HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

TỐNG NGUYÊN SƠN

PHÁT HIỆN CÂU CHỨA GỢI Ý TRÊN DIỄN ĐÀN TRỰC TUYẾN SỬ DỤNG MẠNG NƠ - RON

LUẬN VĂN THẠC SĨ KỸ THUẬT

(*Theo định hướng ứng dụng*)

HÀ NỘI - 2020

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

TỐNG NGUYÊN SƠN

PHÁT HIỆN CÂU CHỨA GỢI Ý TRÊN DIỄN ĐÀN TRỰC TUYẾN SỬ DỤNG MẠNG NƠ-RON

Chuyên ngành: Hệ thống thông tin

Mã số: 8.48.01.04

LUẬN VĂN THẠC SĨ KỸ THUẬT

(*Theo định hướng ứng dụng*)

NGƯỜI HƯỚNG DẪN KHOA HỌC: TS. NGÔ XUÂN BÁCH

HÀ NỘI – 2020

LỜI CAM ĐOAN

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi dưới sự hướng dẫn của Tiến sĩ Ngô Xuân Bách. Các kết quả đạt được trong luận văn là sản phẩm của riêng cá nhân, không sao chép của người khác. Nội dung của luận văn có tham khảo và sử dụng một số thông tin, tài liệu từ các nguồn sách, tạp chí được liệt kê trong danh mục các tài liệu tham khảo.

Tác giả luận văn ký và ghi rõ họ tên

Tống Nguyên Sơn

LỜI CẢM ƠN

Tôi xin gửi lời cảm ơn chân thành nhất đến Thầy TS. Ngô Xuân Bách, người đã tận tình hướng dẫn, hỗ trợ và giúp đỡ tôi rất nhiều trong nghiên cứu luận văn. Thầy đã đưa ra những định hướng, nhận xét và góp ý quý giá để luận văn này được hoàn thành tốt nhất.

Kính gửi lời cảm ơn đến quý Thầy, Cô giảng viên đã tận tình giảng dạy và truyền đạt những kiến thức chuyên môn cần thiết trong quá trình tôi được học tập tại Học viện Công nghệ Bưu chính Viễn thông.

Xin gửi lời biết ơn đến gia đình đã không ngừng quan tâm, động viên, ủng hộ về mặt tinh thần lẫn vật chất trong suốt thời gian tôi tham gia khóa học và thực hiện luận văn này.

Cảm ơn các bạn lớp Cao học M18CQIS02B đã giúp đỡ và đồng hành cùng tôi trong những năm tháng học tập tại nhà trường.

Thời gian thực hiện luận văn còn khá ngắn, kinh nghiệm về lĩnh vực xử lý ngôn ngữ tự nhiên của bản thân còn hạn chế, luận văn cũng còn nhiều thiếu sót rất mong nhận được những ý kiến đóng góp của quý Thầy Cô và các bạn để tôi có thể hoàn thiện luận văn một cách tốt nhất.

Xin trân trọng cảm ơn!

MỤC LỤC

BẢNG DANH MỤC THUẬT NGỮ TIẾNG ANH

Viết tắt	Tiếng Anh	Tiếng Việt
	Social Network	Trực tuyến
	Social media phenomena	Cộng đồng mạng
	Fanpage	Trang thông tin trên trực tuyến
	Neural	Mạng nơron
	Deep neural network	Mạng nơron sâu
	Deep Learning	Là phương pháp học sâu trong AI
	Filter	Bộ lọc
	Convolutional	Tích chập (Xoắn)
NLP	Natural Language Processing	Xử lý ngôn ngữ tự nhiên
TF	Term Frequency	Tần số xuất hiện 1 từ trong 1 văn bản
N-Gram	N-Gram	Tần suất xuất hiện của n kí tự liên tiếp
IDF	Inverse Document Frequency	Tần số nghịch của 1 từ trong tập văn bản
AI	Word embedding	Từ nhúng (chuyển từ thành vector số)
NN	Neural Network	Mạng nơron nhân tạo
NLP	Natural language processing	Xử lý ngôn ngữ tự nhiên
CNN	Convolutional Neural Network	Mạng nơron tích chập
RNN	Recurrent Neural Network	Mạng nơron tái phát
GRU	Gated Recurrent Unit	Là một cơ chế gating trong mạng neural tái phát
LSTM	Long short-term memory	Là một cơ chế mạng nơron tái phát.

DANH MỤC BẢNG BIỂU

DANH MỤC HÌNH

LỜI NÓI ĐẦU

Trong thời gian qua, nhu cầu sử dụng mạng xã hội trực tuyến của người dùng không ngừng tăng lên, các trang mạng xã hội trực tuyến phổ biến như là Facebook, Twitter, Instagram, youtube, G+, blog v.v ngày càng phát triển. Con người sử dụng mạng xã hội trực tuyến không chỉ để giải trí như: cập nhật trạng thái, kết bạn, tán gẫu, nói chuyện mà họ còn dùng mạng xã hội trực tuyến như một nơi để chia sẻ thông tin, ý kiến trao đổi những nhu cầu, mong muốn, ý định hay dự định của họ trên các diễn đàn trực tuyến. Xuất phát từ thực tế đó việc phát hiện, phân loại những lời gợi ý mong muốn, ý định của người dùng sẽ mang lại giá trị thương mại, dịch vụ rất lớn.

Trong luận văn này, chúng tôi tập trung vào bài toán phát hiện câu chứa gợi ý trên các diễn đàn trực tuyến. Đây là bài toán có đầu vào là một câu được người dùng đăng lên các diễn đàn trực tuyến, câu đó có thể là những chia sẻ ,trao đổi cảm nhận, kinh nghiệm về các sản phẩm, dịch vụ, các vấn đề đời sống và mọi thứ xung quanh mà chính người dùng đó đã trải nghiệm và chúng ta cần phải xác định xem các chia sẻ, các câu đó có chứa gợi ý gì hay không? Nếu các câu có chứa gợi ý của người dùng thì gợi ý về nhu cầu, mong muốn, ý định v.v của người dùng đó về vấn đề gì như *:* du lịch, đồ ăn, thức uống, nghề nghiệp, giáo dục, hàng hóa & dịch vụ, sự kiện & hoạt động, không có ý định cụ thể. Bên cạnh đó, không phải tất cả những chia sẻ của người dùng đều thể hiện lời gợi ý rõ ràng và là nguồn dữ liệu, tài nguyên có ích. Vì vậy, luận văn sẽ tập trung chủ yếu vào phát hiện và phân loại các câu có chứa gợi ý của người dùng trên diễn đàn trực tuyến. Việc phát hiện, phân loại câu chứa gợi ý của người dùng đã và đang là đề tài nghiên cứu thời sự, mang tính cấp thiết hiện nay. Với các khách hàng, doanh nghiệp hay các nhà cung cấp dịch vụ việc biết được gợi ý, mong muốn của người dùng sẽ giúp họ cải tiến tốt hơn sản phẩm, hệ thống của mình để đảm bảo cung cấp đúng nội dung khách hàng cần, mở rộng số lượng người dùng quan tâm, quảng bá thương hiệu, hình ảnh. Hơn thế nữa, kết quả của bài toán phân loại câu chứa gợi ý người dùng có thể được ứng dụng làm đầu vào cho nhiều nghiên cứu khác như xây dựng hệ tư vấn xã hội dựa trên gợi ý người dùng, dự đoán sở thích người dùng, dự đoán xu hướng tương lai.

Luận văn "***Phát hiện câu chứa gợi ý trên diễn đàn trực tuyến sử dụng mạng***

Nơ-Ron" thực hiện khảo sát, nghiên cứu các phương pháp xây dựng hệ thống phân loại câu chứa gợi ý được quan tâm nhất hiện nay. Từ đó đưa ra phương pháp phân loại câu phù hợp nhất cho hệ thống phân loại câu bằng tiếng Anh. Dựa trên những hướng tiếp cận đã đề cập ở trên, trong luận văn này, chúng tôi tiến hành áp dụng làm thực nghiệm dựa trên sự kết hợp một số đặc trưng ngôn ngữ tiếng Anh.

Các đặc trưng này sẽ được biểu diễn dưới dạng vectơ và làm đầu vào cho các thuật toán. Sau khi thu được kết quả của mô hình phân lớp *CNN, RNN,*LSTM luận văn sử dụng phương pháp để kiểm tra và lựa chọn kết quả tốt nhất. Kết quả thực nghiệm tốt nhất đạt được khi sử dụng thuật toán LSTM. Cụ thể kết quả thực nghiệm cho kết quả tốt nhất với bài toán " ***Phát hiện câu chứa gợi ý trên diễn đàn trực tuyến sử dụng mạng Nơ-Ron"***

Nội dung của luận văn gồm 03 chương:

Chương 1: Giới thiệu bài toán phân loại câu chứa gợi ý

Nội dung của chương, tổng quan nhất về gợi ý của người dùng trên diễn đàn trực tuyến, bài toán phân loại câu chứa gợi ý trên diễn đàn trực tuyến và cuối cùng là hướng tiếp cận nhằm giải quyết bài toán đề ra.

Chương 2: Phương pháp học máy cho bài toán phân loại câu chứa gợi ý trên diễn đàn trực tuyến

Nội dung của chương là trình bày một số phương pháp trích chọn lấy đặc trưng để giải quyết bài toán, các phương pháp học máy thống kê được sử dụng để tiến hành thực nghiệm cho bài toán phân loại câu chứa gợi ý trên diễn trực tuyến sử dụng mạng Nơ Ron.

Chương 3: Thực nghiệm và đánh giá

Nội dung chương nhằm nêu rõ và chi tiết các bước trong quá trình giải quyết bài toán. Trong chương này cũng sẽ trình bày quá trình thực hiện và thực nghiệm, đưa ra một số đánh giá, nhận xét các kết quả thu được.

Phần kết luận: Tóm lược những kết quả đạt được của luận văn. Đồng thời đưa ra những hạn chế, những điểm cần khắc phục và đưa ra định hướng nghiên cứu trong thời gian sắp tới.

CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN PHÂN LOẠI CÂU CHỨA GỢI Ý

Trong chương này, luận văn trình bày giới thiệu chung về lĩnh vực xử lý ngôn ngữ tự nhiên (phần 1.1)và các ứng dụng trong thực tế (phần 1.2), cái nhìn tổng quan về bài toán phân loại câu chứa gợi ý, các cách tiếp cận bài toán, các nghiên cứu liên quan và kết quả luận văn đã đạt được.

1.1. Giới thiệu về xử lý ngôn ngữ tự nhiên

Xử lý ngôn ngữ tự nhiên (natural language processing – NLP) [3] [4] là một lĩnh vực nghiên cứu của trí tuệ nhân tạo, tập trung vào nghiên cứu các phương pháp, kỹ thuật cho phép xử lý ngôn ngữ tự nhiên bằng máy tính, từ đó xây dựng các chương trình, hệ thống máy tính xử lý ngôn ngữ của con người.

Xử lý ngôn ngữ tự nhiên được áp dụng trong nhiều bài toán và ứng dụng thực tế, trong nhiều lĩnh vực:

Nhận dạng tiếng nói: Nhận dạng tiếng nói rồi chuyển chúng thành văn bản tương ứng. Giúp thao tác của con người trên các thiết bị nhanh hơn và đơn giản hơn. Đây cũng là bước đầu tiên cần phải thực hiện trong ước mơ thực hiện giao tiếp giữa con người với robot. Nhận dạng tiếng nói có khả năng trợ giúp người khiếm thị rất nhiều.

Tổng hợp tiếng nói: Từ một văn bản tự động tổng hợp thành tiếng nói. Giống như nhận dạng tiếng nói, tổng hợp tiếng nói là sự trợ giúp tốt cho người khiếm thị, nhưng ngược lại nó là bước cuối cùng trong giao tiếp giữa robot với người.

Dịch máy (machine translate): Như tên gọi đây là chương trình dịch tự động từ ngôn ngữ này sang ngôn ngữ khác.

Tìm kiếm và truy xuất thông tin: Đặt câu hỏi và chương trình tự tìm ra nội dung phù hợp nhất. Thông tin ngày càng đầy lên theo cấp số nhân, đặc biệt với sự trợ giúp của internet việc tiếp cận thông tin trở nên dễ dàng hơn bao giờ hết. Việc khó khăn lúc này là tìm đúng nhất thông tin mình cần giữa muôn vàn tri thức và đặc biệt thông tin đó phải đáng tin cậy.

Tóm tắt văn bản: Từ một văn bản dài tóm tắt thành một văn bản ngắn hơn theo

mong muốn nhưng vẫn chứa những nội dung thiết yếu nhất.

Khai phá dữ liệu văn bản: Từ rất nhiều tài liệu khác nhau phát hiện ra tri thức mới. Thực tế để làm được điều này rất khó, nó gần như là mô phỏng quá trình học tập, khám phá khoa học của con người, đây là lĩnh vực đang trong giai đoạn đầu phát triển.

Trích chọn thông tin: là quá trình phân tích, xử lý dữ liệu để trích chọn các thông tin hữu ích, có cấu trúc từ nguồn thông tin phi cấu trúc hoặc bán cấu trúc. Thông thường quá trình này bao gồm ba bước chính là: xác định thực thể, xác định mối liên hệ và trích xuất sự kiện

Khai phá quan điểm: là lĩnh vực nghiên cứu mà cố gắng để làm cho hệ thống tự động xác định quan điểm của con người từ văn bản được viết bằng ngôn ngữ tự nhiên. Khai phá quan điểm nghiên cứu về ý kiến, tình cảm, quan niệm chủ quan, đánh giá, thái độ, thẩm định, cảm xúc… được thể hiện trong văn bản.

1.2. Bài toán phát hiện câu chứa gợi ý trên diễn đàn trực tuyến

1.2.1. Phân loại dữ liệu văn bản

Phân loại dữ liệu văn bản là quá trình phân lớp một đối tượng dữ liệu vào một hay nhiều lớp cho trước nhờ một mô hình phân lớp mà mô hình này được xây dựng dựa trên một tập hợp các đối tượng dữ liệu đã được gán nhãn từ trước gọi là tập dữ liệu học (tập huấn luyện). Quá trình phân lớp còn được gọi là quá trình gán nhãn cho các đối tượng dữ liệu.

Các bài toán phân loại dữ liệu văn bản thường thấy :

- Phân loại dữ liệu văn bản theo chủ đề văn hóa, xã hội, chính trị, thể thao,…
- Phân loại câu : phân loại quan điểm, câu chứa quan điểm (tích cực, tiêu cực)
- Phân loại câu hoặc văn bản theo cảm xúc như: vui buồn, yêu , ghét, tức, giận…
- Phân loại câu theo chức năng ngữ pháp: Câu bị động, câu chủ động, câu so sánh…
- Phát hiện ý định người dùng

Trong nội dung luận văn này sẽ tập trung vào bài toán phát hiện câu chứa gợi ý trên diễn đàn trực tuyến.

1.2.2. Phát biểu bài toán phân loại phát hiện câu chứa gợi ý

Bài toán phân loại câu, phân loại văn bản được thấy rất nhiều trong các ứng dụng NLP (xử lý ngôn ngữ tự nhiên). Bài toán phát hiện câu chứa gợi ý trên diễn đàn trực tuyến sử dụng mạng Nơ- ron nhằm khai thác gợi ý có thể được định nghĩa là trích xuất các gợi ý từ văn bản phi cấu trúc, trong đó thuật ngữ 'gợi ý' đề cập đến cách diễn đạt các mẹo, lời khuyên, khuyến nghị,…v.v. Câu chứa gợi ý là câu thể hiện các ý kiến, góp ý, mong muốn đối với con người, thương hiệu, tranh luận xã hội, các sản phẩm thương mại, dịch vụ ..v.v thường được thể hiện thông qua các đánh giá trực tuyến, blog, diễn đàn thảo luận hoặc nền tảng truyền thông xã hội và có xu hướng chứa các biểu thức hàm ý về lời khuyên, mẹo, cảnh báo, khuyến nghị, ...v.v. Ví dụ, các đánh giá trực tuyến có thể chứa các gợi ý về cải tiến sản phẩm, chất lượng sản phẩm, hoặc các dịch vụ theo ý kiến của các cá nhân đã trải nghiệm và nền tảng gợi ý thường yêu cầu lời khuyên cụ thể từ người dùng của họ, trong đó nó được cung cấp cho người dùng khác. Khai thác gợi ý vẫn còn là một lĩnh vực tương đối mới so với phân tích tình cảm, đặc biệt là trong bối cảnh những tiến bộ gần đây trong các phương pháp tiếp cận dựa trên mạng thần kinh để thể hiện tính năng học tập. Nghiên cứu khai thác gợi ý có thể thúc đẩy sự tham gia của cả các thực thể thương mại, cũng như các cộng đồng nghiên cứu làm việc về các vấn đề như khai thác ý kiến, học tập có giám sát, học tập đại diện,.. v.v. Khai thác gợi ý như là một nhiệm vụ phân loại câu, trong đó dự đoán lớp phải được thực hiện trên mỗi câu của một văn bản có ý kiến nhất định, các lớp là "gợi ý" và " không gợi ý". Vì vậy, ta cần phải nghiên cứu xây dựng một mô hình thuật toán để hiểu được ý nghĩa của câu để quyết định xem câu đó có phải là câu chứa gợi ý hay không chứa gợi ý. Trong đó ác thuật toán CNN(Mạng Nơ-ron tích chập), RNN(Mạng Nơ-ron hồi quy), LSTM(Mạng nơ-ron có bộ nhớ ngắn dài) trong mô hình mạng Nơ-ron là những thuật toán phân loại có hiệu quả và có độ chính xác cao để giải quyết bài toán. Việc Phát hiện câu chứa gợi ý trên diễn đàn trực tuyến sử dụng mạng Nơ-ron gặp nhiều khó khăn như: câu dài, nhiều từ viết tắt, sai chính tả, các ký hiệu sai và không đúng cú pháp,chất lượng, độ tin cậy thấp. Những yếu tố này làm giảm hiệu quả phát hiện câu chứa gợi ý dựa trên cách xử lý truyền thống.

Phát hiện câu chứa gợi ý là bài toán cho một câu S, dự đoán một nhãn L cho S trong đó L ∈ {có chứa gợi ý, không chứa gợi ý}. Để xử lý thì cần có Tập dữ liệu của các câu S đã được phân loại và gán nhãn L gọi là tập dữ liệu huấn luyện, sau đó sử dụng các thuật toán để huấn luyện. Bài toán được phát biểu như sau:

- ***Đầu vào***: Cho một câu trên diễn đàn trực tuyến
- ***Đầu ra***: L ∈ {có chứa gợi ý, không chứa gợi ý }.

Ví dụ:

Câu chứa gợi ý: ‘‘Hãy để thêm một bình hoa trong phòng, phòng sẽ đẹp hơn’’

Câu không chứa gợi ý: ‘‘Bình hoa này trong phòng rất đẹp”

1.2.3. Ý nghĩa bài toán:

Phát hiện câu chứa gợi ý có thể được ứng dụng trong việc thu thập nhu cầu liên quan đến suy luận ngữ nghĩa trên nhiều ứng dụng của ngôn ngữ tự nhiên như: các ý kiến của người tiêu dùng đối với các thực thể thương mại như thương hiệu, dịch vụ và sản phẩm thường được thể hiện thông qua đánh giá trực tuyến, blog, diễn đàn trực tuyến hoặc nền tảng truyền thông xã hội. Trên thế giới cũng đã có nhiều nghiên cứu về phát hiện câu chứa gợi ý sử dụng mô hình mạng Nơ-ron, Ví dụ như tại Semeval2019Task9/Subtask-A [5], đã có nhóm nghiên cứu đăng ký tham gia vào nhiệm vụ cùng nhiều nhà nghiên cứu khác thực hiện bên ngoài. Tuy nhiên tại Việt Nam chưa có nhiều dự án được nghiên cứu, triển khai và áp dụng vào trong thực tế.

1.3. Các nghiên cứu liên quan

Trong những năm gần đây, trên thế giới cũng đã có nhiều nghiên cứu về phát hiện câu chứa gợi ý sử dụng mô hình mạng Nơ-ron, Ví dụ như tại Semeval2019Task9/Subtask-A [5], đã có nhóm nghiên cứu đăng ký tham gia vào nhiệm vụ cùng nhiều nhà nghiên cứu khác thực hiện bên ngoài. Tuy nhiên tại Việt Nam chưa có nhiều dự án được nghiên cứu, triển khai và áp dụng vào trong thực tế.

Ngoài ra, tác giả Ahmed Husseini Orabi cùng cộng sự đã thực hiện một đề tài rất thiết thực và có ý nghĩa về việc sử dụng học sâu để phát hiện trầm cảm của người dùng Twitter: *“Học sâu để phát hiện trầm cảm của người dùng twitter”* [7]. Công trình trình bày việc xử lý ngôn ngữ tự nhiên trên trực tuyến twitter, thực hiện đánh giá và

so sánh trên một số mô hình học sâu, cụ thể là 3 mô hình CNN và 1 mô hình RNN và đưa ra kết quả về vấn đề rối loạn tâm thần và làm tiền đề cho hệ thống phát hiện các hành vi, cảm xúc tiêu cực của người dùng cá nhân trên trực tuyến.

"Phân tích ý định của người dùng trong văn bản ngôn ngữ tự nhiên"[13] của tác giả Kröll, M., & Strohmaier, M. (2009, September) đăng trên hội nghị kỷ yếu lần thứ 15 về vấn đề thu thập tri thức (pp. 197-198) tập trung vào việc thực hiện việc phân tích, phân loại ý định cụ thể của người dùng.

Bên cạnh đó đã có rất nhiều công trình nghiên cứu của các tác giả [8], [9], [10], [11], [12] liên quan đến việc khai phá quan điểm, phân tích ý định từ nhiều nguồn dữ liệu với các phương pháp khác nhau như sử dụng phương pháp SVM, sử dụng mô hình mạng nơron hồi quy, mô hình mạng nơron tích chập,… với kết quả rất khả quan và hứa hẹn sẽ phát triển và bùng nổ trong những năm tới.

Luận văn sẽ tham khảo, tìm hiểu và giới thiệu về các phương pháp phổ biến, sau đó sẽ áp dụng và đưa ra kết quả đánh giá cũng như đề xuất giải pháp để phát hiện câu chứa gợi ý trên diễn đàn trực tuyến. Những đóng góp ban đầu của luận văn như: xử lý tiền dữ liệu, phân lớp dữ liệu trên các phương pháp khác nhau sẽ làm cơ sở ban đầu trong việc đánh giá và lựa chọn các phương pháp, mô hình học máy sao cho phù hợp, làm tiền đề cho các ứng dụng tự động, phân tích sử dụng dữ liệu sau này.

1.4. Kết luận chương

Trong chương 1, luận văn đã giới thiệu tổng quan về bài toán xử lý ngôn ngữ tự nhiên. Tìm hiểu bài toán phân loại câu, văn bản và giới thiệu bài toán phát hiện câu chứa gợi ý trên diễn đàn trực tuyến, từ đó đưa ra những vấn đề cần làm rõ và giải quyết trong luận văn.

Trong chương 2, luận văn sẽ trình bày về hướng giải quyết cho bài toán phát hiện câu chứa gợi ý và đi sâu hơn trình bày về các phương pháp sẽ áp dụng để giải quyết bài toán.

CHƯƠNG 2: PHƯƠNG PHÁP PHÁT HIỆN CÂU CHỨA GỢI Ý SỬ DỤNG HỌC MÁY

Trong chương 2, luận văn tập trung trình bày một số phương pháp giải quyết bài toán (phần 2.1) và các thuật toán mô hình mạng Nơron được sử dụng khi làm thực nghiệm : CNN,RNN và LSTM (phần 2.2)

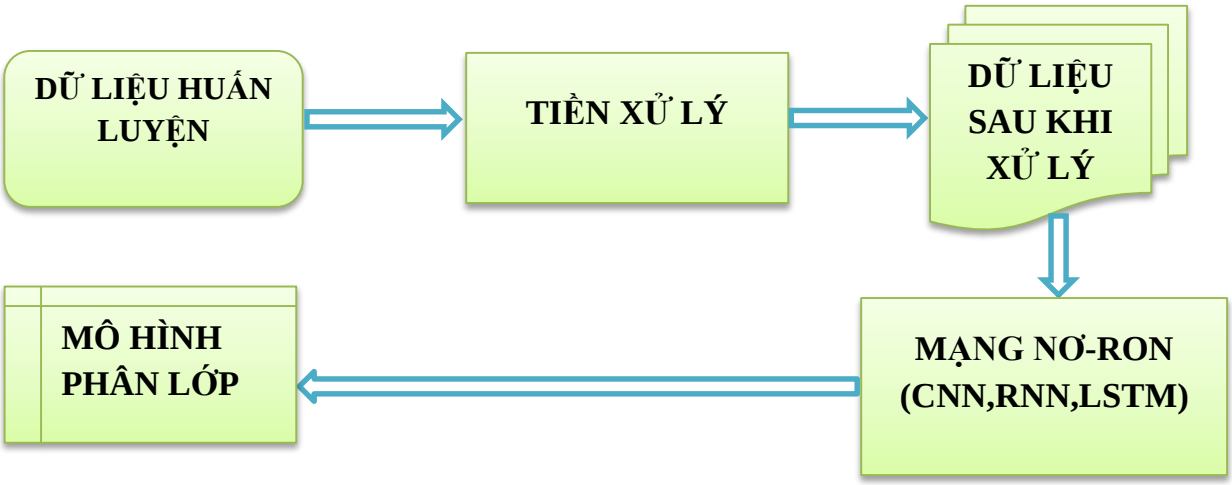
2.1. Phương pháp giải quyết bài toán:

Qua quá trình nghiên cứu, tập hợp các dòng trạng thái trên các diễn đàn trực tuyến và đã thu thập được tập các nội dung chia sẻ về những vấn đề xung quanh của người dùng qua bộ dữ liệu Semeval2019Task9/Subtask-A bao gồm: khoảng 833 câu [5], mục đích sẽ xác định nội dung các câu đó là câu có chứa gợi ý hay câu không chứa gợi ý. Luận văn đã tham khảo và tìm hiểu sau đó đưa ra được các bước thực hiện để xây dựng phương pháp giải quyết cho bài toán phát hiện câu chứa gợi ý được chia làm 2 giai đoạn sau:

- Giai đoạn huấn luyện
- Giai đoạn phân lớp.

a, Giai đoạn huấn luyện:

Giai đoạn này nhận đầu vào là tập dữ liệu huấn luyện gồm các nội dung dưới dạng văn bản đã được gán nhãn, sau khi xử lý tập dữ liệu và áp dụng các thuật toán huấn luyện sẽ cho đầu ra là một mô hình phân loại, cụ thể:



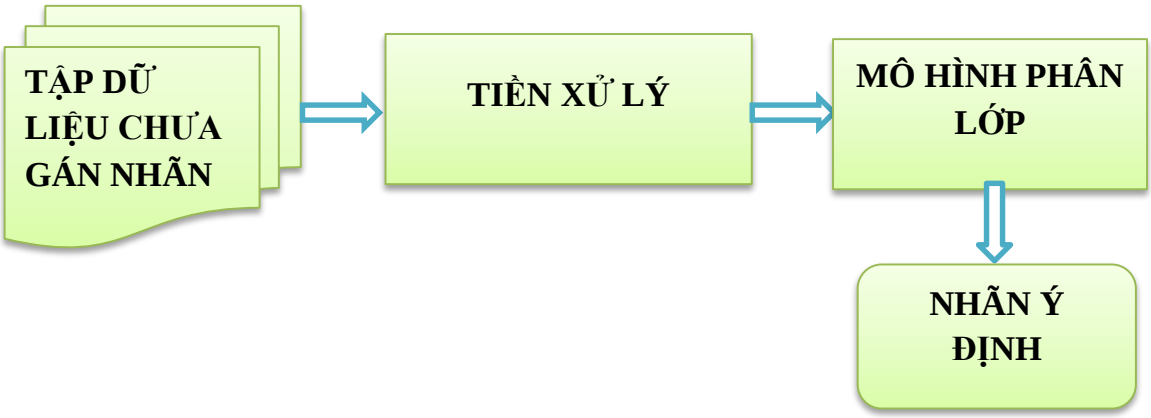
Hình 2.1 Mô hình giai đoạn huấn luyện

Trong đó các bước cụ thể sau :

- Tiền xử lý: Chuyển đổi các dòng trạng thái trong tập dữ liệu thành một hình thức phù hợp để phân loại.
- Dữ liệu sau khi xử lý: Tập dữ liệu đã được xử lý ở bước tiền xử lý như : lọc nhiễu, loại bỏ các thông tin dư thừa…
- Thuật toán huấn luyện: Thủ tục huấn luyện để tìm ra các tham số tối ưu, có thể sử dụng các thuật toán khác nhau, trong phạm vi luận văn chúng tôi sử dụng thuật toán học máy gồm: Mạng Nơ-Ron tích chập (CNN), mạng Nơ-ron hồi qui (RNN), Mạng Nơ-Ron có bộ nhớ ngắn dài (LSTM)

b, Giai đoạn phân lớp :

Nhận đầu vào là nội dung trạng thái của người dùng dưới dạng ngôn ngữ tự nhiên, sau quá trình xử lý và áp dụng mô hình phân loại sẽ cho ra nhãn phân loại của câu dữ liệu văn bản đầu vào, cụ thể được biểu diễn dưới sơ đồ sau:



Hình 2.2: Mô hình giai đoạn phân lớp

Tương tự như các bước trong giai đoạn huấn luyện, giai đoạn phân lớp có nhiệm vụ cụ thể:

- *Tiền xử lý:* Chuyển đổi các dòng trạng thái trong tập dữ liệu thành một hình thức phù hợp để phân loại như lọc nhiễu, loại bỏ các từ không mang ý định.
- *Mô hình phân lớp:* Sử dụng các thuật toán như Mạng Nơ-Ron tích chập (CNN), mạng Nơ-ron hồi qui (RNN), Mạng Nơ-Ron có bộ nhớ ngắn dài (LSTM) để tiến hành phân loại và gán nhãn ý định.

Dựa vào sơ đồ 2.1 và 2. 2 trên ta có thể dễ dàng nhận thấy:

Mô hình kiến trúc hệ thống tổng quát cho bài toán phân loại câu chứa gợi ý gồm các bước chính. Sau đây sẽ giới thiệu chi tiết các thành phần quan trọng của bài toán phân loại câu chứa gợi ý nói riêng và bài toán phân loại văn bản nói chung cho tập dữ liệu thu thập được trên trang trực tuyến gồm 833 dòng trạng thái [5] và được lưu trữ trong file dữ liệu.

2.1.1. Tiền xử lý dữ liệu

Tiền xử lý dữ liệu là một bước rất quan trọng trong quá trình phân loại dữ liệu.Các kỹ thuật tiền xử lý dữ liệu phổ biến hiện nay bao gồm: xử lý dữ liệu bị khuyết (missing data), mã hóa các biến nhóm (encoding categorical variables), chuẩn hóa dữ liệu (standardizing data), co giãn dữ liệu (scaling data),v.v. Một số lỗi thường mắc phải trong khi thu thập dữ liệu là tính không đủ chặt chẽ, logic. Vì vậy, dữ liệu chứa các giá trị vô nghĩa và không có khả năng kết nối dữ liệu, ví dụ dữ liệu là các con số, các ký tự đặc biệt, các #hastag. Ở bước này chúng tôi sẽ tiến hành xử lý những dạng dữ liệu không chặt chẽ nói trên, những dữ liệu dạng này được xem như thông tin dư thừa, không có giá trị. Bởi vậy, đây là một quá trình rất quan trọng vì dữ liệu này nếu không được "làm sạch" sẽ gây nên những kết quả sai lệch nghiêm trọng.

Trước khi tiến hành xây dựng dữ liệu thực nghiệm, chúng tôi sẽ tiến hành lọc và loại bỏ một số dữ liệu không cần thiết từ tập dữ liệu đã thu thập từ các diễn đàn trực tuyến.

2.1.2. Lọc nhiễu (loại bỏ từ không mang nghĩa)

Các từ không có nghĩa ở đây là các con số, các ký tự đặc biệt và không mang nghĩa. Ví dụ: "@@", "!! "EU !!!!!!!!!!!!!!!!!!!!!!!!!!!!!, #2@",...

2.1.3. Loại bỏ các từ phổ biến (stop word):

Ngôn ngữ cũng giống như một đống gạo bị lẫn với thóc. Việc cần làm đó chính là chọn ra các hạt gạo chất lượng tốt nhất từ đống thóc đó. Những hạt thóc đó được gọi là stop words tức là những từ không có ý nghĩa lắm đối với việc phân loại. Để tiết kiệm không gian lưu trữ và gia tăng tốc độ xử lý, sẽ không ghi nhận lại những từ quá phổ biến, quá chung chung và những từ này gọi là stop word [18]

{'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 2.3: Một số stopword trong tiếng Anh [18]

Phần tiếp theo sẽ trình bày các mô hình mạng Nơ-ron được sử dụng trong luận văn.

2.2. Giới thiệu chung mô hình mạng Nơ-ron:

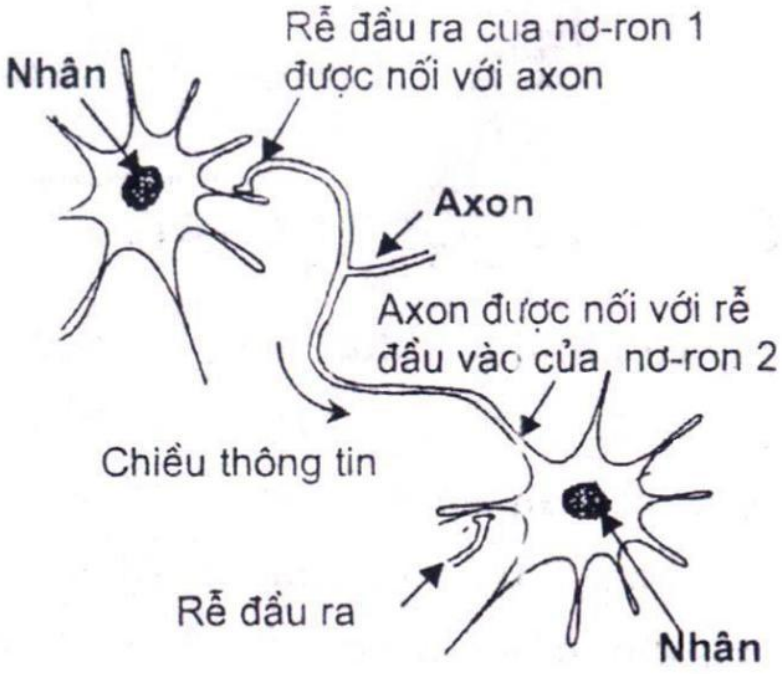
2.2.1. Mạng Nơ-ron nhân tạo (ANN)

Mạng neural nhân tạo (Artificial Neural Network- ANN)[1] là mô hình xử lý thông tin được mô phỏng dựa trên hoạt động của hệ thống thần kinh của sinh vật, bao gồm số lượng lớn các Neuron được gắn kết để xử lý thông tin. ANN giống như bộ não con người, được học bởi kinh nghiệm (thông qua huấn luyện), có khả năng lưu giữ những kinh nghiệm hiểu biết (tri thức) và sử dụng những tri thức đó trong việc dự đoán phân loại các dữ liệu chưa biết (unseen data. Mạng neural nhân tạo đã được sử dụng rộng rãi từ những năm 1980 cho đến nay, vẫn được áp dụng rộng rãi trong nhiều ngành khoa học.Mốt số kiến trúc mạng Nơ ron phổ biến như: Mạng nơ ron tích chập(CNN), mạng nơ ron hồi qui(RNN), mạng nơ ron sâu(DNN), mạng bộ nhớ ngắn dài(LSTM),….

2.2.2. 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 2.4 mô tả thông tin được truyền từ nơ-ron 1 qua axon đến nơ-ron 2



Hình 2.4: Mô hình mạng nơ ron sinh học[24]

(Nguồn: https://cs231n.github.io/)

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.

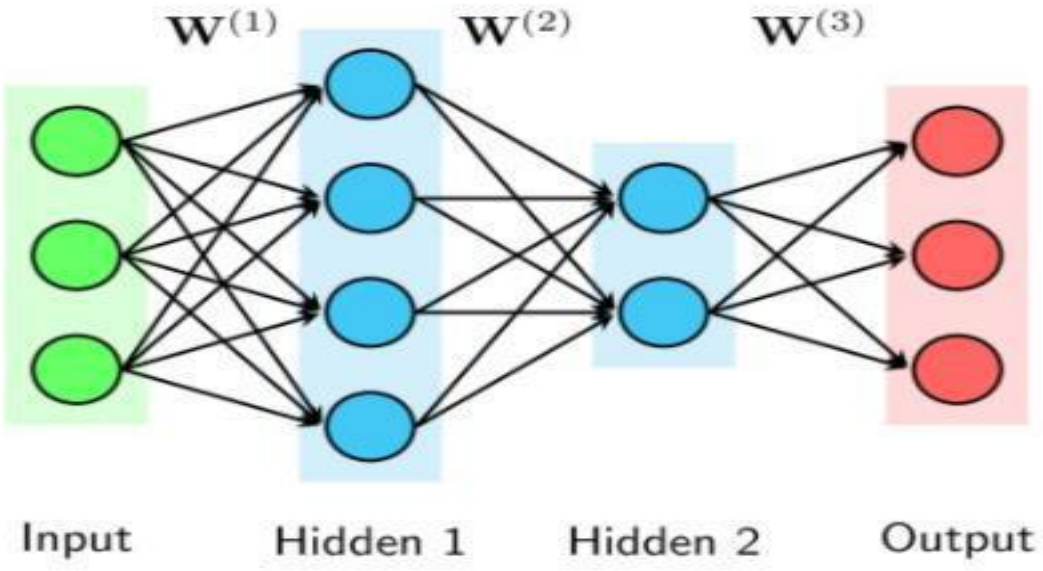
Các nhà khoa học đã tìm hiểu và lấy nguyên lý cấu trúc của mạng nơ-ron sinh học để xây dựng thành mô hình mạng neural nhân tạo.

2.2.3. 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 cách thức xử lý thông tin của hệ thống nơ-ron sinh học[1][24]. 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 gây lỗi là nhỏ nhất. Một mạng neural nhân tạo sẽ có 3 kiểu tầng:

- **Tầng vào** (*input layer*): Là tầng bên trái cùng của mạng thể hiện cho các đầu vào của mạng.
- **Tầng ra** (*output layer*): Là tầng bên phải cùng của mạng thể hiện cho các đầu ra của mạng.
- **Tầng ẩn** (*hidden layer*): Là tầng nằm giữa tầng vào và tầng ra thể hiện cho việc suy luận logic của mạng.

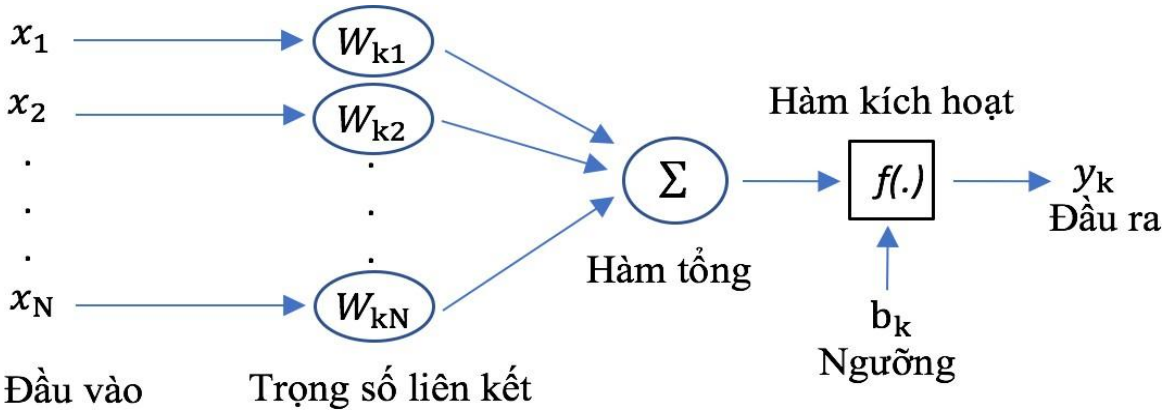
Một mạng neural nhân tạo chỉ có 1 tầng vào và 1 tầng ra nhưng có thể có nhiều tầng ẩn. Mỗi nơ-ron sẽ nhận tất cả đầu vào từ các nơ-ron ở tầng trước đó và sử dụng một hàm kích hoạt dạng (*activation function*) phi tuyến như *sigmoid, ReLU, tanh* để tính toán đầu ra. Hình 2.5 là ví dụ về một ANN có 2 lớp ẩn



Hình 2.5: Mạng neural 2 lớp ẩn[24]

(Nguồn: https://cs231n.github.io/)

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 2.6: Mô hình cấu tạo một neural

(Nguồn: https://tiendv.wordpress.com/2016/11/19/neural-networks/)

Trong đó, mỗi đầu vào 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 đầu vào là một thuộc tính của khách hàng như thu nhập, nghề nghiệp, tuổi, số conv.v. Đầu ra 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:

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

Hình 2.7 : Công thức tính hàm tổng của một Nơ-ron

Kết quả trên hình 2.7 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 hàm chuyển đổi (transfer function) được sử dụng để xử lý đầu ra 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.

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

Hinh 2.8 : Công thức tính hàm chuyển đổi

Kết quả hình 2.8 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. Mạng nơron nhân tạo đã được sử dụng để giải quyết nhiều bài toán thuộc nhiều lĩnh vực của các ngành khác nhau. Các nhóm ứng dụng mà mạng nơ-ron nhân tạo đã được áp dụng rất có hiệu quả là:

- Bài toán phân lớp (classification): Phân loại các đối tượng quan thành thành các nhóm. Ví dụ: phân loại chữ viết, nhận diện hình ảnh ...
- Bài toán dự đoán (preditive): Mạng nơ-ron nhân tạo đã được ứng dụng thành công trong việc xây dựng các mô hình dự báo sử dụng tập dữ liệu trong quá khứ để dự đoán số liệu trong tương lai. Ví dụ: dự báo thiên tai, dự báo chứng khoán ...
- Bài toán điều khiển và tối ưu hoá: Nhờ khả năng học và xấp xỉ hàm mà mạng nơ-ron nhân tạo đã được sử dụng trong nhiều hệ thống điều khiển tự động cũng như góp phần giải quyết những bài toán tối ưu trong thực tế.

2.3. Mạng nơron tích chập CNN:

CNN- Mạng Nơ-ron tích chập ra đời nhằm khắc phục các nhược điểm của Deep neural network do các mạng lưới đào tạo ngày càng phức tạp. Mạng Nơ-ron Tích Chập được giới thiệu bởi Bengio, Le Cun, Bottou và Haffner vào năm 1998. Mạng nơron tích chập [19] là một trong những mạng truyền thẳng đặc biệt, một mô hình học sâu phổ biến và tiên tiến nhất hiện nay. Hầu hết các hệ thống nhận diện và xử lý ảnh hiện nay đều sử dụng mạng nơron tích chập vì tốc độ xử lý nhanh và độ chính xác cao. Trong mạng nơron truyền thống, các tầng được coi là một chiều, thì trong mạng nơron tích chập, các tầng được coi là 3 chiều, gồm: chiều cao, chiều rộng và chiều sâu. Mạng nơron tích chập có hai khái niệm quan trọng: kết nối cục bộ và chia sẻ tham số. Những khái niệm này góp phần giảm số lượng trọng số cần được huấn luyện, do đó tăng nhanh được tốc độ tính toán.

Mạng nơron tích chập gồm có 3 tầng chính:

1.Tầng tích chập

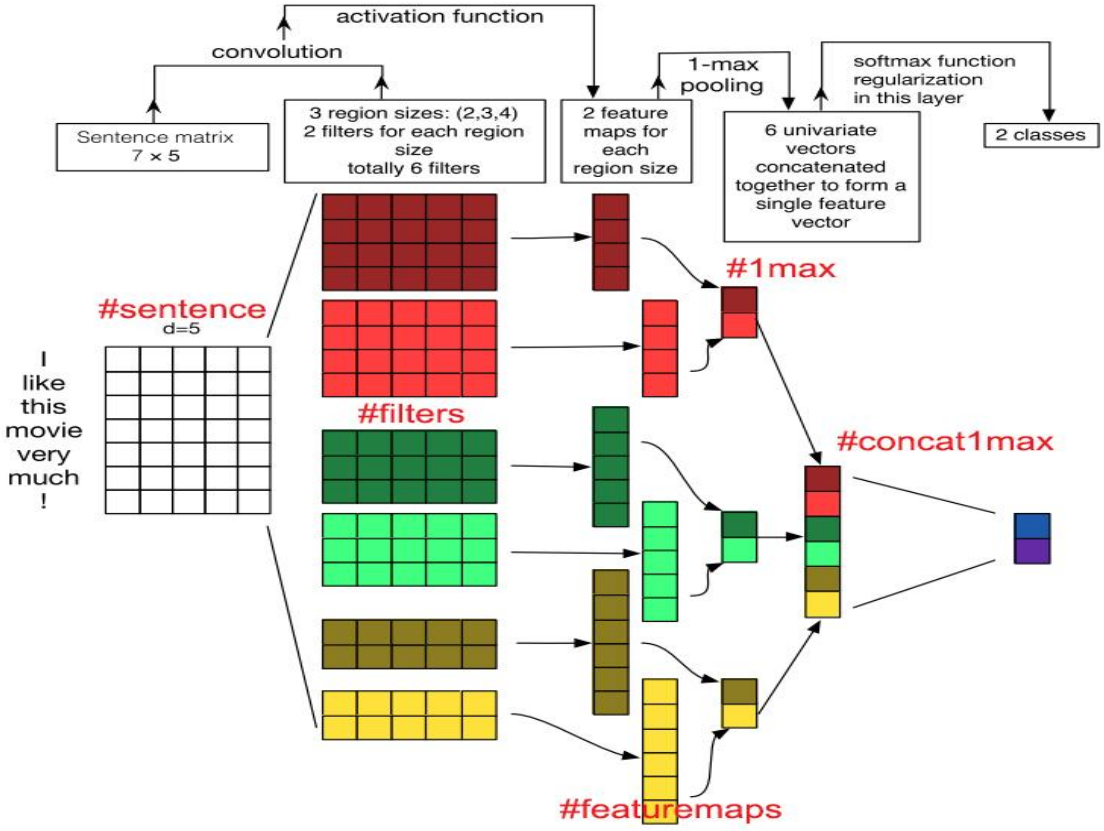
2.Tầng gộp (pooling layer)

3.Tầng được kết nối đầy đủ (fully-connected).

Tầng kết nối đầy đủ giống như các mạng nơron thông thường, và tầng tích chập thực hiện tích chập nhiều lần trên tầng trước. Tầng gộp có thể làm giảm kích thước mẫu trên từng khối 2x2 của tầng trước đó. Ở các mạng nơron tích chập, kiến trúc mạng thường chồng ba tầng này để xây dựng kiến trúc đầy đủ. Thuật toán CNN đã cho thấy sự thành công trong một số bài toán phân loại văn bản.

Trong xử lý ngôn ngữ tự nhiên (NLP), được biến đổi các câu hay văn bản thành một ma trận đầu vào. Mỗi dòng của ma trận tương ứng với một token (một từ trong câu, nhưng cũng có thể là một ký tự– character). Nghĩa là, mỗi dòng là một vector đại diện cho một từ. Thông thường, những vector này là word embedding (word2vec hay GloVe), nhưng chúng cũng có thể là one-hot vector (một cách đánh chỉ số sự xuất hiện của từ này trong dữ liệu từ điển – vocabulary).

Sau quá trình tìm hiểu và tham khảo, với điều kiện thiết bị thực nghiệm còn hạn chế, với kiến trúc CNN, luận văn quyết định áp dụng mô hình như ví dụ sau:



Hình 2.9: Mô hình thuật toán CNN [15]

(Nguồn: Zhang, Y., & Wallace, B. (2015). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification.)

Tại mô hình Convolutional Neural Network [15][27] này được sử dụng 3 bộ lọc có các kích thước khác nhau: 2, 3, 4, và mỗi một kích thước lại có 2 bộ lọc khác nhau. Các bộ lọc thực hiện nhân tích chập(convolution) lên ma trận của câu văn đầu vào và mỗi bộ lọc tạo ra một map lưu trữ các đặc trưng(features map). 6 map đặc trưng này từng map sẽ đi qua 1-max pooling – Tức là giá trị lớn nhất trong mỗi map đặc trưng sẽ được lưu lại.

Do vậy, một vector có 1 phần tử được tạo ra ở mỗi map đặc trưng. Sau đó, các giá trị này được nối lại với nhau tạo nên lớp áp chót ở trong hình. Và cuối cùng, kết quả này đi qua một hàm softmax và nhận được là một vector đặc trưng và dùng nó để dự đoán nhãn cho văn bản. Trong ví dụ này chúng tôi giả sử đây là bài toán phân loại văn bản chỉ có 2 lớp(*binary classification*) nên ở đây đầu ra chỉ có 2 trạng thái.

Ví dụ trên sử dụng câu "I like this movie very much!". Câu văn này có 6 từ và một dấu câu nữa. Dấu câu này ở đây sẽ được coi như là một từ. Như vậy là sẽ có 7 từ tất cả. Ở đây tôi chọn số chiều của word vector là 5(tức là mỗi từ sẽ là một vector kích thước 1×5). Giả sử ta gọi d là số chiều của word vector. Như vậy kích thước ma trận của cả câu văn này là 7 x 5.

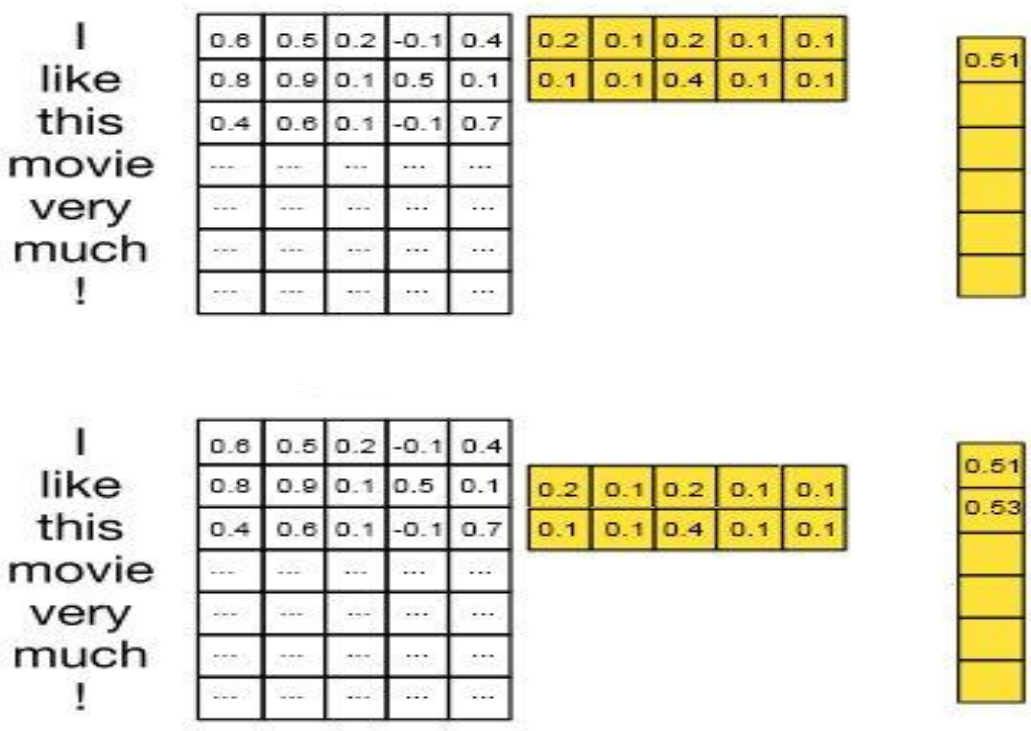
Giải thích thêm: Từ trong trường hợp này chính là một đặc trưng. Tùy bài toán mà người ta có coi dấu câu là đặc trưng hay không. Điều đó phụ thuộc vào việc nó có hữu dụng không. Tức là nếu lấy dấu câu thì có giúp tăng độ chính xác hay không. Do vậy, có nhiều trường hợp dấu câu sẽ bị loại bỏ.

Một tính chất mong muốn của thuật toán CNN bên xử lý ảnh là giữ được tính không gian 2 chiều theo góc quan sát của máy tính. Văn bản cũng có tính chất này như ảnh. Nhưng thay vì 2 chiều, văn bản chỉ có một chiều và đó là các chuỗi từ liên tiếp. Ở ví dụ trên mỗi từ lại là một vector 5 chiều, do vậy ta cần cố định số chiều của bộ lọc cho phù hợp với số chiều của từ. Như vậy bộ lọc của chúng ta nên có kích thước (? x 5). Dấu ? thể hiện số từ(số hàng) mà chúng ta muốn lấy vào.

Hình 2.9 trên, #filters là minh họa cho các bộ lọc. Đây không phải là bộ lọc để

lọc bỏ các phần tử khỏi ma trận bị lọc. Điều này sẽ được giải thích kỹ hơn ở đoạn tiếp theo. Ở đây, tác giả sử dụng 6 bộ lọc, các bộ lọc được sử dụng có kích thước (2, 3, 4) từ.

Trong đoạn này, chúng tôi sẽ từng bước giải thích cách bộ lọc nhân tích chập(convolutions / filtering).



Hình 2.10: Cách nhân tích chập giữa ma trận input với bộ lọc[27]

Ảnh phía trên thực hiện việc tính toán cho bộ lọc có kích thước là 2(2 từ). Với giá trị đầu tiên, bộ lọc màu vàng kích thước 2 x 5 thực hiện nhân từng thành phần với 2 hàng đầu tiên của văn bản(I, like).Nó thực hiện bằng cách:

$$0.51 = 0.6 \times 0.2 + 0.5 \times 0.1 + 0.2 \times 0.2 + ... + 0.1 \times 0.1.$$

Tiếp theo với giá trị thứ 2, bộ lọc màu vàng giữ nguyên nhưng lần này nhân tích chập với ma trận văn bản của 2 từ (like, this) theo cách tương tự:

$$0.53 = 0.8 \times 0.2 + 0.9 \times 0.1 + ... + 0.7 \times 0.1.$$

Cứ như vậy, ma trận bộ lọc(màu vàng) sẽ lùi xuống một dòng cho đến khi hết ma trận văn bản. Như vậy ma trận kết quả sẽ có kích thước là:

$$(7-2 + 1 \times 1) = (6 \times 1).$$

Để bảo đảm giá trị của map đặc trưng, chúng ta cần sử dụng một activation function(vd. ReLU). Áp dụng ReLU vẫn cho chúng ta ma trận có kích thước là 6 x1.

Lưu ý rằng kích thước của map đặc trưng phụ thuộc vào ma trận văn bản và ma trận bộ lọc. Nói cách khác, map đặc trưng sẽ có kích thước thay đổi chứ không cố định. Để đưa ma trận đặc trưng này về kích thước như nhau. Hoặc trong nhiều trường hợp người ta chỉ muốn giữ lại các đặc trưng tiêu biểu**.** Chúng ta có thể sử dụng max-pooling để lấy ra các giá trị lớn nhất trong map đặc trưng. Điều này giúp giảm chiều dữ liệu, tăng tốc độ tính toán.

Trong ví dụ này, tôi sử dụng 1-max-pooling. Tức là lấy ra 1 giá trị lớn nhất trong từng map đặc trưng. Việc này giúp ta có ma trận output có cùng kích thước. Ở đây kích thước sau khi áp dụng #1max là (1×1). Như vậy, ta chỉ lấy 1 đặc trưng trội nhất ở tất cả các lớp cnn để phục vụ cho bài toán.

Sau khi áp dụng 1-max pooling, chúng ta đã có những vector có kích thước cố định là 1×1 của 6 thành phần(bằng số bộ lọc). Vector cố định kích thước này sau đó được đưa vào một hàm softmax(lớp fully-connected) để giải quyết việc phân loại.

Như vậy, #concat1max thực hiện việc kết hợp tất cả các đặc trưng ở bước trước lại. Đưa ra một vector đặc trưng cuối cùng để đưa vào fully-connected. Tôi xin không giải thích kỹ về fully-connnected trong bài này. Sau đó, thường chúng ta sẽ qua một lớp softmax để đưa output về xác suất các nhãn có tổng bằng 1.

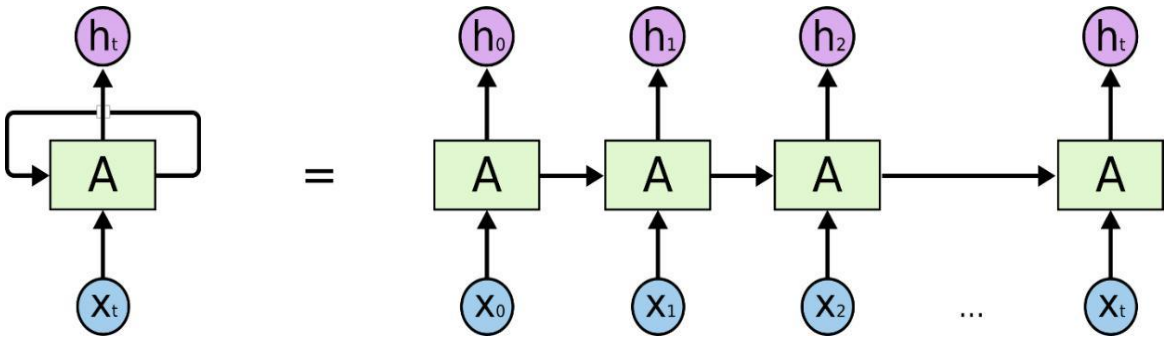
Hơn nữa, các độ đo lỗi ở giai đoạn phân loại này sau đó sẽ được đưa lại vào các tham số đóng vai trò là một phần của quá trình học để cải thiện độ chính xác cho mô hình.

2.4. Mạng nơron hồi quy RNN:

a. Giới thiệu mạng nơron hồi quy RNN:

Mạng nơ-ron hồi quy RNN là một trong những mô hình Deep Learning trong hệ thống trí tuệ nhân tạo. RNN ra đời với ý tưởng chính là sử dụng một bộ nhớ để lưu lại thông tin từ những bước tính toán xử lý trước để dựa vào nó có thể đưa ra dự đoán chính xác nhất cho bước dự đoán hiện tại. Cơ bản thì nó là một mạng neural hồi quy là một mạng neural chứa một vòng lặp bên trong nó. Mạng nơron hồi quy RNN

[20][21] được đưa ra 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 2.11: Mô hình mạng RNN không kiểm soát[21]

Với Hình 2.11 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.. Sử dụng mạng RNN có rất nhiều ứng dụng như nhận dạng giọng nói, mô hình hóa ngôn ngữ, dịch, nhận dạng ảnh.

Tuy nhiên, mạng RNN có vấn đề lưu trữ thông tin ngữ cảnh phụ thuộc lâu dài. Xét 2 ví dụ sau đây:

Ví dụ 1: Trên đường phố rất nhiều cây xanh.

Ví dụ 2: Tôi lớn lên ở Hà Tĩnh, tôi có thể nhớ hết các con phố tại Hà Tĩnh.

Với ví dụ 1, ta không cần thông tin ngữ cảnh, nhưng trong ví dụ 2 các thông tin phía trước đó gợi ý rằng từ tiếp theo có thể liên quan đến tên của một thành phố. Trong ví dụ 2, khoảng cách giữa 2 phụ thuộc này là lớn hơn. Để đưa ra dự đoán này, bắt buộc mạng RNN phải lưu trữ toàn bộ các từ vào trong bộ nhớ. Trong phạm vi khoảng cách phụ thuộc này thấp thì có thể khả thi, nhưng nếu với khoảng cách cực lớn, đoạn văn dài thì việc 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.

Vector trạng thái ẩn ht 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 2.12: Công thức tính vector trạng thái ẩn tại thời điểm t

Hình 2.12, W^H và W^X trong công thức là hai ma trận trọng số. Ma trận W^X được sử dụng để nhân với vector đầu vào xt 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 xt tại thời điểm hiện tại và giá trị của vector ẩn ht-1 của trạng thái t-1 trước đó. Vậy giá trị ht 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 ht sẽ bị ảnh hưởng nhiều hơn bởi ht-1 mà không mấy bị ảnh hưởng bởi xt . Nói một cách khác, vector trạng thái ẩn ht thấy rằng từ xt 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, ht sẽ có giá trị xấp xỉ so với ht-1

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 2.13: Hàm softmax

Hàm softmax hình 2.13 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.

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

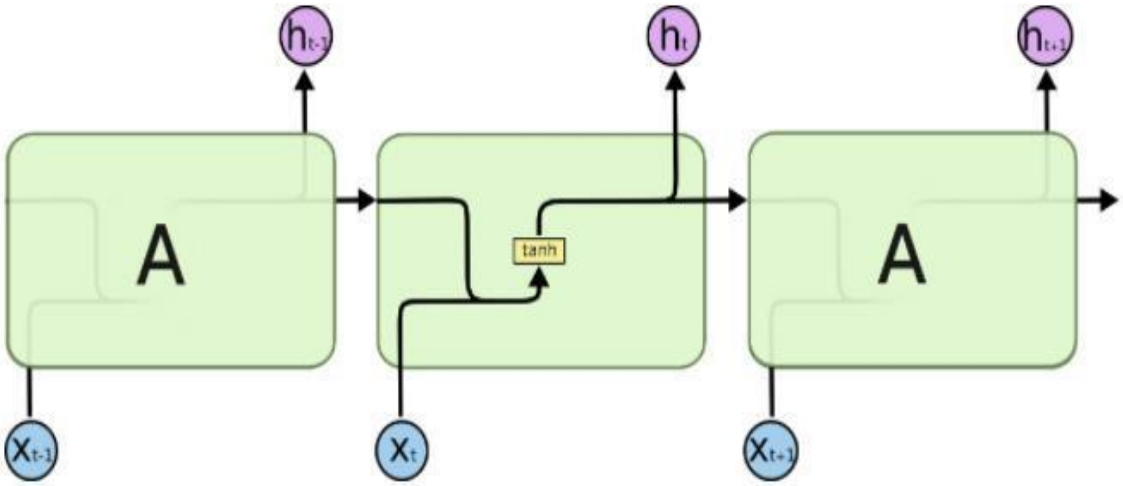
Xét một câu hỏi sau: " Số thứ nhất bằng 1. Chiếc xe đang chạy trên đường. Số thứ hai bằng 3. Tổng của hai số bằng mấy?". Ở mức độ lưu trữ thông tin cao, mạng RNN có thể lưu trữ toàn bộ các thông tin của 4 câu kể trên. Sau đó, RNN xác định ngữ cảnh câu hỏi cũng như giá trị của số thứ nhất và số thứ hai. tôi thấy rằng câu "Chiếc xe đang chạy trên đường" 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.

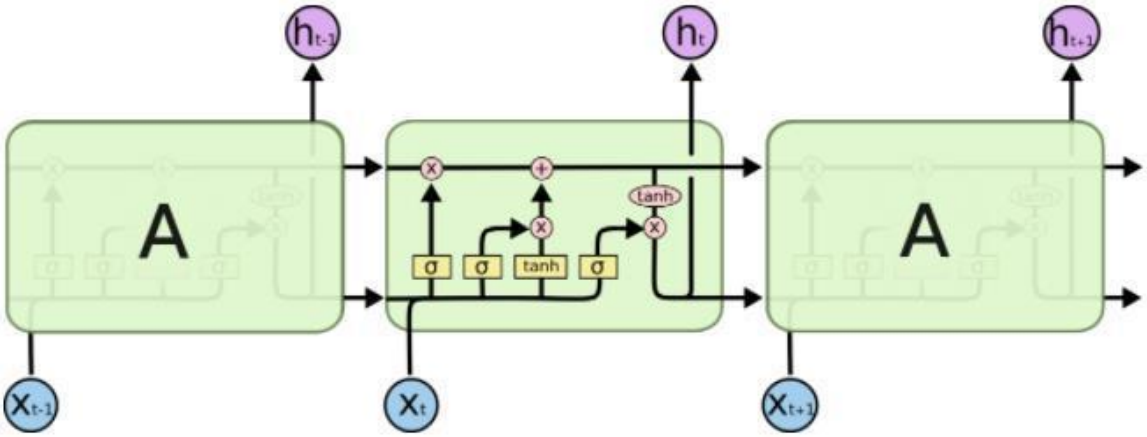
2.5. Mạng nơ-ron có bộ nhớ ngắn dài LSTM:

Mạng nơ-ron có bộ nhớ ngắn dài LSTM [21] là một loại RNN đặc biệt nó được cải tiến của mạng RNN nhằm giải quyết vấn đề học, lưu trữ thông tin ngữ cảnh có khả năng học các phụ thuộc dài. Với mô hình RNN, tại thời điểm t thì giá trị của vector ẩn ht chỉ được tính bằng một hàm tanh nói cách khác trong RNN tiêu chuẩn module lặp lại này có cấu trúc đơn giản với một lớp tanh duy nhất.



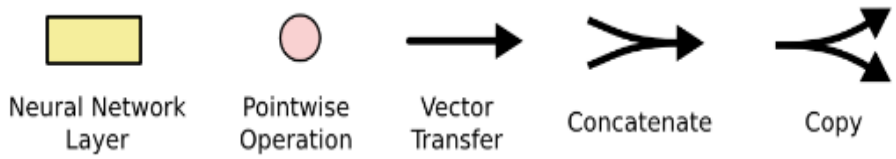
Hình 2.14: Module xử lý tính ht của RNN [21]

Các LSTM cũng có cấu trúc mắt xích giống như chuỗi này, nhưng các module lặp có cấu trúc khác hẳn. Thay vì chỉ có một layer mạng nơ ron, thì LSTM có tới bốn layer, tương tác với nhau theo một cấu trúc cụ thể rất đặc biệt.



Hình 2.15: Module lặp lại của mạng LSTM chứa 4 lớp tương tác[21]

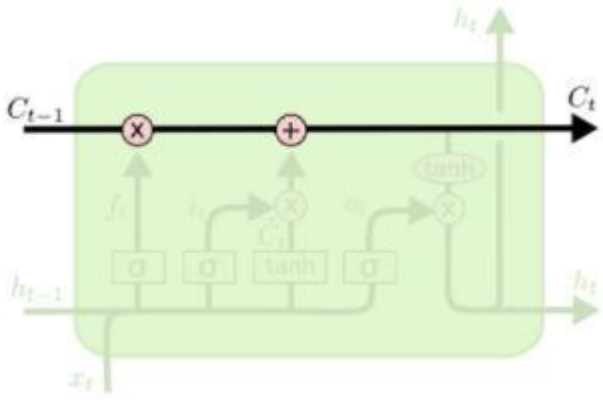
Các ký hiệu sử dụng trong mạng LSTM gồm có:



- Hình chữ nhật là các lớp ẩn của mạng nơ-ron
- Hình tròn biểu diễn toán tử Pointwise

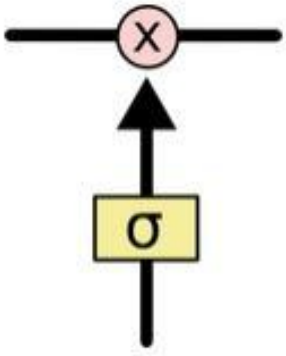
- Đường kẻ gộp lại với nhau biểu thị phép nối các toán hạng
- Đường rẽ nhánh biểu thị cho sự sao chép từ vị trí này sang vị trí khác.

Trạng thái nhớ Cell state là quan trọng với LSTM, đường kẻ ngang chạy dọc ở trên cùng của hình 2.13. Cell state chạy xuyên thẳng suốt toàn bộ mắt xích, giống như băng chuyền, chỉ một vài tương tác nhỏ tuyến tính (minor linear interaction) được thực hiện, rất dễ dàng để thông tin chỉ chạy dọc theo. Do vậy giúp cho thông tin ít bị thay đổi xuyên suốt trong quá trình lan truyền.



Hình 2.16: Cell state trong LSTM giống như một băng chuyền[21]

LSTM có khả năng loại bỏ hoặc thêm bớt thông tin vào cell state, được điều chỉnh quy định một cách cẩn thận bởi các cấu trúc gọi là cổng (gate). Các gate này là tuỳ chọn để định nghĩa thông tin đi qua. Chúng được tạo bởi lớp mạng thần kinh sigmoid và một nhân các thao tác toán tử pointwise.

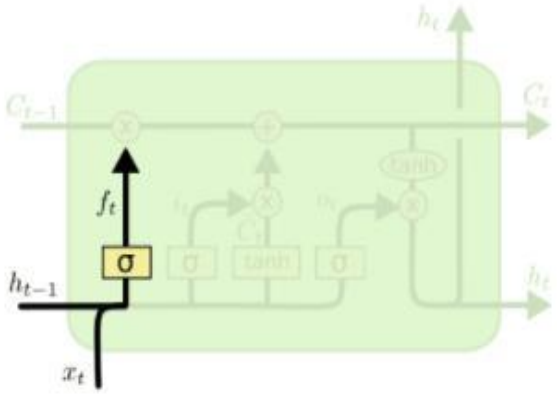


Hình 2.17: Cổng trạng thái LSTM [21]

Sigmoid layer outputs có giá trị từ 0 – 1, mô tả mức độ thông tin các thành phần được phép truyền qua tại mỗi lớp mạng. Nếu kết quả bằng 0 điều này có nghĩa rằng "không để cho bất kỳ thứ gì đi qua", ngược lại nếu thu được giá trị là 1 thì có nghĩa là "cho phép mọi thứ đi qua". Một LSTM có ba trong số các cổng như vậy để bảo vệ và điều khiển kiểm soát trạng thái cell state.

Các bước cơ bản về quá trình hoạt động của LSTM như sau. Bước thứ nhất của mô hình LSTM là quyết định xem thông tin nào cần loại bỏ khỏi cell state. Tiến trình này được thực hiện thông qua một sigmoid layer gọi là "forget gate layer" (cánh gate quên lãng) – 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

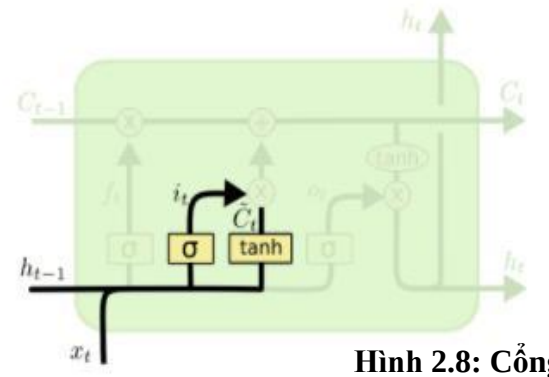
.



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

Hình 2.7: Cổng chặn ft[21]

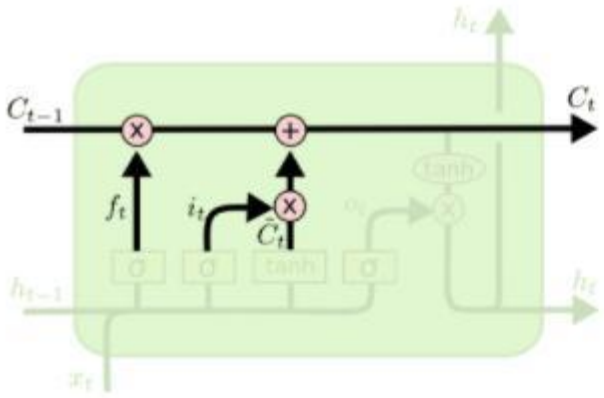
Bước thứ hai, tại cell state cần quyết định thông tin nào cần được lưu lại. Có hai phần là single sigmoid layer được gọi là "input gate layer"- cổng vào quyết định các giá trị sẽ cập nhật. Tiếp theo, một tanh layer tạo ra một vector mới $\tilde{C}t$ được thêm vào trong cell state.



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

Hình 2.8: Cổng vào it và tanh[21]

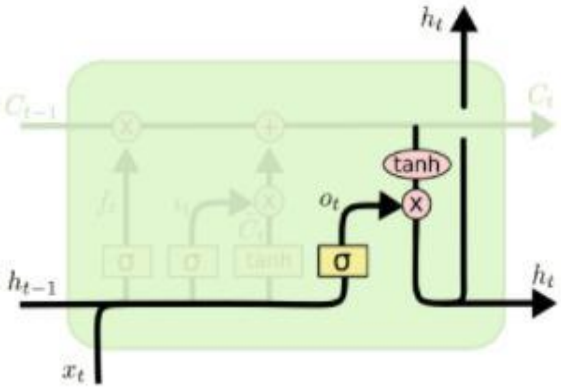
Bước thứ ba, sẽ kết hợp hai thành phần này lại để cập nhật vào cell state. Lúc cập nhật vào cell state cũ, C_{t-1}, vào cell state mới C_t. Đưa state cũ hàm f_t, để quên đi những gì trước đó. Sau đó, sẽ thêm (i_t*C_t). Đây là giá trị ứng viên mới, co giãn (scale) số lượng giá trị mà ta muốn cập nhật cho mỗi state.



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

Hình 2.9: Giá trị state Ct [21]

Bước cuối cùng, cần quyết định xem thông tin đầu ra là gì. Dữ liệu đầu ra này cần dựa trên cell state, nhưng sẽ được lọc bớt thông tin. Đầu tiên, áp dụng lớp sigmoid đơn để quyết định xem phần nào của cell state dự định sẽ đầu ra. Sau đó, 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 muốn output ra ngoài.



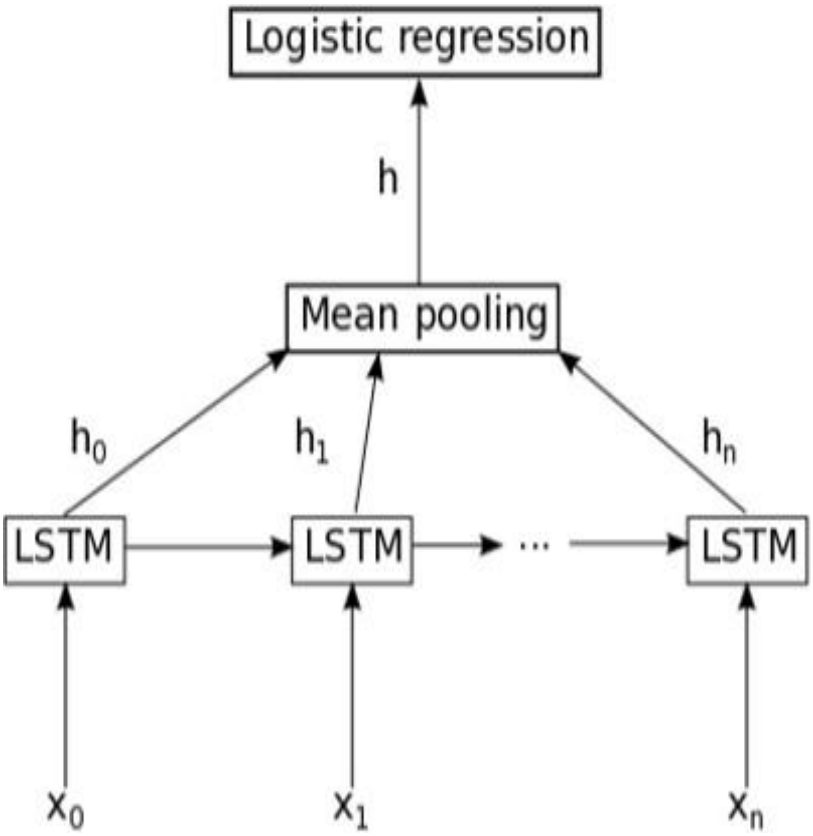
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Hình 2.10: Giá trị cổng ra và vector trạng thái ẩn ht[21]

Mạng bộ nhớ ngắn dài 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 ft, cổng vào it và cổng ra ot. Trong đó cổng chặn ft chính là tư tưởng chủ đạo của

mạng LSTM khi cho phép điều khiển lượng thông tin đầu vào ht-1 từ các thời điểm trước.

Với ưu điểm về lưu trữ phụ thuộc dài, model sử dụng để huấn luyện trong luận văn này là model LSTM. Mô hình được luận văn sử dụng được mô tả trong hình 2.19 gồm một lớp LSTM duy nhất sau đó là một lớp tổng hợp trung bình (full-connection) và một lớp hồi quy logistic.



Hình 2.11: Mô hình LSTM luận văn sử dụng

(*http://hoctructuyen123.net/tong-quan-ve-phan-tich-cam-xuc-trong-tieng-viet/)*

Từ một chuỗi đầu vào x0, x1,… xn 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 h0, h1,…hn . 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.

2.6. Kết luận chương 2:

Chương 2 đã giới thiệu về hướng tiếp cận, các công trình nghiên cứu, kỹ thuật liên quan để phục vụ giải quyết bài toán. Chương này đi sâu về áp dụng phương pháp học máy phân lớp và phương pháp biểu diễu các đặc trưng mô hình trong bài toán phát hiện câu chứa gợi ý trên diễn đàn trực tuyến sử dụng mạng Nơ ron.

Chương tiếp theo sẽ trình bày về hệ thống phát hiện câu chứa gợi ý trên diễn đàn trực tuyến, mô hình giải quyết bài toán, tập dữ liệu sử dụng, cách thức tiến hành thực nghiệm, kết quả thực nghiệm.

CHƯƠNG 3: THỰC NGHIỆM VÀ ĐÁNH GIÁ

Dựa vào những thuật toán áp dụng cho bài toán phân loại câu và tìm hiểu các phương pháp giải quyết bài toán, trong chương này, luận văn trình bày chi tiết quá trình thực nghiệm gồm có quá trình thu thập, thiết lập thực nghiệm, các phương pháp làm thực nghiệm, kết quả và đánh giá sau thực nghiệm.

3.1. Thông tin về bộ dữ liệu

Luận văn sử dụng dữ liệu thực nghiệm được thu thập từ Bộ dữ liệu tiếng Anh Semeval2019Task9/Subtask-A [5] tổng hợp từ các câu trên diễn đàn trực tuyến về sự phát triển nền tảng Window với dữ liệu huấn luyện 8500 câu và dữ liệu kiểm thử 833 câu. Mỗi câu được gắn nhãn phân loại theo các mục "có gợi ý", "không gợi ý" . Với dữ liệu này giúp chúng ta huấn luyện cho mạng neural của máy.

Mô tả file dữ liệu gồm:

Bảng 3.1: Mô tả dữ liệu thực nghiệm

Số thứ tự	**Tên cột**	**Ghi chú**
1	ID	Mã của câu
2	Classification	Phân loại câu: Giá trị sẽ là "có gợi ý" hay "không gợi ý"
3	Sentence	Nội dung của câu (tối đa 140 ký tự)

(Nguồn: https://github.com/Semeval2019Task9/Subtask-A *)*

Bộ dữ liệu tiếng Anh Semeval2019Task9/Subtask-A [5] này gồm các thư mục để huấn luyện, kiểm thử và cho kết quả

- Data:

  File "**V1.4_Training_new.csv**" sử dụng cho huấn luyện

  File "**SubtaskA_EvaluationData_labeled.csv**" sử dụng cho kiểm thử

Bảng 3.2: Mô tả phân loại nhãn cho các tập dữ liệu thực nghiệm

Tập dữ liệu	***Câu Có gợi ý***	***Câu Không gợi ý***	***Tổng cộng***
Dữ liệu huấn luyện	2085	6415	8500
Dữ liệu kiểm thử	87	746	833

3.2. Môi trường thực nghiệm:

3.2.1. Ngôn ngữ lập trình python:

Python là một ngôn ngữ lập trình thông dịch do Guido van Rossum tạo ra năm 1990 [22]. Python hoàn toàn tạo kiểu động và dùng cơ chế cấp phát bộ nhớ tự động, do vậy nó tương tự như Perl, Ruby, Scheme, Smalltalk, và Tcl. Python được phát triển trong một dự án mã mở, do tổ chức phi lợi nhuận Python Software Foundation quản lý.

➢ Sau đây là các đặc điểm của Python:

- Ngữ pháp đơn giản, dễ đọc.
- Vừa hướng thủ tục (procedural-oriented), vừa hướng đối tượng (object-oriented)
- Hỗ trợ module và hỗ trợ gói (package)
- Xử lý lỗi bằng ngoại lệ (Exception)
- Kiểu dữ liệu động ở mức cao.
- Có các bộ thư viện chuẩn và các module ngoài, đáp ứng tất cả các nhu cầu lập trình.
- Có khả năng tương tác với các module khác viết trên C/C++ (Hoặc Java cho Jython, hoặc .Net cho IronPython).
- Có thể nhúng vào ứng dụng như một giao tiếp kịch bản (scripting interface).

```
# Python 3: Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

Hình 3.1: Mô tả cú pháp, các dòng lệnh trong Python

(Nguồn: https://www.python.org/)

Hiện nay ngôn ngữ **Python** được xếp hạng thứ 3 trong Top 10 các ngôn ngữ lập trình phổ biến nhất đang được thế giới sử dụng:

Bảng 3.3: Bảng xếp hạng các ngôn ngữ lập trình năm 2020

Feb-20	Feb-19	Change	Programming Language	Ratings	Change
1	1		Java	17.36%	1.48%
2	**2**		**C**	**16.77%**	**4.34%**
3	3		Python	9.35%	1.77%
4	4		C++	6.16%	-1.28%
5	7	^	C#	5.93%	3.08%
6	5	v	Visual Basic .NET	5.86%	-1.23%
7	6	v	JavaScript	2.06%	-0.79%
8	8		PHP	2.02%	-0.25%
9	9		SQL	1.53%	-0.37%
10	20	^^	Swift	1.46%	0.54%

(Nguồn: https://www.tiobe.com/)

3.2.2 Giới thiệu về thư viện TensorFlow

TensorFlow là một thư viện do nhóm phát triển Google Brain của Google phát triển và phát hành mã nguồn mở vào tháng 11/2015. TensorFlow được cho là sử dụng trong nhiều sản phẩm thương mại của Google [23]. Hiện tại được sử dụng nhiều trong quá trình hiện thực hoá mạng neural trong Deep Learning.

Các khái niệm cơ bản:

Tensor: Tensor là khái niệm cơ bản nhất trong TensorFlow.

- Tensor là cấu trúc dữ liệu được sử dụng trong toàn TensorFlow, đại diện cho tất cả các loại dữ liệu. Hay nói cách khác là tất cả các loại dữ liệu đều là tensor.
- Việc trao đổi dữ liệu trong quá trình xử lý chỉ thông qua tensor.
- Hiểu đơn giản thì tensor là mảng n chiều hay list cộng thêm 1 vài thứ thú vị khác.
- Tensor có 3 thuộc tính là **Rank**, **Shape**, **Type**. **Rank**: là số chiều của dữ liệu.

Bảng 3.4: Mô tả rank của tensor [23]

Rank	đơn vị số học	Ví dụ Python
0	Scalar	*s = 483*
1	Vector	*v = [1.1, 2.2, 3.3]*
2	Matrix	*m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]*
3	3-Tensor	*t = [[[2], [4], [6]], [[8], [10], [12]], [[14], [16], [18]]]*
n	n-Tensor	*(n chiều) ….*

Shape: là chiều của tensor.

Ví dụ: t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]] có shape là [3, 3].

Bảng 3.5: Mô tả cú pháp shape của tensor [23]

Rank	Shape	Số chiều (Dimension)	Ví dụ
0	[]	0-D	*A 0-D tensor. A scalar.*
1	[D0]	1-D	*A 1-D tensor with shape [5].*
2	[D0, D1]	2-D	*A 2-D tensor with shape [3, 4].*
3	[D0, D1, D2]	3-D	*A 3-D tensor with shape [1, 4, 3].*
n	[D0, D1, … Dn-1]	n-D	*A tensor with shape [D0, D1, ... Dn-1].*

Type: kiểu dữ liệu.

Bảng 3.6: Mô tả kiểu dữ liệu trong tensorflow [23]

Loại dữ liệu	Loại python	Mô tả
DT_FLOAT	tf.float32	*32 bits floating point.*
DT_DOUBLE	tf.float64	*64 bits floating point.*
DT_INT8	tf.int8	*8 bits signed integer.*
DT_INT16	tf.int16	*16 bits signed integer.*
DT_INT32	tf.int32	*32 bits signed integer.*
DT_INT64	tf.int64	*64 bits signed integer.*

DT_UINT8	tf.uint8	*8 bits unsigned integer.*
DT_STRING	tf.string	*Variable length byte arrays. Each element of a Tensor is a byte array.*
DT_BOOL	tf.bool	*Boolean.*
DT_COMPLEX64	tf.complex64	*Complex number made of two 32 bits floating points: real and imaginary parts.*
DT_QINT8	tf.qint8	*8 bits signed integer used in quantized Ops.*
DT_QINT32	tf.qint32	*32 bits signed integer used in quantized Ops.*
DT_QUINT8	tf.quint8	*8 bits unsigned integer used in quantized Ops.*

Variable: lưu trạng thái (state) sau khi tính toán graph.

3.3. Phương pháp thực nghiệm:

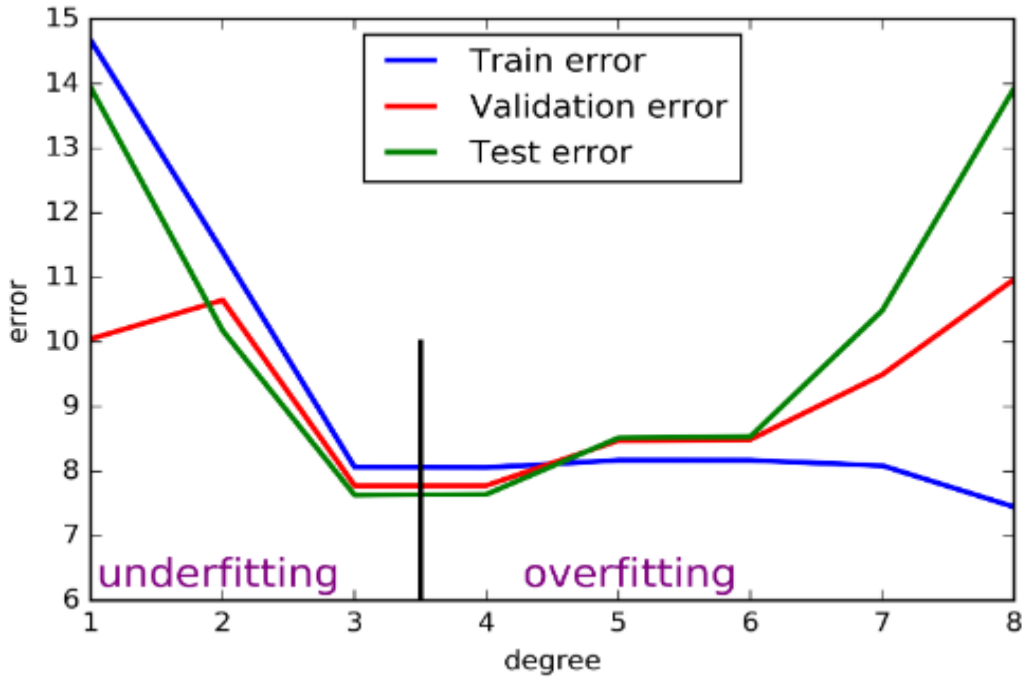
3.3.1. Cách chia dữ liệu:

Với việc chia tập dữ liệu ra thành hai tập nhỏ: training data và test data. Phương pháp sẽ là trích từ tập training data ra một tập con nhỏ và thực hiện việc đánh giá mô hình trên tập con nhỏ này. Tập con nhỏ được trích ra từ training set này được gọi là validation set. Lúc này, training set là phần còn lại của training set ban đầu**.** Train error được tính trên training set mới này, và có một khái niệm nữa được định nghĩa tương tự như trên validation error, tức error được tính trên tập validation.

Với mô hình sao cho cả train eror *và* validation error đều nhỏ, qua đó có thể dự đoán được rằng test error cũng nhỏ. Phương pháp thường được sử dụng là sử dụng nhiều mô hình khác nhau. Mô hình nào cho validation error nhỏ nhất sẽ là mô hình tốt.

Bắt đầu từ mô hình đơn giản, sau đó tăng dần độ phức tạp của mô hình. Tới khi nào validation error có chiều hướng tăng lên thì chọn mô hình ngay trước đó. Với mô hình càng phức tạp, train error có xu hướng càng nhỏ đi.

Hình 3.2 mô tả ví dụ phía trên với bậc của đa thức tăng từ 1 đến 8. Tập validation bao gồm 10 điểm được lấy ra từ tập training ban đầu



Hình 3.2: Lựa chọn mô hình dựa trên validation[26]

Xét hai đường màu lam và đỏ, tương ứng với train error *và* validation error. Khi bậc của đa thức tăng lên, train error có xu hướng giảm. Điều này dễ hiểu vì đa thức bậc càng cao, dữ liệu càng được fit. Quan sát đường màu đỏ, khi bậc của đa thức là 3 hoặc 4 thì validation error thấp, sau đó tăng dần lên. Dựa *vào* validation error, ta có thể xác định được bậc cần chọn là 3 hoặc 4. Quan sát tiếp đường màu lục, tương ứng *với* test error, thật là trùng hợp, với bậc bằng 3 hoặc 4, test error cũng đạt giá trị nhỏ nhất, sau đó tăng dần lên.

Việc không sử dụng test data khi lựa chọn mô hình ở trên nhưng vẫn có được kết quả khả quan vì ta giả sử rằng validation data và test data có chung một đặc điểm nào đó. Và khi cả hai đều là unseen data, error trên hai tập này sẽ tương đối giống nhau.

Nhắc lại rằng, khi bậc nhỏ (bằng 1 hoặc 2), cả ba error đều cao, ta nói mô hình bị underfitting.

Trong nhiều trường hợp, để hạn chế số lượng dữ liệu để xây dựng mô hình. Nếu lấy quá nhiều dữ liệu trong tập training ra làm dữ liệu validation, phần dữ liệu còn lại của tập training là không đủ để xây dựng mô hình. Lúc này, tập validation

phải thật nhỏ để giữ được lượng dữ liệu cho training đủ lớn. Tuy nhiên, một vấn đề khác nảy sinh. Khi tập validation quá nhỏ, hiện tượng overfitting lại có thể xảy ra với tập training còn lại. Đề khắc phục hiện tượng này ta dùng kiểm chứng chéo(Cross validation)

Cross validation là một cải tiến của validation với lượng dữ liệu trong tập validation là nhỏ nhưng chất lượng mô hình được đánh giá trên nhiều tập validation khác nhau. Một cách thường được sử dụng là chia tập training ra k tập con không có phần tử chung, có kích thước gần bằng nhau. Tại mỗi lần kiểm thử , được gọi là run, một trong số k tập con được lấy ra làm validata set. Mô hình sẽ được xây dựng dựa vào hợp của k−1 tập con còn lại. Mô hình cuối được xác định dựa trên trung bình của các train error và validation error. Cách làm này còn có tên gọi là k-fold cross validation.

Khi k bằng với số lượng phần tử trong tập training ban đầu, tức mỗi tập con có đúng 1 phần tử, ta gọi kỹ thuật này là leave-one-out.

Sklearn hỗ trợ rất nhiều phương thức cho phân chia dữ liệu và tính toán scores của các mô hình.

Tham số quan trọng trong kỹ thuật này là k, đại diện cho số nhóm mà dữ liệu sẽ được chia ra. Vì lý do đó, nó được mang tên k-fold cross-validation. Khi giá trị của k được lựa chọn, người ta sử dụng trực tiếp giá trị đó trong tên của phương pháp đánh giá. Ví dụ với k=10, phương pháp sẽ mang tên 10-fold cross-validation.

Kỹ thuật này thường bao gồm các bước như sau:

- ✓ Xáo trộn dataset một cách ngẫu nhiên
- ✓ Chia dataset thành k nhóm
- ✓ Với mỗi nhóm:
  - Sử dụng nhóm hiện tại để đánh giá hiệu quả mô hình
  - Các nhóm còn lại được sử dụng để huấn luyện mô hình
  - Huấn luyện mô hình
  - Đánh giá và sau đó hủy mô hình
- ✓ Tổng hợp hiệu quả của mô hình dựa từ các số liệu đánh giá

Kết quả tổng hợp thường là trung bình của các lần đánh giá. Ngoài ra việc bổ sung thông tin về phương sai và độ lệch chuẩn vào kết quả tổng hợp cũng được sử dụng trong thực tế.

Giá trị k là thông số quan trọng để có thể đánh giá chính xác mô hình, vậy thì lựa chọn thông số này như thế nào?

Ba cách phổ biến để lựa chọn k:

- **Đại diện**: Giá trị của k được chọn để mỗi tập train/test đủ lớn, có thể đại diện về mặt thống kê cho dataset chứa nó.
- **k=10**: Giá trị của k được gán cố định bằng 10, một giá trị thường được sử dụng và được chứng minh là cho sai số nhỏ, phương sai thấp (thông qua thực nghiệm).
- **k=n**: Giá trị của k được gán cố định bằng n , với n là kích thước của dataset, như vậy mỗi mẫu sẽ được sử dụng để đánh giá mô hình một lần. Cách tiếp cận này còn có tên leave-one-out cross-validation.

Giá trị k=10 là một cấu hình rất phổ biến.

Sau quá trình nghiên cứu và tìm hiểu các phương pháp đánh giá thực nghiệm, luận văn đề xuất sử dụng phương pháp K-fold Cross Validation.

3.3.2. Cách thức đánh giá:

Để đánh giá hiệu quả phân lớp luận văn sử dụng các độ đo F-score. Giá trị F-Score phụ thuộc vào Precision và Recall. Trong đó, Precision là độ đo thể hiện độ chính xác của bộ phân lớp, được xác định bằng số bình luận được phân lớp đúng trên tổng số bình luận được phân vào lớp đó. Recall là độ đo thể hiện khả năng không phân lớp sai các bình luận, được xác định bằng số bình luận được phân lớp đúng trên tổng số bình luận thực tế thuộc lớp đó. F-score là độ được xác định thông qua Precision và Recall (giá trị của các độ đo này càng cao thì bộ phân lớp càng có hiệu quả phân lớp tốt). Cụ thể:

$$\Pr ecision = \frac{TP}{TP + FP}$$

$$\Re call = \frac{TP}{TP + FN}$$

$$F - score = \frac{2 \times \Pr ecison \times \Re call}{\Pr ecison + \Re call}$$

Hình 3.3: Công thức tính độ đo

Trong đó:

TP (True Positive): là số bình luận được phân lớp là y và việc phân lớp này là đúng.

FP (False Positive): là số bình luận được phân lớp là y nhưng việc phân lớp này là sai.

FN (False Negative): là số bình luận thuộc lớp y nhưng bị gán nhãn vào lớp khác.

Quá trình thực nghiệm thuật toán gồm các giai đoạn chính:

- Tiền xử lý dữ liệu: Loại bỏ các dư thừa, các từ vô nghĩa trong câu.
- Xây dựng bộ phân lớp dữ liệu: Sử dụng CNN, RNN và LSTM

Tiền xử lý dữ liệu: Luận văn sử dụng ngôn ngữ python để xử lý các dữ liệu

Mô hình phân lớp: Mô hình mà luận văn sử dụng được mô tả trong phần 2.3 về mô hình CNN và phần 2.4 về mô hình RNN, 2.5 về mô hình LSTM.

Trong nghiên cứu này thực hiện việc xây dựng sử dụng mạng Nơ ron cho việc phát hiện câu chứa gợi ý trên diễn đàn trực tuyến. Bao gồm các kiến trúc mạng CNN, RNN,LSTM. Với việc huấn luyện cho các mô hình để phát hiện câu chứa gợi ý trên diễn đàn trực tuyến cho ra một lớp softmax để phân loại câu. Mạng neural sâu được đào tạo trên các từ nhúng (embedding words) từ trước tập huấn luyện về các số liệu thông thường thu thập thông tin.

Dựa trên các mô hình mạng neural trong nghiên cứu xây dựng chương trình thực nghiệm cho phân loại câu chứa gợi ý trên diễn đàn trực tuyến . Bao gồm thiết kế các mạng neural CNN, RNN, LSTM trên ngôn ngữ python, huấn luyện và kiểm thử cho các tập dữ liệu, sau đó cho ra kết quả là độ chính xác của các giải thuật để kiểm tra xem thuật toán nào cho ra kết quả dự đoán tốt nhất.

3.4. Tiến hành thực nghiệm

3.4.1. Xây dựng các thành phần chung cho các mô hình:

➢ **Xây dựng class và khai báo tham số đầu vào cho các mô hình mạng:**

Mỗi mô hình mạng neural sẽ được khai báo bởi 1 class với hàm init để khởi tạo các tham số đầu vào của mạng. Các tham số đầu vào cho mô hình mạng:

```
import tensorflow as tf



class Text_Merge_NN(object):

    """

    A neural network mixed (CNN, RNN) for text classification.

    Uses an embedding layer, followed by a convolutional, max-pooling and
softmax layer.

    Uses an embedding layer, GRUCell and output states of RNN
    """


    def __init__(self, sequence_length, num_classes, vocab_size, embedding_size,
filter_sizes, num_filters, hidden_unit, l2_reg_lambda=0.0):
```

sequence_length: Chiều dài của 1 tin nhắn. Các câu trong tập dữ liệu đã được thêm padded để có cùng độ dài cho các tin nhắn.

num_classes: Số phân loại cho lớp đầu ra (out layer)

vocab_size: Kích thước của từ vựng. Cần thiết để xác định kích thước của lớp nhúng (embedding layer), có dạng (shape) [vocabulary_size, embedding_size].

embedding_size: Số chiều của các từ nhúng *(trong nghiên cứu sừ dụng giá trị 300).*

filter_sizes: Số từ được tích chập với nhau trong bộ lọc.

num_filters: Số lượng bộ lọc trên một kích thước bộ *lọc (trong nghiên cứu sử dụng giá trị là 128).*

hidden_unit: Số lớp ẩn trong mạng neural tái phát *(trong nghiên cứu sử dụng giá trị là 128).*

l2_reg_lambda: Giá trị tính tỉ lệ thất bại, mặc định là 0.0 *(trong nghiên cứu sử dụng giá trị là 1.0).*

➢ ***Định nghĩa dữ liệu đầu vào để truyền cho mô hình mạng***

```
# Placeholders for input, output and dropout

self.input_x = tf.placeholder(tf.int32, [None, sequence_length], name="input_x")

self.input_y = tf.placeholder(tf.float32, [None, num_classes], name="input_y")
self.dropout_keep_prob_cnn = tf.placeholder(tf.float32, name="dropout_keep_prob_cnn")

self.dropout_keep_prob_rnn = tf.placeholder(tf.float32, name="dropout_keep_prob_rnn")

self.batch_size = tf.placeholder(tf.int32, name='batch_size') self.pad =
tf.placeholder(tf.float32, [None, 1, embedding_size, 1], name='pad')

self.real_len = tf.placeholder(tf.int32, [None], name='real_len')
```

tf.placeholder: Tạo ra một biến vị trí để đưa vào mạng khi thực hiện huấn luyện hay được kiểm thử. Tham số thứ 2 là hình của bộ dữ liệu đầu vào, sử dụng **None** cho phép mạng có thể xử lý các lô kích thước tùy ý.

➢ **Lớp nhúng (Embedding layer)**

tf.device('/cpu:0'): Buộc một hoạt động được thực hiện trên CPU. Mặc định

```
# Embedding layer

with tf.device('/cpu:0'), tf.name_scope("embedding"): self.W = tf.Variable(

        tf.random_uniform([vocab_size, embedding_size], -1.0, 1.0), name="W")

    self.embedded_chars = tf.nn.embedding_lookup(self.W, self.input_x) emb =
    tf.expand_dims(self.embedded_chars, -1)


# Zero paddings so that the convolution output have dimension batch x sequence_length
x emb_size x channel

num_prio = (filter_sizes[0] - 1) // 2

num_post = (filter_sizes[0] - 1) - num_prio

pad_prio = tf.concat([self.pad] * num_prio, 1, name="pad_prio") pad_post =
tf.concat([self.pad] * num_post, 1, name="pad_post") self.embedded_chars_pad =
tf.concat([pad_prio, emb, pad_post], 1, name="embedded_chars_pad")
```

TensorFlow sẽ cố gắng đưa hoạt động trên GPU nếu có sẵn.

tf.name_scope: Tạo một tên mới phạm vi với tên "**embedding**". Phạm vi cho biết thêm tất cả các hoạt động vào một nút cấp cao nhất được gọi là "**embedding**" để có được một hệ thống phân cấp tốt khi thể hiện trong TensorBoard.

self.W: Là ma trận nhúng mà chúng được học trong quá trình huấn luyện. Nghiên cứu khởi tạo nó bằng cách sử dụng phân bố ngẫu nhiên ngẫu nhiên.

tf.nn.embedding_lookup: Tạo ra hoạt động nhúng thực sự. Kết quả của hoạt động nhúng là một ma trận có dạng 3 chiều **[None, sequence_length, embedding_size]**.

Hoạt động của tích chập **conv2d** của TensorFlow mong muốn một bộ kiểm soát 4 chiều có kích thước tương ứng với lô, chiều rộng, chiều cao và kênh. Kết quả của nhúng của mô hình không chứa kích thước kênh, vì vậy nghiên cứu tạo ra hình dạng mới **[None, sequence_length, embedding_size,1]**.

Tính các điểm số (scores) và dự đoán kết quả (predictions)

```
# Final (unnormalized) scores and predictions (CNN + GRNN)

output = outputs_final[0]

with tf.variable_scope('Output'):
     tf.get_variable_scope().reuse_variables() one =
     tf.ones([1, hidden_unit], tf.float32) for i in
     range(1, len(outputs_final)):

          ind = self.real_len < (i + 1) ind =
          tf.to_float(ind)

          ind = tf.expand_dims(ind, -1) mat =
          tf.matmul(ind, one)

          output = tf.add(tf.multiply(output, mat),
tf.multiply(outputs_final[i], 1.0 - mat))


with tf.name_scope('output'):

     W = tf.Variable(tf.truncated_normal([hidden_unit, num_classes], stddev=0.1),
name='W')

     b = tf.Variable(tf.constant(0.1, shape=[num_classes]), name='b') l2_loss +=
     tf.nn.l2_loss(W)

     l2_loss += tf.nn.l2_loss(b)

     self.scores = tf.nn.xw_plus_b(output, W, b, name='scores') self.predictions =
     tf.argmax(self.scores, 1, name='predictions')
```

Sử dụng vector đặc tính từ kết quả kết hợp hai kết quả của 2 mạng, nghiên cứu ra các dự đoán bằng cách làm một phép nhân và chọn lớp với điểm số cao nhất. Sử dụng **tf.nn.xw_plus_b** để thực hiện phép nhân ma trận W_x+b.

- **Tính tỉ lệ lỗi và độ chính xác:**

```
# CalculateMean cross-entropy loss
with tf.name_scope("loss"):
    losses = tf.nn.softmax_cross_entropy_with_logits(logits=self.scores, labels=self.input_y)
    self.loss = tf.reduce_mean(losses) + l2_reg_lambda * l2_loss

# Accuracy
with tf.name_scope("accuracy"):
    correct_predictions = tf.equal(self.predictions, tf.argmax(self.input_y,
1))
    self.accuracy = tf.reduce_mean(tf.cast(correct_predictions, "float"), name="accuracy")
```

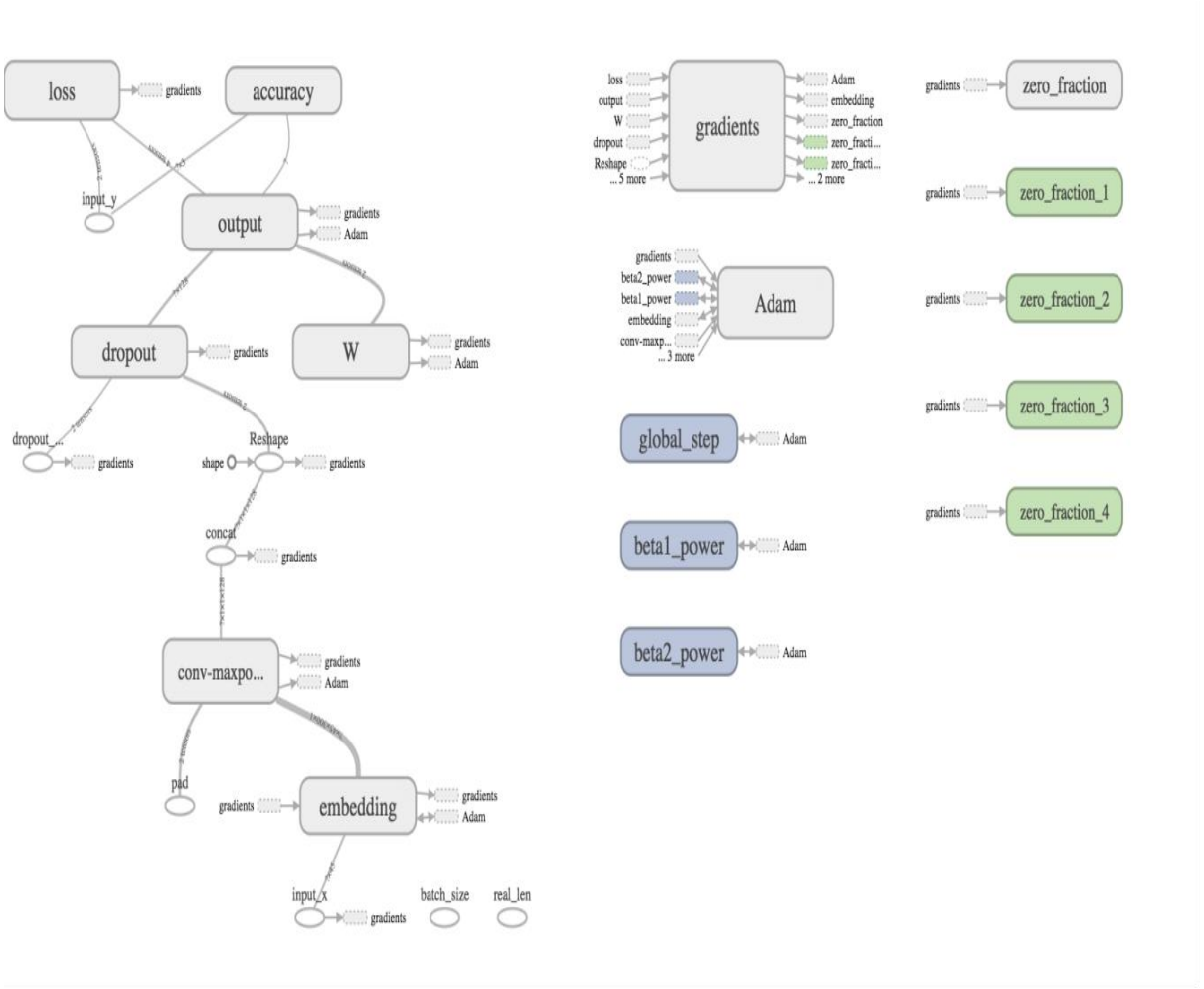
Chức năng **tf.nn.softmax_cross_entropy_with_logits** thuận tiện tính toán tỉ lệ mất cho mỗi lớp, cho dự đoán và các nhãn đầu vào chính xác, kết quả nhận được số lượng mất với kích thước lô và dữ liệu đào tạo. Xác định số lượng chính xác để theo dõi trong quá trình đào tạo và thử nghiệm.

3.4.2. Mã lệnh cài đặt các mô hình bằng ngôn ngữ Python trên Tensorflow:

3.4.2.1. Mô hình mạng neural CNN (Lớp tích chập và max-pooling)

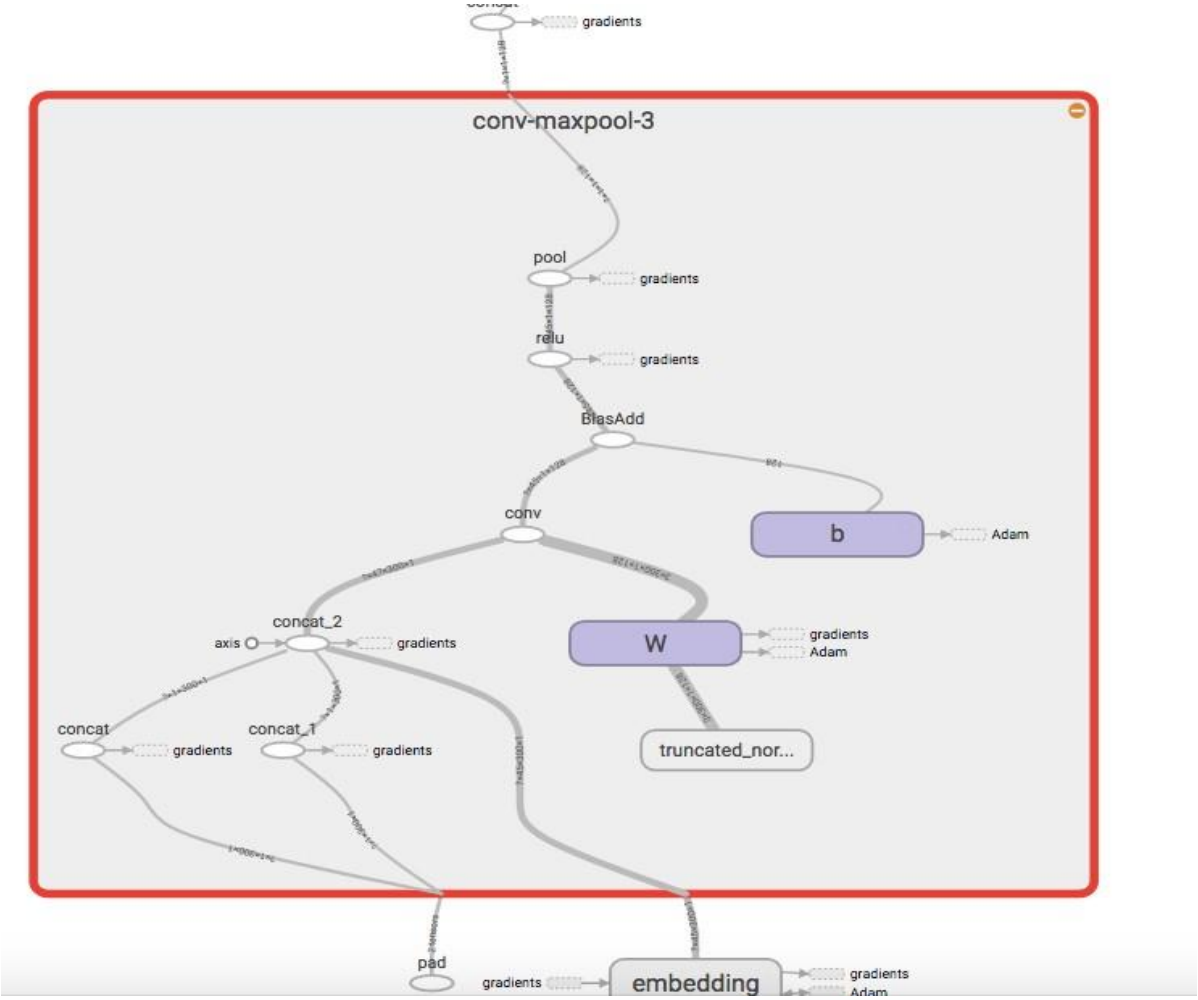
Xây dựng lớp tích chập cho CNN và theo sau đó là max-pooling sử dụng bộ lọc có kích thước bằng 3.

Mô hình mạng trong TensorBoard:



Hình 3.4: Mô hình mạng CNN trong nghiên cứu.

Ở đây, **W** là ma trận lọc và **h** là kết quả của việc áp dụng độ phi tuyến cho đầu ra xoắn. Mỗi bộ lọc trượt trên toàn bộ nhúng, nhưng thay đổi trong bao nhiêu từ nó bao gồm. "VALID" padding có nghĩa là mô hình lướt bộ lọc qua câu mà không cần thêm bộ đệm vào, thực hiện một chập có giới hạn cho chúng ta một đầu ra với hình dạng [1, sequence_length - filter_size + 1, 1, 1]. Thực hiện max-pooling trên đầu ra của một kích thước bộ lọc cụ thể cho ra một tensor của hình dạng [batch_size, 1, 1, num_filters]. Đây thực chất là một vector đặc tính, trong đó kích thước cuối cùng tương ứng với các tính năng của mô hình có dạng [None, 1, 1, num_filters]

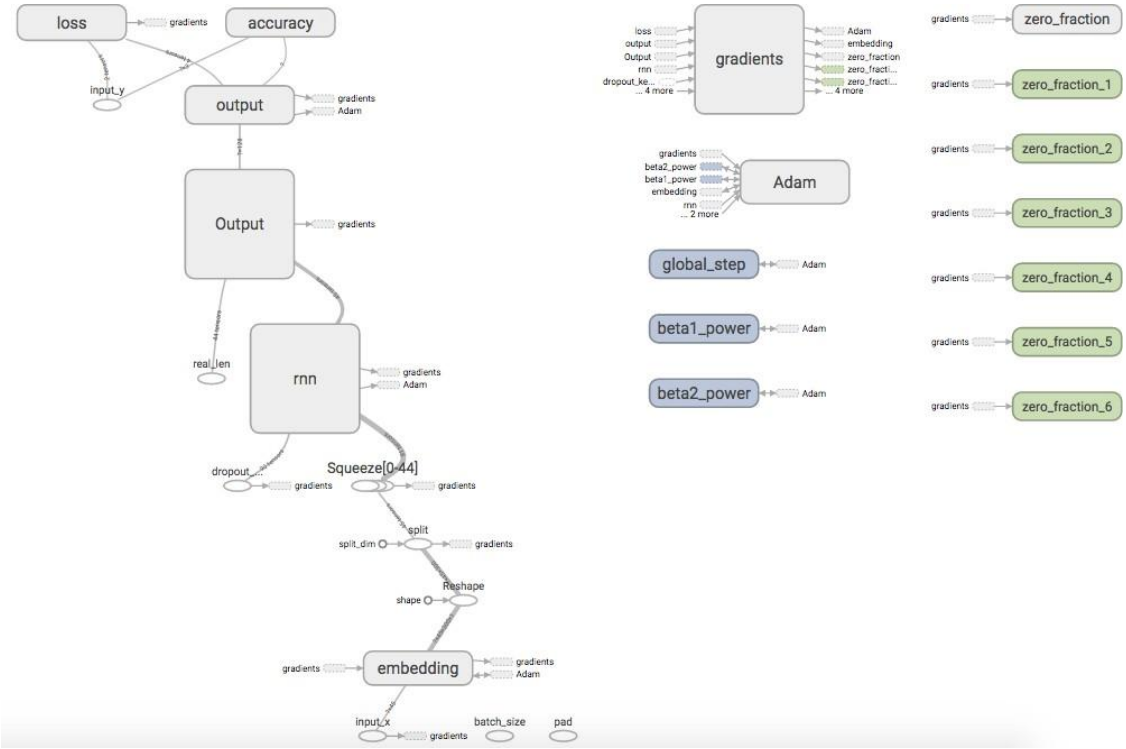


Hình 3.5: Mô hình conv-maxpool trong mạng CNN

3.4.2.2. Mô hình mạng neural RNN (Lớp ẩn sử dụng GRU cell)

Dữ liệu đầu vào **input** là từ nhúng (embedding word) 3 chiều dạng **[None, sequence_length, embedding_size],** được chuyển đổi sang dãy các vector 2 chiều dạng **[None, embedding_size].** Trong RNN các vector 2 chiều sẽ được lặp lại (recurrent) từ các **gru_cell_0** đến **gru_cell_38**. Mỗi cell sẽ cho ra các dropout_ là kết quả của từng cell, bên cạnh đó cũng truyền tải các kết quả đó cho các cell tiếp theo. Đầu ra của RNN là dãy các vector có dạng **[None, hidden_unit].**

Mô hình mạng trong TensorBoard:



Hình 3.6: Mô hình mạng RNN nghiên cứu

3.4.2.3. Mô hình LSTM:

Để huấn luyện mô hình LSTM đưa vào mô hình batch_size số câu trong một lượt huấn luyện. Cách đưa vào batch_size 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. .

Xây dựng mô hình LSTM sử dụng thư viện TensorFlow [23] Trước tiên, cần tạo TensorFlow graph. Để xây dựng TensorFlow graph, 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 thể 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 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 echpo,bacth size, 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.

3.5. Kết quả chạy thực nghiệm

Với mạng Nơron tôi nhận thấy trong các mô hình mạng CNN,RNN,LSTM các tham số ảnh hưởng tới mạng mô hình là Epoch và Batch size. Trong mô hình mạng các ma trận các lớp gần nhau và nhận thấy tham số Epoch và Bath size có ảnh hưởng nhất định tới kết quả mô hình đạo tạo.

Epoch là một hyperparameter trong ANN, được dùng để định nghĩa số lần learning algorithm hoạt động trên model, một epoch hoàn thành là khi tất cả dữ liệu training được đưa vào mạng neural network một lần (đã bao gồm cả 2 bước forward và backward cho việc cập nhật internal model parameters).

Thường chúng ta cần một số lượng lớn Epoch để training cho ANN (10, 100, 500, 1000…) tuy nhiên cũng còn tùy thuộc vào bài toán và tài nguyên máy tính. Một cách khác là sử dụng Learning Curve để tìm số epoch.

Một tập training dataset có thể được chia nhỏ thành các batches (sets, parts).

Một batch sẽ chứa các training samples, và số lượng các samples này được gọi là batch size. Cần lưu ý có 2 khái niệm khác nhau là batch size và number of batches (số lượng các batches) or iterations. Tùy thuộc vào batch size mà GD sẽ có các biến thể khác nhau:

- Batch Gradient Descent: Batch Size = Size of Training Dataset
- Stochastic Gradient Descent: Batch Size = 1
- Mini-Batch Gradient Descent: 1 < Batch Size < Size of Training Set

Thông thường thi Mini-Batch Gradient Descent được sử dụng nhiều cho các bài toán tối ưu vì tính hội tụ ổn định hơn so với Stochastic Gradient Descent. Dữ liệu trước khi đưa vào 2 dạng này thường được chọn một cách ngẫu nhiên từ training dataset. Đối với Mini-Batch Gradient Descent thì batch size thường được chọn là lũy thừa của 2 (32, 64, 128, 256…) vì tốc độ tính toán sẽ tối ưu cho các arithmetic algorithms của CPU và GPU. Cách chọn batch size cũng tùy theo yêu cầu của bài toán. Trường hợp chia không "chẵn" số batch size theo training dataset thì batch cuối cùng sẽ có ít samples hơn các batches khác.

Nhận thấy điều này tôi đã thiết lập mô hình với các cặp tham số Epoch và Batch size, với Epoch tôi lần lượt chạy với các vòng lặp 5,10,20, với Batch size tôi chạy với các vòng lặp 32,64,128 với số lượng đặc trưng không đổi sẽ cho ra kết quả thể hiện trong bảng sau:

a, Kết quả thực nghiệm:

❖ ***Kết quả với mô hình CNN:***

Bảng 3.7: Kết quả sử dụng mô hình CNN

CNN									
Epoch	5			10			20		
Batch size	32			64			128		
Accuracy %	71.16			74.25			81.52		
Độ đo	***Pre%***	***Re%***	***F1%***	***Pre%***	***Re%***	***F1%***	***Pre%***	***Re%***	***F1%***
Gợi ý	77.58	70.20	73.71	80.98	78.32	79.63	84.07	79.16	81.45
Không gợi ý	77.32	72.03	74.62	77.99	78.67	78.33	88.91	82.42	83.04

Dựa vào bảng kết quả 3.7 ta có thể thấy:

- Với Epoch= 20 và Batch size = 128 cho kết quả cao nhất với độ chính xác accuracy là 81.52%
- Với Epoch= 10 và Batch size = 64 cho kết quả thấp hơn với độ chính xác accuracy là 74.25%
- Với Epoch= 5 và Batch size = 32 cho kết quả thấp nhất với độ chính xác accuracy là 71.16%
- Chênh lệch độ chính xác accuracy giữa kết quả cao nhất và thấp nhất là *10.36%*
- Độ chính xác cao nhất với cặp tham số Epoch= 20 và Batch size = 128

tập trung vào nhãn ***"Gợi ý"*** với độ đo trung bình điều hòa F1 là *81.45% ,* nhãn " ***Không gợi ý***" với độ đo trung bình điều hòa F1 là 83.04%.

➢ Với mô hình **CNN**, ta có thể thấy được kết quả khả quan nhất nếu sử dụng với cặp tham số Epoch= 20 và Batch size = 128

❖ ***Kết quả với mô hình RNN:***

Bảng 3.8: Kết quả sử dụng mô hình RNN

RNN									
Epoch	5			10			20		
Batch size	32			64			128		
Accuracy %	69.46			71.28			76.81		
Độ đo	***Pre%***	***Re%***	***F1%***	***Pre%***	***Re%***	***F1%***	***Pre%***	***Re%***	***F1%***
Gợi ý	69.49	60.67	64.81	80.76	67.02	73.03	82.35	73.68	77.81
Không gợi ý	54.6	89.0	67.7	76.92	66.56	71.42	72.72	86.02	78.88

Dựa vào bảng kết quả 3.8 ta có thể thấy:

- Với Epoch= 20 và Batch size = 128 cho kết quả cao nhất với độ chính xác accuracy là 76.81%
- Với Epoch= 10 và Batch size = 64 cho kết quả thấp hơn với độ chính xác accuracy là 71.28%
- Với Epoch= 5 và Batch size = 32 cho kết quả thấp nhất với độ chính xác accuracy là 69.46%

- Chênh lệch độ chính xác accuracy giữa kết quả cao nhất và thấp nhất là *7.35%*
- Độ chính xác cao nhất với cặp tham số Epoch= 20 và Batch size = 128

tập trung vào nhãn ***"Gợi ý"*** với độ đo trung bình điều hòa F1 là *77.81%* , nhãn " ***Không gợi ý***" với độ đo trung bình điều hòa F1 là 78.88%.

➢ Với mô hình **RNN**, ta có thể thấy được kết quả khả quan nhất nếu sử dụng với cặp tham số Epoch= 20 và Batch size = 128

❖ ***Kết quả với mô hình LSTM:***

Bảng 3.9: Kết quả sử dụng mô hình LSTM

LSTM									
Epoch	5			10			20		
Batch size	32			64			128		
Accuracy %	72.42			75.07			83.26		
Độ đo	***Pre%***	***Re%***	***F1%***	***Pre%***	***Re%***	***F1%***	***Pre%***	***Re%***	***F1%***
Gợi ý	68.18	80.03	73.61	72.72	86.02	78.81	75.93	89.70	82.29
Không gợi ý	66.29	83.09	73.75	72.47	88.76	79.76	83.57	86.22	84.87

Dựa vào bảng kết quả 3.9 ta có thể thấy:

- Với Epoch= 20 và Batch size = 128 cho kết quả cao nhất với độ chính xác accuracy là 83.26%
- Với Epoch= 10 và Batch size = 64 cho kết quả thấp hơn với độ chính xác accuracy là 75.07%
- Với Epoch= 5 và Batch size = 32 cho kết quả thấp nhất với độ chính xác accuracy là 72.42%
- Chênh lệch độ chính xác accuracy giữa kết quả cao nhất và thấp nhất là *10.84%*
- Độ chính xác cao nhất với cặp tham số Epoch= 20 và Batch size = 128

tập trung vào nhãn ***"Gợi ý"*** với độ đo trung bình điều hòa F1 là *82.29%* , nhãn " ***Không gợi ý***" với độ đo trung bình điều hòa F1 là 84.87%

Với mô hình **LSTM**, ta có thể thấy được kết quả khả quan nhất nếu sử dụng với cặp tham số Epoch= 20 và Batch size = 128

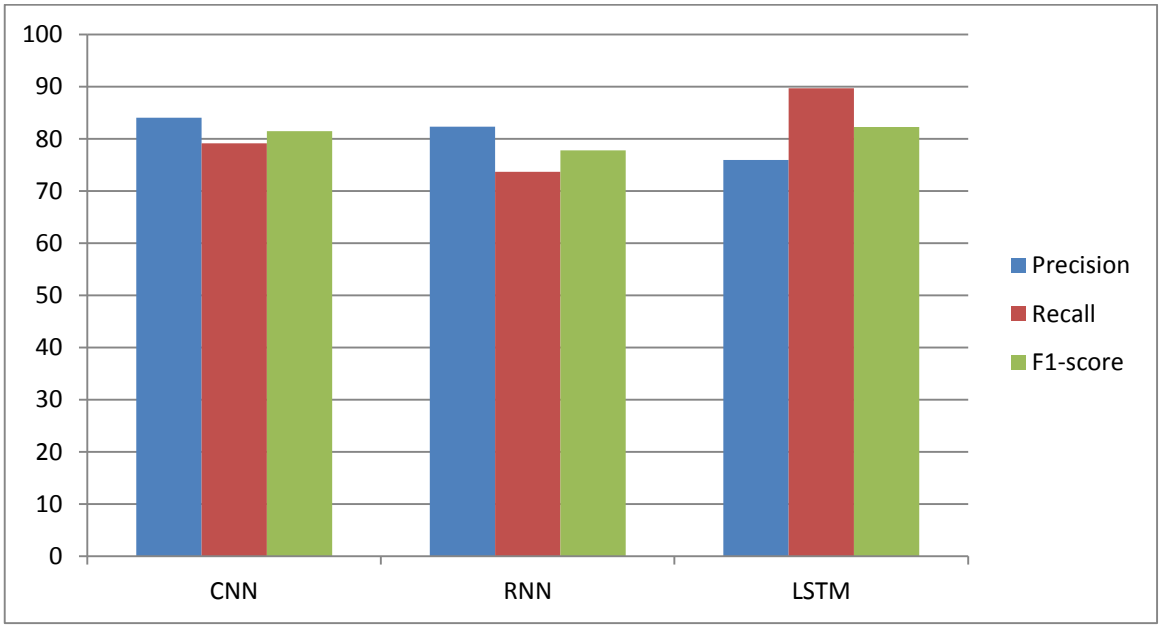
b, So sánh kết quả thực nghiệm giữa ba mô hình CNN,RNN,LSTM:

Thực hiện lấy kết quả tốt nhất giữa cặp tham số Epoch= 20 và Batch size = 128 của các mô hình đã cho ra kết quả để thực hiện so sánh được biểu diễn qua bảng sau:

Bảng 3.10: Kết quả so sánh giữa các mô hình

Mô hình	**CNN**			**RNN**			**LSTM**		
Accuracy %	81.52			76.81			83.26		
Độ đo	***Pre%***	***Re%***	***F1%***	***Pre%***	***Re%***	***F1%***	***Pre%***	***Re%***	***F1%***
Gợi ý	84.07	79.16	81.45	82.35	73.68	77.81	75.93	89.70	82.29
Không gợi ý	88.91	82.42	83.04	72.72	86.02	78.88	83.57	86.22	84.87

Sau khi tiến hành thực nghiệm cho kết quả giữa thuật toán CNN,RNN và LSTM tôi thực hiện so sánh kết quả của các phương pháp theo từng nhãn *" Gợi ý", "Không gợi ý"* được biểu diễn bằng các biểu đồ sau:



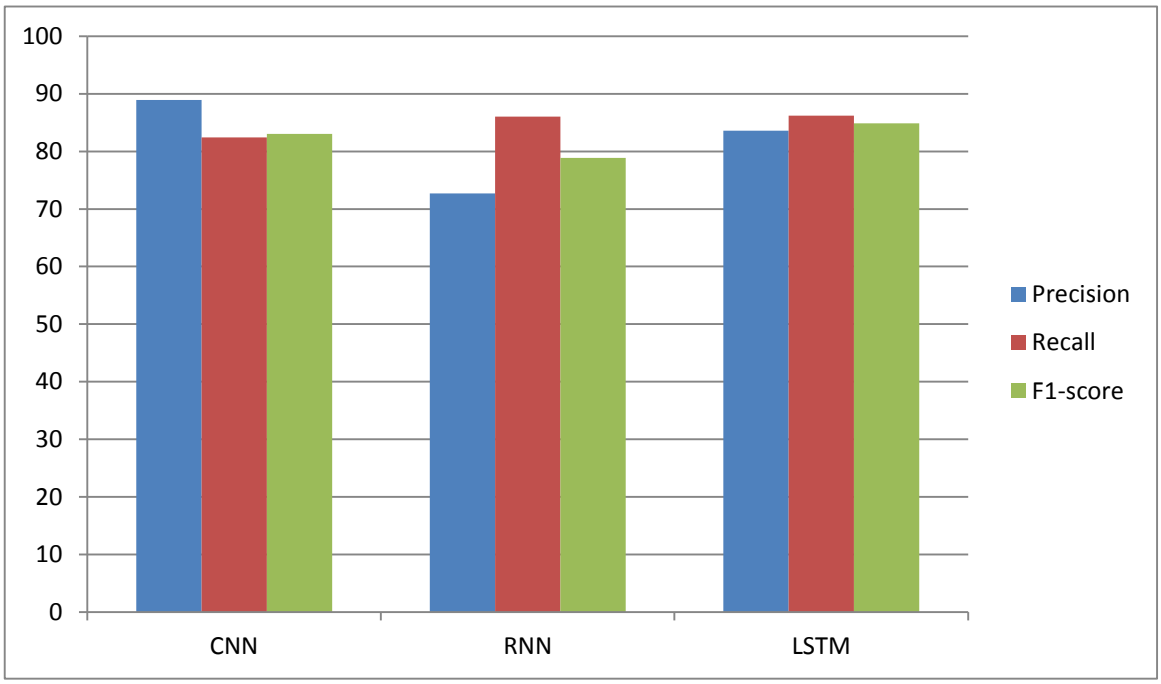
Hình 3.7: Biểu đồ so sánh giữa mô hình CNN, RNN, LSTM với nhãn "Gợi ý"

Dựa vào kết quả bảng 3.10 và hình 3.7 với nhãn " ***Gợi ý***" ta thấy :

- Thuật toán mô hình LSTM cho kết quả cao nhất với độ chính xác F1 là 82.29 %
- Thuật toán mô hình CNN cho kết quả thấp hơn với độ chính xác F1 là 81.45%
- Thuật toán mô hình RNN cho kết quả thấp hơn với độ chính xác F1 là 77.81%

- Độ chênh lệch F1 giữa kết quả cao nhất và thấp nhất là: 4,48%

➢ Với nhãn " ***Gợi ý***" ta có thể thấy kết quả khả quan nếu sử dụng mô hình LSTM

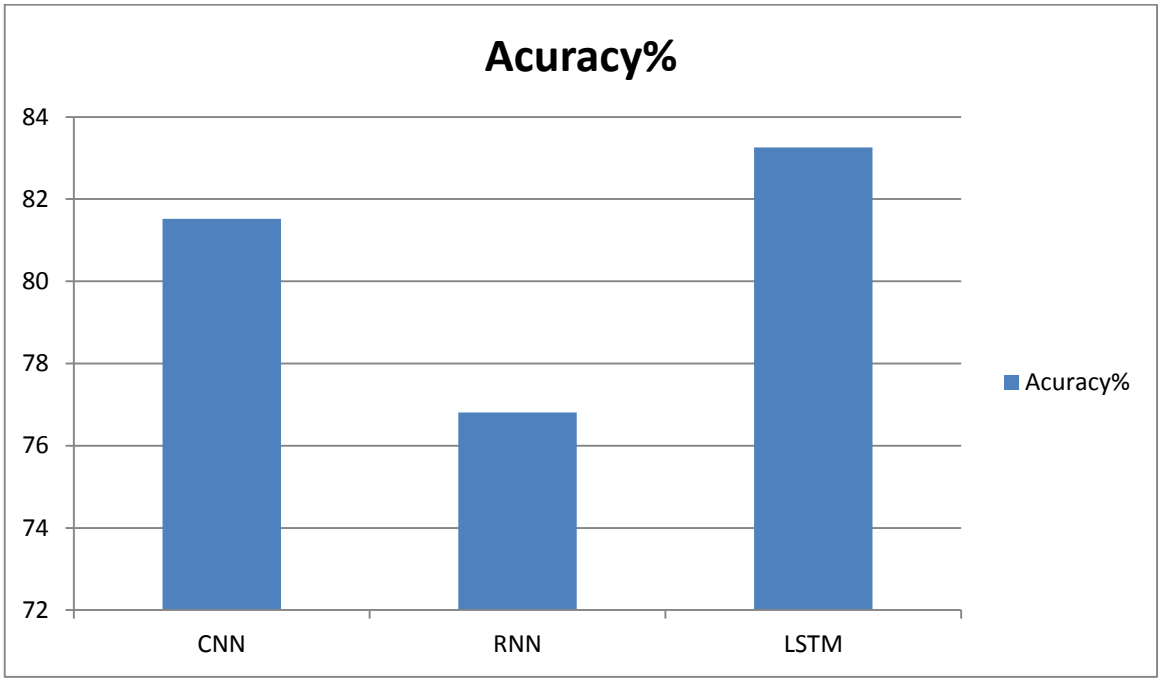


Hình 3.8: Biểu đồ so sánh mô hình CNN, RNN, LSTM với nhãn "Không gợi ý"

Dựa vào kết quả bảng 3.10 và hình 3.8 với nhãn " **Không *gợi ý***" ta thấy :

- Thuật toán mô hình LSTM cho kết quả cao nhất với độ chính xác F1 là 84.87 %
- Thuật toán mô hình CNN cho kết quả thấp hơn với độ chính xác F1 là 83.04%
- Thuật toán mô hình RNN cho kết quả thấp hơn với độ chính xác F1 là 78.88%
- Độ chênh lệch F1 giữa kết quả cao nhất và thấp nhất là: 5.99%

➢ Với nhãn " **Không *gợi ý***" ta có thể thấy kết quả khả quan nếu sử dụng mô hình LSTM



Hình 3.9: Biểu đồ so sánh độ chính xác của các mô hình

Từ bảng kết quả hình 3.9 và bảng 3.10, tôi thấy rằng :

- Mô hình LSTM có kết quả chính xác acuracy cao nhất là 83.26%,
- Mô hình CNN có kết quả chính xác acuracy thấp hơn 81.52%
- Mô hình RNN có kết quả chính xác acuracy thấp nhất 76.81%
- Chênh lệch độ chính xác accuracy giữa kết quả cao nhất và thấp nhất là 6.45%.

Điều đó chứng tỏ, độ chệnh lệch giữa các mô hình là không quá cao. Ta có thể thấy độ chính xác dù chênh lêch kết quả giữa hai mô hình không nhiều nhưng phương pháp cũng đã giúp cải thiện độ chính xác của việc phân loại

3.6. Nhận xét và đánh giá

Dựa vào các số liệu trên, 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 CNN, RNN để lựa chọn mô hình áp dụng cho đề tài " ***Phát hiện câu chứa gợi ý trên diễn đàn trực tuyến sử dụng mạng Nơ-Ron"***, tôi đề xuất và đánh giá cao mô hình LSTM hơn cả. Bên cạnh đó, các số liệu trung bình cũng như độ chênh lệch độ chính xác của mô hình LSTM cho kết quả khả quan nhất.

Tóm lại các mô hình mạng neural CNN và RNN, LSTM cho thấy một cách nhìn mới trong việc phân loại câu văn bản nói riêng và xử lý ngôn ngữ tự nhiên nói chung,

bằng cách sử dụng học chuyên sâu và kết hợp mô hình mạng neural. Luận văn cũng đã đưa ra các nhận xét, đánh giá và so sánh các mô hình, các bộ phân lớp, từ đó đưa ra được một mô hình tốt nhất trong việc giải quyết bài toán phân loại câu chưa gợi ý người dùng trên diễn đàn trực tuyến đã nêu.

KẾT LUẬN

Xử Lý Ngôn Ngữ Tự Nhiên nói chung và đặc biệt là phân loại câu chứa gợi ý người dùng nói riêng ngày càng đóng vai trò quan trọng trong các hoạt động thương mại, mua bán,du lịch… hiện nay.Trong luận văn này, chúng tôi tiến hành nghiên cứu phương pháp nhằm cải thiện độ chính xác cho bài toán phân loại câu văn bản, cụ thể là cải thiện độ chính xác cho bài toán phân loại câu chứa gợi ý trên diễn đàn trực tuyến. Bài toán này được xác định là một bài toán có độ phức tạp và có nhiều ứng dụng trong thực tế. Phương pháp giải quyết của luận văn tập trung vào việc nâng cao

độ chính xác trong việc phân loại được các ý định của người dùng thông qua diễn đàn trực tuyến. Bằng việc sử dụng mô hình phân lớp quen thuộc CNN và RNN,LSTM cùng với tập dữ liệu thu được từ diễn đàn trực tuyến, luận văn đã đưa ra số phương pháp để giải quyết cho bài toán đề ra. Quá trình thực nghiệm đạt được kết quả khả quan, cho thấy tính đúng đắn của việc lựa chọn cũng như kết hợp các phương pháp, đồng thời hứa hẹn nhiều tiềm năng phát triển hoàn thiện.

Nhìn chung, luận văn đã đạt được một số kết quả như:

- Trình bày một cách khái quát, tổng quan nhất và nêu lên ý nghĩa, vai trò quan trọng của bài toán phân loại câu chứa gợi ý người dùng trên diễn đàn trực tuyến.
- Nghiên cứu các mô hình khác nhau cho bài toán phân loại câu chứa gợi ý.
- Nghiên cứu và làm thực nghiệm với các thuật toán học máy khác nhau.
- So sánh và phân tích các kết quả thực nghiệm, đưa ra kết quả tốt nhất.

Luận văn vẫn còn một số hạn chế như:

- Nghiên cứu dựa trên số lượng dữ liệu còn ít và chưa đầy đủ.
- Kết quả thực nghiệm đạt được vẫn chưa thực sự cao
- Chỉ thử nghiệm đối với tập dữ liệu bằng tiếng anh

Về hướng phát triển tương lai, chúng tôi sẽ tiến hành thu thập và phát triển trên một tập dữ liệu lớn hơn và dựa trên nhiều đặc trưng hơn để góp phần cải thiện khả năng phân loại. Bên cạnh đó chúng tôi cũng sẽ nghiên cứu và thử nghiệm với một số thuật toán khác để tìm ra thuật toán phù hợp nhất với bài toán phân loại câu chứa gợi ý người dùng trực tuyến bằng tiếng Việt. Khắc phục lỗi trong quá trình xử lý để nâng cao kết quả thực nghiệm.

DANH MỤC 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] Ngo Xuan Bach, Tu Minh Phuong, “Leveraging User Ratings for Resource- Poor Sentiment Classification”, In Proceedings of the 19th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES), Procedia Computer Science, pp. 322–331, 2015.

[3] Nguyễn Minh Thành, *Phân loại văn bản*, Luận văn môn học Xử lý ngôn ngữ tự nhiên, Đại học quốc gia Thành phố Hồ Chí Minh, 01/2011

[4] Từ Minh Phương. Giáo trình nhập môn trí tuệ nhân tạo. Nhà xuất bản Thông tin và Truyền thông, 2016

Tiếng Anh

[5] https://github.com/Semeval2019Task9/Subtask-A

[6] Zhang Y, Wallace B. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. arXiv preprint arXiv:151003820. 2015; PMID: 463165.

[7] Ahmed Husseini Orabi, Prasadith Buddhitha, Mahmoud Husseini Orabi, Diana Inkpen, "*Deep Learning for Depression Detection of Twitter Users*", 2018.

[8] Awais Athar, Simone Teufel, *"Detection of Implicit Citations for Sentiment Detection"*, 2012.

[9] 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.

[10] Maria Karanasou, Christos Doulkeridis, Maria Halkidi, *"DsUniPi: An SVM-based Approach for Sentiment Analysis of Figurative Language on Twitter"*, 2015.

[11] Peng Chen, Zhongqian Sun Lidong Bing, Wei Yang, *"Recurrent Attention Network on Memory for Aspect Sentiment Analysis"*, 2017.

[12] Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, Bo Xu, *"Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling",* 2016.

[13] Kröll, M., & Strohmaier, M. (2009, September). Analyzing human intentions in natural language text. In Proceedings of the fifth international conference on Knowledge capture (pp. 197-198). ACM

[14] Kim Y. Convolutional Neural Networks for Sentence Classification. 2014

[15] Zhang Y, Wallace B. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. arXiv preprint

arXiv:151003820. 2015; PMID: 463165

DANH MỤC WEBSITE THAM KHẢO

[16] Wikipedia: http://www.wikipedia.org

[17] Google : https://www.google.com

[18] Danh sách stop word tiêu chuẩn http://www.ranks.nl/stopwords

[19] https://d2l.ai/chapter_convolutional-neural-networks/index.html

[20] http://karpathy.github.io/2015/05/21/rnn-effectiveness/

[21] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[22] https://www.python.org/

[23] https://www.tensorflow.org

[24] https://cs231n.github.io/neural-networks-1/

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

[26] https://machinelearningcoban.com/2017/03/04/overfitting/

[27] http://www.joshuakim.io/understanding-how-convolutional-neural-network-cnn-perform-text-classification-with-word-embeddings/