có trong từ điển; nếu số lượng lớn thì số lượng từ được nhận ra sẽ nhiều khi sử dụng word2vec với số lượng đặc trưng lớn (khoảng 300) thì độ phức tạp tính toán sẽ tăng lên rất nhiều.



Hình 4.16 Thử nghiệm với độ dài câu bằng 50 từ

Thử nghiệm 3: So sánh với một số phương pháp khác

Các phương pháp được so sánh gồm KNN, SVM, Gaussian, ANN. Kết quả cho thấy sử dụng LSTM cho kết quả khá khả quan.

Thuật toán	Độ chính xác	
	Train	Test
Nearest Neighbors accuracy	74.63%	78.32%
Linear SVM accuracy	79.55%	81.82%
Gaussian Process accuracy	79.52%	79.68%
Neural Net accuracy	79.52%	79.12%
LSTM	85.12%	82.76%

Hình 4.17 Kết quả trên bộ ngữ liệu tiếng Anh

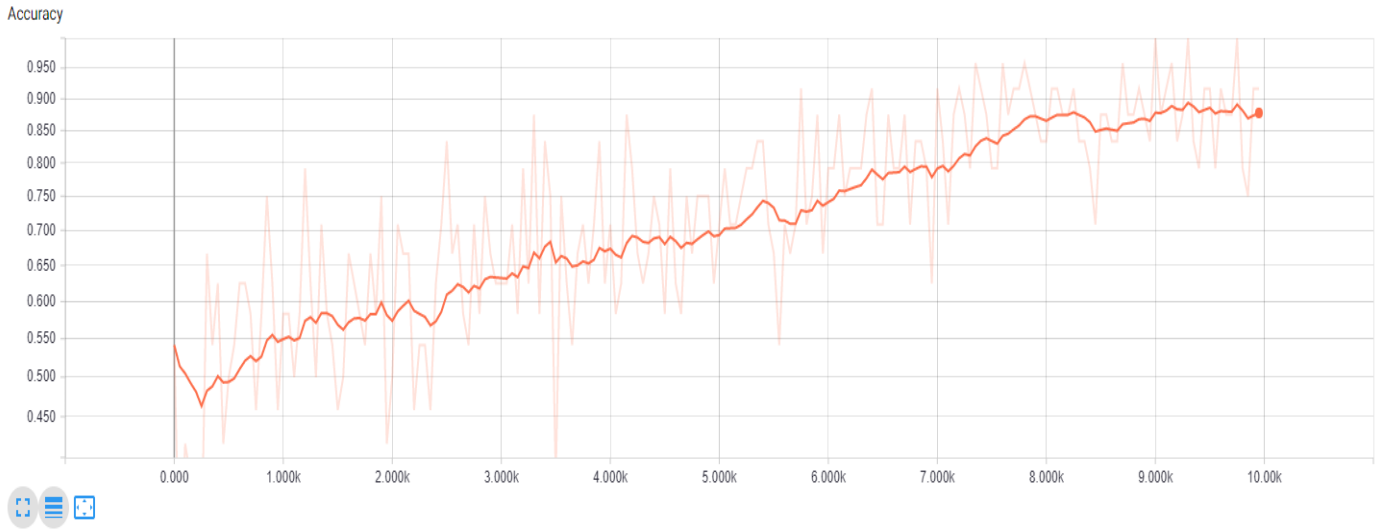
4.3.2 Một số thử nghiệm và kết quả trên bộ ngữ liệu tiếng Việt

Thuật toán	Độ chính xác	
	Train	Test
Nearest Neighbors accuracy	55.7%	38.5%
Linear SVM accuracy	56.9%	40.5%
Gaussian Process accuracy	62.3%	42.9%
Neural Net accuracy	73.3%	41.3%
LSTM	87.83%	43.7%

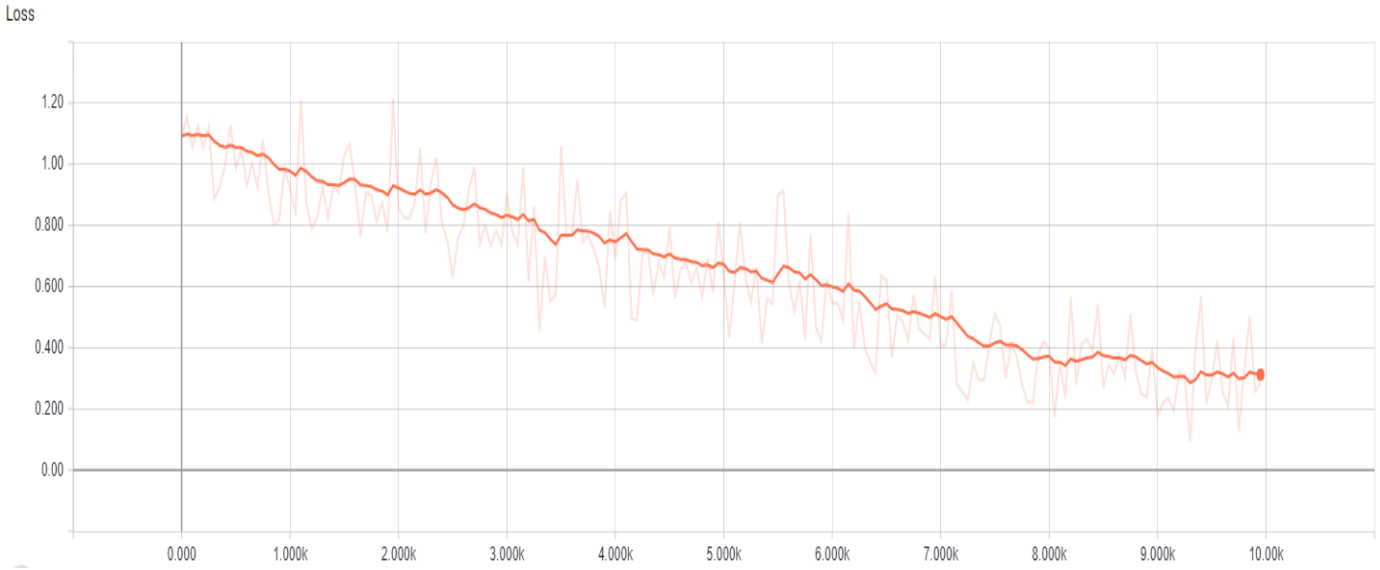
Hình 4.18 Kết quả trên bộ ngữ liệu tiếng Việt

Bộ ngữ liệu tiếng Việt hiện tại có số lượng câu còn ít, ngoài ra có rất nhiều từ bị viết tắt, viết sai theo các cách khác nhau. Ví dụ như để chỉ "không" – tập dữ liệu có các từ "ko","k","khog". Khi áp dụng những thuật toán như word2vec để tính toán word embedding thường cho số lượng tham số lớn dễ gây hiện tượng overfitting.

Kết quả tốt nhất hiện ghi nhận sử dụng vocab_size = 2000, max_seq_len = 20, số feature của word2vec bằng 50, tuy nhiên vẫn bị overfitting.



Hình 4.19 Độ chính xác trong quá trình train bộ dữ liệu tiếng Việt với LSTM



Hình 4.20 Hàm chi phí trong quá trình train bộ dữ liệu tiếng Việt với LSTM

4.4 Nhận xét

Kết quả trên bộ ngữ liệu tiếng Anh là khá tốt, kết quả khi sử dụng model LSTM cho kết quả tốt hơn so với các thuật toán SVM, KNN, Gaussian hay ANN. Trong tập dữ liệu tiếng Anh đã chọn một số tham số như sau

- Số feature of vector = 128
- Dropout = 0.8
- Activation = ‘softmax’
- Optimizer = ‘adam’
- Learning_rate = 0.001

Kết quả bộ ngữ liệu tiếng Việt bị overfitting. Hiện tượng này xảy ra khi độ chính xác trên tập train tốt nhưng độ chính xác trên tập test lại rất thấp. Nguyên nhân được xác định là do bộ ngữ liệu tiếng Việt có số lượng mẫu ít, khi train trong mạng neural có nhiều tham số rất không tốt và hay dẫn đến overfitting. Việc này không thể cải thiện kể cả khi dropout thêm. Sau khi quan sát bộ ngữ liệu tiếng Việt thì thấy có rất nhiều từ là tên riêng (Ví dụ: iphone, asus) hay viết tắt (Ví dụ: k thay cho không) dù đã loại bỏ stopword. Đây thực sự là thách thức trong việc thu thập dữ liệu tự nhiên đặc biệt bằng tiếng Việt.

CHƯƠNG 5: KẾT LUẬN

Mạng neural LSTM có thể được sử dụng rộng rãi trong bài toán xử lý ngôn ngữ tự nhiên như sentiment analysis. Đặc biệt là có thể tận dụng được ưu điểm của việc xử lý dạng chuỗi và thứ tự các từ trong câu. Tuy nhiên, các nghiên cứu LSTM cho sentiment analysis chưa tận dụng được đầy đủ các tài nguyện về sentiment như Sentiment lexicon, từ phủ định hay từ chỉ mức độ.

Với việc định nghĩa max_seq_len thì cách làm này là chấp nhận được đối với tập ngữ liệu mà luận văn sử dụng. Tập ngữ liệu là tập phản hồi của người dùng có số lượng từ không lớn hơn 100. Do đó, có thể xem xét việc lấy max_seq_len số từ đưa vào LSTM để huấn luyện là có thể tổng quát hóa được câu cần xét. Tuy nhiên, đối với tập phản hồi có số từ lớn hơn thì tôi phải xem xét việc vector hóa mà không làm mất mát quá nhiều ý nghĩa của câu do việc chọn đại diện max_seq_len không là không đủ để đại diện cho câu. Một phương pháp thường được sử dụng là dùng TF-IDF kết hợp với một thuật toán giảm số chiều như LDA (Linear Discriminant Analysis).

LSTM là một mô hình kỹ thuật hiệu quả trong bài toán xử lý chuỗi và hiện đang được các nhà nghiên cứu sử dụng rất nhiều. Tuy nhiên, LSTM không phải là một kỹ thuật vạn năng mà cứ bài toán về NLP là lại áp dụng được. Nó còn căn cứ vào nhiều yếu tố như tập ngữ liệu, đặc tính của tập ngữ liệu. Vì đôi khi sử dụng một thuật toán ML lại cho kết quả tốt hơn như SVM, Decision Tree hay ANN.

Nhận thấy rằng, những nghiên cứu gần đây sử dụng các phương pháp học máy và Deep Learning giống như trận sóng thần áp đảo trong NLP. Tuy nhiên, người làm vẫn nên trú trọng bổ sung các kiến thức về ngôn ngữ học và semantic. Bởi ngoài việc trong một vài trường hợp, việc sử dụng một vài rule là cách giải quyết tối ưu nhất so với việc train một mô hình ngôn ngữ đồ sộ. Mà nhờ các kiến thức về ngôn ngữ học, người nghiên cứu có thể cân nhắc được mô hình NLP tốt nhất có thể giải quyết bài toán cũng như biểu diễn đầu vào bằng những đặc trưng có ý nghĩa.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1] Bùi Công Cường, Nguyễn Doãn Phước (2001). Hệ mờ, mạng nơ-ron và ứng dụng. Nhà xuất bản Khoa học và kỹ thuật. Hà Nội.

[2] Vũ Hữu Tiệp, Blog Machine Learning Cơ bản tại địa chỉ https://machinelearningcoban.com/

[3] Lưu Tuấn Anh (2012), Bộ tách từ Đông Du https://github.com/rockkhuya/DongDu

Tiếng Anh

[4] Hochreiter and Schmidhuber (1997), Long short-term memory

[5] B. Liu (2009), Handbook Chapter: Sentiment Analysis and Subjectivity. Handbook of Natural Language Processing, Handbook of Natural Language Processing. Marcel Dekker, Inc. New York, NY, USA.

[6] B.Liu (2015), Sentiment analysis: mining sentiments, opinions and emotions, Cambridge University Press, ISBN 9781107017894

[7] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean (2013), Efficient Estimation of Word Representations in Vector Space In Proceedings of Workshop at ICLR.

[8] Andrew Ng, Machine Learning course on Coursera

[9] Christopher Olah (2015), Understanding LSTM networks in Colah's blog

[10] Andrej Karpathy (2015), The Unreasonable Effectiveness of Recurrent Neural Network at Andrej Karpathy blog

[11] McCormick, C. (2016). Word2vec Tutorial - The Skip-Gram Model.

[12] Google (2013), Word2vec model https://code.google.com/archive/p/word2vec/

[13] J. McAuley and J. Leskovec (2013), From Amateurs to Connoisseurs: Modeling the Evolution of User Expertise through Online Reviews

[14] The statistic of social media usage (2014) http://thesocialskinny.com/103-crazy-social-media-statistics-to-kick-off-2014/

[15] Kishori K. Pawar, Pukhraj P Shrishrimal, R. R. Deshmukh (2015) Twitter Sentiment Analysis: A Review ISSN 2229-5518

[16] Python Programming Language https://www.python.org/

[17] Jure Leskovec, Web data Amazon Fine Foods reviews (2014) https://snap.stanford.edu/data/web-FineFoods.html

[18] TensorFlow https://www.TensorFlow.org/

[19] Scikit Learn http://scikit-learn.org/stable/