ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

VÕ MINH TÂM

KHÓA LUẬN TỐT NGHIỆP

ĐÁNH GIÁ ĐỘ HÀI HƯỚC TRÊN CÂU TIÊU ĐỀ TIN TỨC ĐÃ CHỈNH SỬA

CỬ NHÂN NGÀNH KHOA HỌC MÁY TÍNH

TP. HỒ CHÍ MINH, 2020

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

VÕ MINH TÂM - 16521798

KHÓA LUẬN TỐT NGHIỆP

ĐÁNH GIÁ ĐỘ HÀI HƯỚC TRÊN CÂU TIÊU ĐỀ TIN TỨC ĐÃ CHỈNH SỬA

CỬ NHÂN NGÀNH KHOA HỌC MÁY TÍNH

GIẢNG VIÊN HƯỚNG DẪN

TS NGUYỄN LƯU THÙY NGÂN

ThS NGUYỄN VĂN KIỆT

TP. HỒ CHÍ MINH, 2020

DANH SÁCH HỘI ĐỒNG BẢO VỆ KHÓA LUẬN

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số 523/QĐ-ĐHCNTT ngày 25 tháng 08 năm 2020 của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

1. PGS. TS. Quản Thành Thơ – Chủ tịch.
2. ThS. Nguyễn Trọng Chỉnh – Thư ký.
3. TS. Huỳnh Ngọc Tín – Ủy viên.

LỜI CẢM ƠN

Lời đầu tiên, tôi xin gửi lời cảm ơn chân thành và bày tỏ lòng biết ơn sâu sắc đến Cô Nguyễn Lưu Thùy Ngân và Thầy Nguyễn Văn Kiệt, những người đã trực tiếp hướng dẫn, dạy bảo tôi trong suốt quá trình tham gia vào nhóm nghiên cứu và thực hiện khóa luận. Tôi cũng xin chân thành cảm ơn Thầy Dương Ngọc Hảo, anh Đặng Văn Thìn, anh Nguyễn Đức Vũ, bạn Nguyễn Vũ Anh Khoa, bạn Huỳnh Văn Tín, bạn Huỳnh Văn Tú và toàn thể thầy, cô, anh, chị và các bạn trong nhóm nghiên cứu đã luôn hỗ trợ tôi hoàn thành khóa luận này. Tôi cũng chân thành cảm ơn Thầy Huỳnh Ngọc Tín đã có những góp ý quý báu trong quá trình phản biện giúp tôi hoàn thiện khóa luận hơn.

Để có vốn kiến thức nền tảng cho việc thực hiện khóa luận, tôi xin chân thành cảm ơn quý thầy, cô trường Đại học Công nghệ Thông tin – ĐHQG HCM đã tận tình dạy bảo, cho tôi nhiều kiến thức hữu ích để có thể áp dụng vào khóa luận này cũng như sử dụng trong cuộc sống. Tôi cũng không quên cảm ơn các tác giả của các công trình mà tôi trích dẫn, nhờ vào những nghiên cứu của họ mà tôi mới có thể hoàn thành tốt luận văn này. Đặc biệt, tôi xin gửi lời cảm ơn chân thành đến nghiên cứu sinh Nabil Hossain của trường Đại học Rochester, New York, Hoa Kỳ, người đã rất nhiệt tình giải đáp những thắc mắc của tôi về bộ dữ liệu được sử dụng.

Tôi đã vận dụng những kiến thức tích lũy được từ nhà trường, thầy cô và bạn bè, kết hợp với việc tự học và nghiên cứu những kiến thức mới để hoàn thành khóa luận này. Tuy nhiên, do kiến thức của bản thân còn hạn chế nên trong quá trình thực hiện khó tránh khỏi thiếu sót. Vì vậy, tôi rất mong nhận được những đóng góp mang tính xây dựng từ quý thầy, cô và các bạn sinh viên.

Xin chân thành cảm ơn!

Sinh viên

Võ Minh Tâm

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc Lập - Tự Do - Hạnh Phúc

ĐỀ CƯƠNG CHI TIẾT

TÊN ĐỀ TÀI: ĐÁNH GIÁ ĐỘ HÀI HƯỚC TRÊN CÂU TIÊU ĐỀ TIN TỨC ĐÃ CHỈNH SỬA

TÊN ĐỀ TÀI TIẾNG ANH: ASSESSING THE FUNNINESS OF EDITED NEWS HEADLINES

Cán bộ hướng dẫn: TS. Nguyễn Lưu Thùy Ngân, ThS. Nguyễn Văn Kiệt

Thời gian thực hiện: Từ ngày 09/03/2020 đến ngày 11/07/2020.

Sinh viên thực hiện:

Võ Minh Tâm **Lớp: KHTN2016**

Email: 16521798@gm.uit.edu.vn **Điện thoại: 0962840078**

Nội dung đề tài:

1. **Giới thiệu đề tài:**
   - Đề tài là bài toán thứ 7 của cuộc thi SemEval-2020. Có hai bài toán con là hồi quy và so sánh. Khóa luận này sẽ tập trung giải quyết bài toán hồi quy: đánh giá độ hài hước trên câu tiêu đề tin tức tiếng Anh đã được áp dụng một chỉnh sửa nhỏ (microedit).
   - Bộ dữ liệu: **Humicroedit** được cung cấp bởi ban tổ chức cuộc thi.
2. **Mô tả bài toán:**
   - Đầu vào: cho hai câu tiêu đề tin tức tiếng Anh gồm: **một câu tiêu đề gốc** và **một câu đã chỉnh sửa** từ câu gốc. Có thể sử dụng một câu hoặc cả hai

câu.

- Đầu ra: một số thực trong đoạn [0, 3] biểu thị độ hài hước trên **câu tiêu đề đã chỉnh sửa**.

Ví dụ mẫu dữ liệu:

Mã số	**Câu tiêu đề gốc**	**Từ thay thế**	**Điểm thành phần**	**Điểm trung bình**
76	In an apparent first , Iran and Israel **<engage/>** each other militarily	**slap**	20000	0.4
827	New Orleans takes down 1st of 4 Confederate **<statues/>**	**birds**	00000	0
119	No more ' monkey business ' ? Trump touts big **<jobs/>** number as proof of improvement	**monkey**	22100	1
8877	Hillary Clinton Needs to **<Move/>** On	**party**	33222	2.4

3. Mục tiêu:

Khóa luận đặt ra các mục tiêu cụ thể như sau:

- Tìm hiểu cách tạo ra sự hài hước thông qua một "chỉnh sửa nhỏ" (microedit) từ bộ dữ liệu Humicroedit.
- Áp dụng các mô hình học sâu kết hợp từ các mô hình đã có như LSTM, GRU, CNN, Bi-RNN nhằm tạo ra mô hình cho kết quả tốt hơn mô hình mẫu (baseline) của ban tổ chức và nằm trong nhóm 30 đội có kết quả tốt

nhất.

- Áp dụng và đánh giá tác động của các phương pháp tiền xử lý dữ liệu và các kỹ thuật chuẩn hóa, hiệu chỉnh siêu tham số lên mô hình.
- Xây dựng chương trình demo.

4. Phạm vi nghiên cứu:

- Các khái niệm, nhận định liên quan đến sự hài hước.
- Các câu tiêu đề tin tức bằng tiếng Anh trong bộ dữ liệu Humicroedit.

5. Đối tượng:

- Các cơ sở lý thuyết về sự hài hước áp dụng vào các câu tiêu đề tin tức tiếng Anh thông qua một chỉnh sửa nhỏ.
- Các phương pháp học sâu LSTM, GRU, Bi-RNN, CNN và các mô hình kết hợp cho bài toán hồi quy trên văn bản.

6. Phương pháp thực hiện:

- Chuẩn hóa, xử lý dữ liệu trong bộ dữ liệu Humicroedit.
- Cài đặt các mô hình học sâu cơ bản như GRU, LSTM, BiGRU, BiLSTM, Transformer...
- Đề xuất mô hình đánh giá mức độ hài hước cho câu tiêu đề thông qua việc kết hợp các mô hình học sâu đã tìm hiểu là RNN và CNN.
- Áp dụng một số kỹ thuật tối ưu nâng cao hiệu suất của mô hình đã đề xuất.

7. Kết quả mong đợi:

Mô hình đề xuất cho kết quả tốt hơn mô hình baseline và nằm trong top 30.

Kế hoạch thực hiện:

Thời gian	Công việc thực hiện

09/03 – 09/04	Tìm hiểu đề tài, bộ dữ liệu và các công trình liên quan
10/04 – 10/05	Tìm hiểu các phương pháp xử lý dữ liệu, các mô hình học sâu LSTM, GRU, Bi-RNN, CNN và các phương pháp kết hợp các mô hình này.
11/05 – 11/06	Tiến hành cài đặt, thử nghiệm các mô hình đã tìm hiểu và đề xuất trên bộ dữ liệu được cung cấp.
12/06 – 30/06	Cải tiến mô hình, áp dụng các kỹ thuật tối ưu để thiện kết quả.
30/06 – về sau	Viết báo cáo.

Xác nhận của CBHD 1 (Ký tên và ghi rõ họ tên) TS. Nguyễn Lưu Thùy Ngân **Xác nhận của CBHD 2** (Ký tên và ghi rõ họ tên) ThS. Nguyễn Văn Kiệt	**TP. HCM, ngày 24 tháng 07 năm 2020** **Sinh viên** (Ký tên và ghi rõ họ tên) Võ Minh Tâm

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Tên đầy đủ
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
BiLSTM	Bidirectional Long Short-Term Memory
GRU	Gated Recurrent Unit
BiGRU	Bidirectional Gated Recurrent Unit
CNN	Convolutional Neural Network
Word2vec	Word to Vector
GloVe	Global Vectors for Word Prepresentation
SemEval	International Workshop on Semantic Evaluations
CBOW	Continous Bag Of Words
Vector	Véc-tơ

TÓM TẮT KHÓA LUẬN

Khóa luận "Đánh giá độ hài hước trên câu tiêu đề tin tức đã chỉnh sửa" được lấy cảm hứng từ một bài toán của cuộc thi SemEval-2020. Kết quả sau cùng là mô hình kết hợp các kỹ thuật học sâu CNN-BiLSTM-BiLSTM cho kết quả tốt nhất trong số các mô hình được sử dụng. Mô hình này có kết quả được xếp hạng thứ 18 so với kết quả chính thức từ cuộc thi và thứ 13 nếu tính trong phiên đánh giá phụ.

Khóa luận này gồm sáu chương, phần tài liệu tham khảo và phụ lục, chi tiết được mô tả như sau:

- Chương 1: TỔNG QUAN ĐỀ TÀI. Chương này giới thiệu về đề tài (gồm phần đặt vấn đề, mô tả bài toán, mục tiêu, phạm vi, đối tượng nghiên cứu) và nêu lên giá trị của đề tài.
- Chương 2: DỮ LIỆU. Mô tả chi tiết về bộ dữ liệu Humicroedit như mục tiêu xây dựng, nguồn thu thập, quá trình chỉnh sửa và gán nhãn, quá trình giám sát và đảm bảo chất lượng. Chương này cũng phân tích một số tác nhân ảnh hưởng đến sự hài hước, sau đó là quá trình xây dựng lại bộ dữ liệu cho việc phân loại câu hài hước.
- Chương 3: CƠ SỞ LÝ THUYẾT. Chương này trình bày kiến thức về các phương pháp biểu diễn từ, kiến trúc của các mô hình học sâu được áp dụng trong khóa luận, một số kỹ thuật đào tạo mạng học sâu và phương pháp đánh giá mô hình.
- Chương 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ. Chương 5 sẽ mô tả chi tiết quá trình tiền xử lý dữ liệu, mô hình được đề xuất và nhận xét về hiệu suất của các mô hình này.
- Chương 5: XÂY DỰNG ỨNG DỤNG DEMO. Một ứng dụng trên nền web làm demo cho mô hình đã được xây dựng sẽ được trình bày ở chương này.
- Chương 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN. Phần này sẽ đánh giá tổng quan đề tài và đề xuất hướng phát triển trong tương lai.

- TÀI LIỆU THAM KHẢO gồm các tài liệu tiếng Anh và tiếng Việt được sử dụng trong khóa luận.
- PHỤ LỤC: trình bày danh sách stopword và chi tiết kết quả thí nghiệm của các mô hình.

Chương 1. TỔNG QUAN ĐỀ TÀI

Chương này gồm các phần giới thiệu tổng quan về đề tài, mô tả bài toán, trình bày mục tiêu, phạm vi, đối tượng nghiên cứu, một số nghiên cứu có liên quan cũng như giá trị về mặt nghiên cứu và ứng dụng mà đề tài mang lại.

1.1. Giới thiệu đề tài

Phát hiện và tạo sự hài hước trong văn bản tiếp tục là bài toán mang tính thách thức trong lĩnh vực xử lý ngôn ngữ tự nhiên. Nếu các nghiên cứu về việc tự động nhận biết sự hài hước có nhiều bước tiến thì hướng nghiên cứu tạo và đánh giá sự hài hước trong văn bản lại ít hơn hẳn. Hơn nữa, phần lớn các bộ dữ liệu và cách tiếp cận cho việc phát hiện sự hài hước hiện nay đều được quy về dạng phân loại nhị phân: hài hước hoặc không hài hước. Tuy nhiên, điều này chưa hợp lý, vì trong thực tế tính hài hước được thể hiện ở nhiều mức độ khác nhau. Do đó, cuộc thi **SemEval-2020**[1] đã giới thiệu bài toán **"Đánh giá độ hài hước trên câu tiêu đề tin tức đã chỉnh sửa"** dựa trên bộ dữ liệu **Humicroedit** được giới thiệu bởi *Nabil* và các cộng sự [1] tại hội **nghị NAACL[2] 2019.** Bài toán được đưa ra với mục đích thúc đẩy các nghiên cứu chuyên sâu trong việc hiểu ngôn ngữ - vốn rất quan trọng trong lĩnh vực trí tuệ nhân tạo. Đây cũng là lý do chính để sinh viên chọn thực hiện đề tài này.

1.2. Mô tả bài toán

Cuộc thi SemEval-2020 đưa ra hai bài toán con cho bài toán *"Đánh giá độ hài hước trên câu tiêu đề tin tức đã chỉnh sửa"* là **hồi quy** và **so sánh**. Khóa luận này sẽ tập trung giải quyết bài toán hồi quy. Đối với bài toán so sánh, có thể tận dụng lại mô hình đã xây dựng từ bài toán hồi quy hoặc xây dựng mô hình mới. Mô tả cụ thể cho bài toán **hồi quy** như sau:

[1] Link: http://alt.qcri.org/semeval2020/index.php?id=tasks [Online] Truy cập: 27/08/2020.
[2] North American Chapter of the Association for Computational Linguistics (NAACL) là thành viên của hội nghị hàng đầu về lĩnh vực ngôn ngữ học tính toán - Association for Computational Linguistics (ACL).

- Đầu vào: cho ***câu tiêu đề tin tức gốc*** và ***câu tiêu đề đã được chỉnh sửa***. Có thể sử dụng một hoặc cả hai câu để xây dựng mô hình.
- Đầu ra: một số thực trong đoạn [0, 3] dự đoán độ hài hước ***trên câu tiêu đề đã được chỉnh sửa***. Độ hài hước tương ứng độ lớn điểm số.

1.3. Mục tiêu nghiên cứu

Nhằm giúp máy tính tự động nhận biết và đánh giá mức độ hài hước trong văn bản nói chung, khóa luận này sẽ cụ thể hóa các mục tiêu sau:

- Tìm hiểu cách tạo ra sự hài hước trong văn bản thông qua một "chỉnh sửa nhỏ" (microedit) từ bộ dữ liệu Humicroedit.
- Tìm hiểu, cải tiến các phương pháp có liên quan, đồng thời áp dụng thêm các mô hình học sâu như RNN, LSTM, GRU, Bidirectional-RNN, CNN, Transformer để giải quyết bài toán.
- Đề xuất được mô hình học sâu cho kết quả tốt hơn mô hình mẫu (baseline) mà ban tổ chức đã sử dụng, đồng thời kỳ vọng mô hình này giúp sinh viên lọt vào nhóm 30 đội có kết quả tốt nhất dựa theo kết quả chính thức được công nhận.
- Đánh giá tác động của các phương pháp tiền xử lý dữ liệu và các kỹ thuật chuẩn hóa, hiệu chỉnh đến hiệu suất của mô hình.
- Xây dựng chương trình ứng dụng demo.

1.4. Phạm vi nghiên cứu

Các câu tiêu đề tin tức bằng tiếng Anh trong bộ dữ liệu Humicroedit.

1.5. Đối tượng nghiên cứu

- Các cơ sở lý thuyết về sự hài hước trong văn bản.
- Các mô hình nhúng từ (word embedding/representation).
- Các phương pháp tiền xử lý, chuẩn hóa dữ liệu.

- Các phương pháp học sâu RNN, LSTM, GRU, Bi-RNN, CNN.
- Mô hình Transformer với cơ chế chú ý (attention mechanism).

1.6. Nghiên cứu liên quan

Một số nghiên cứu liên quan cho bài toán hồi quy trên văn bản đều sử dụng các mô hình học sâu và một số kỹ thuật phổ biến hiện nay. *Zsolt* và *Trevor* [2] chỉ ra việc sử dụng mạng thần kinh tích chập (CNN) giúp trích xuất các đặc trưng n-grams kết hợp cùng các siêu dữ liệu (metadata) cho kết quả vượt trội hơn so với các mô hình tuyến tính truyền thống. *Neşat* và *Murat* [3] cũng áp dụng CNN cho việc rút trích đặc trưng kết hợp với việc sử dụng tập nhúng từ được xây dựng sẵn thay vì phụ thuộc vào từ vựng trên miền dữ liệu tài chính. *Stig*, *Taraka* và *Lilja* [4] cho thấy việc sử dụng GRU – một biến thể của mạng thần kinh hồi quy (RNN) kết hợp cơ chế chú ý (attention mechanism) mang lại kết quả tốt nhất khi áp dụng cho bài toán tự động chấm điểm các bài luận tiếng Na Uy.

Các nghiên cứu đã liệt kê ở trên giải quyết các bài toán hồi quy trên văn bản có đầu vào chỉ là một câu văn hoặc đoạn văn bản với các miền dữ liệu khác nhau. Trong khi đó, mục đích của bài toán mà cuộc thi đưa ra là đánh giá độ hài hước trên câu tiêu đề tin tức tiếng Anh đã được áp dụng một thay đổi nhỏ nhằm phát hiện điểm gây nên sự hài hước. Theo tìm hiểu của sinh viên, hiện nay chưa có công bố nào cho bài toán đánh giá độ hài hước trên văn bản. Thêm vào đó, việc tìm hiểu, áp dụng các kỹ thuật và mô hình mới, mang tính đột phá hay cải tiến các phương pháp sẵn có phù hợp với bài toán cũng là thách thức lớn đối với bản thân sinh viên.

1.7. Giá trị của đề tài

Đề tài sẽ là động lực sẽ là tiền đề thúc đẩy các nghiên cứu và ứng dụng như: xếp hạng hàng loạt các chỉnh sửa trên cùng câu tiêu đề theo mức độ hài hước, tạo các tiêu đề vui nhộn, đề xuất tiêu đề hài hước được cá nhân hóa cho từng độc giả... Đề tài còn có thể mở rộng sang các khía cạnh khác của ngôn ngữ như mức độ quấy rối, yêu thích hoặc chê bai. Tất cả những điều này cho cộng đồng nghiên cứu và đề xuất

các công cụ xử lý ngôn ngữ không chỉ mạnh mẽ trong nhận dạng mẫu mà quan trọng hơn là khả năng hiểu và lý luận ngữ nghĩa theo chiều sâu.

Chương 2. BỘ DỮ LIỆU

Chương này giới thiệu về bộ dữ liệu được sử dụng cho bài toán đánh giá độ hài hước trên câu tiêu đề tin tức đã chỉnh sửa, gồm các nội dung: giới thiệu tổng quan về bộ dữ liệu, mục tiêu xây dựng, nguồn dữ liệu thu thập, quá trình xử lý và gán nhãn, quản lý chất lượng bộ dữ liệu và phân tích các yếu tố tạo nên sự hài hước.

2.1. Giới thiệu bộ dữ liệu Humicroedit

Các bộ dữ liệu nghiên cứu sự hài hước trong văn bản chủ yếu dùng để phân loại một câu là hài hước hoặc không hài hước [5], [6]. Việc xây dựng một bộ dữ liệu có tính ứng dụng cao trong nghiên cứu và thực tiễn là khá khó khăn, đặc biệt đối với một lĩnh vực cụ thể như sự hài hước chẳng hạn. Bộ dữ liệu đòi hỏi sự gán nhãn của con người theo một số ràng buộc nhất định để đảm bảo chất lượng bộ dữ liệu là tốt nhất và có thể sử dụng rộng rãi. *Nabil* và các cộng sự đã giới thiệu một bộ dữ liệu hoàn toàn mới phục vụ cho các nghiên cứu tính toán sự hài hước trong văn bản mang tên **Humicroedit** [1].

Bộ dữ liệu Humicroedit là bộ dữ liệu đầu tiên có thể đánh giá sự hài hước trong văn bản ở mức độ sâu hơn thông qua một chỉnh sửa nhỏ. Bảng 2.1 là một số ví dụ về dữ liệu được sử dụng. Mỗi câu tiêu đề có một mã số, trong đó câu tiêu đề gốc có từ được đánh dấu, và có một từ thay thế được chỉ định sẵn. Mỗi câu sẽ được ít nhất năm giám khảo chấm điểm, độ hài hước cho câu tiêu đề đã chỉnh sửa là trung bình các điểm thành phần của các giám khảo.

Mã số	Câu tiêu đề gốc	Từ thay thế	Điểm thành phần	Điểm trung bình
76	In an apparent first , Iran and Israel <**engage/**> each other militarily	**slap**	20000	0.4
827	New Orleans takes down 1st of 4 Confederate <**statues/**>	**birds**	00000	0
119	No more ' monkey business ' ? Trump touts big <**jobs/**> number as proof of improvement	**monkey**	22100	1
8877	Hillary Clinton Needs to <**Move/**> On	**party**	33222	2.4
6346	Basic income experiment <**receives/**> $ 5 million worth of bitcoin	**eats**	32110	1.4

Bảng 2.1: Ví dụ một số dòng dữ liệu trong bộ dữ liệu Humicroedit.

Đầu tiên, nhóm tác giả thu thập các câu tiêu đề tin tức từ trang thông tin trực tuyến Reddit (reddit.com). Sau đó, nhóm nghiên cứu đề ra tiêu chuẩn chỉnh sửa, gán nhãn và đánh giá chất lượng gán nhãn cũng như chất lượng chuyên gia gán nhãn đến từ Amazon Mechanical Turk (mturk.com). Kết quả thu được 15,095 câu tiêu đề tin tức đã được chỉnh sửa và độ hài hước trung bình tương ứng.

2.2. Mục tiêu xây dựng bộ dữ liệu

Humicroedit được các tác giả xây dựng nhằm mục đích tìm hiểu cách mà sự hài hước được tạo ra bằng cách áp dụng một chỉnh sửa nhỏ lên câu tiêu đề tin tức. Các tiêu đề tin tức đặc biệt phù hợp để nghiên cứu vì chúng mang nhiều thông tin nhưng chỉ với số lượng từ ít. Trong khi các tiêu đề có vẻ như bị giới hạn về ngữ cảnh,

chúng phải cung cấp đủ lượng thông tin cơ sở để độc giả có thể hiểu được. Bằng việc cho phép thực hiện một chỉnh sửa nhỏ, các tác giả có thể tập trung phân tích "điểm tiếp giáp" giữa câu tiêu đề bình thường và tiêu đề mang tính hài hước.

Bộ dữ liệu sẽ được áp dụng một sự chỉnh sửa nhỏ bằng cách thay thế một danh từ đơn hoặc một động từ cho một thực thể danh từ đơn hoặc động từ trong câu gốc. Tuy nhiên, quy tắc thay thế không cho phép:

- Thêm hoặc thay thế cả một cụm danh từ hoặc động từ, ngoại trừ việc thay thế một cụm danh từ là các thực thể (ví dụ như "One World", "Virtual Reality").
- Thay thế một token con trong một thực thể (ví dụ như thay thế từ "States" trong cụm "United States").

Quy tắc đặt ra như vậy để tránh việc chỉnh sửa lên các từ được xem là "parts-of-speech" (POS) vì các chỉnh sửa như thế này không cung cấp đủ sự đa dạng cho tính hài hước. Để nhận ra các thực thể có thể thay thế, nhóm tác giả đã sử dụng named entity recognition (NER) và POS tagging từ bộ công cụ Stanford CoreNLP [7]. Các tên thực thể được cho phép thay thế phải là những thực thể nổi tiếng, dựa theo Microsoft Knowledge Base[3]. Sự chỉnh sửa nhỏ này được gọi là "chỉnh sửa vi mô" (micro-edit). Chỉnh sửa vi mô tiếp cận sự thay đổi nhỏ nhất có thể tạo ra sự hài hước trong văn bản, từ đó cho phép các nhà nghiên cứu tập trung vào điểm gây nên sự hài hước.

2.3. Nguồn thu thập dữ liệu

Bộ dữ liệu được thu thập từ các tiêu đề tin tức đăng trên trang thông tin mạng xã hội nổi tiếng Reddit ở hai miền thông tin là "tin thế giới" (*r/worldnews*) và "tin chính trị" (*r/politics*) từ đầu năm 2017 đến giữa năm 2018 sử dụng Google Big Query[4].

[3] https://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing/ [Online] Truy cập: 27/08/2020.
[4] *https://cloud.google.com/bigquery/* [Online] Truy cập: 27/08/2020.

Nhóm tác giả giữ lại các câu tiêu đề từ 25 nguồn tin tức tiếng Anh, mỗi câu tiêu đề có độ dài tối thiểu 5 từ và tối đa 20 từ với tổng cộng 287,076 tiêu đề.

2.4. Gán nhãn

Các chuyên gia gán nhãn đến từ Mechanical Turk là những người sống ở Mỹ, có tỷ lệ đồng thuận HIT lớn hơn 97% và có hơn 10,000 HIT đã được chấp nhận. Để đảm bảo chất lượng, nhóm chuyên gia sẽ được đánh giá và chia thành hai nhóm: nhóm nhận diện sự hài hước trong câu chỉnh sửa và nhóm chỉnh sửa câu tiêu đề để tạo ra sự hài hước.

2.4.1. Đánh giá đội ngũ chuyên gia cho điểm

Nhóm tác giả xây dựng một số câu tiêu đề mẫu. Các chuyên gia cho điểm sự hài hước trên mỗi câu tiêu đề này theo thang điểm từ 0 đến 3 [8] như Bảng 2.2.

Mức độ hài hước	Điểm
Không hài hước	0
Hài hước nhẹ	1
Hài hước vừa phải	2
Hài hước	3

Bảng 2.2: Thang điểm hài hước được sử dụng trong bộ dữ liệu Humicroedit.

Các tác giả bộ dữ liệu dựa trên kết quả từ các chuyên gia gán nhãn, tính toán sự đồng thuận và cuối cùng chọn ra 150 chuyên gia đủ điều kiện, tương ứng 60% số lượng chuyên gia ban đầu.

2.4.2. Đánh giá đội ngũ chuyên gia thực hiện chỉnh sửa

Các chuyên gia chỉnh sửa này được tham gia một bài kiểm tra thực hiện chỉnh sửa trên một số câu tiêu đề. Các chuyên gia không được phép:

- Sử dụng kỹ thuật tạo sự hài hước tầm thường: thêm các từ ngữ thô tục, tiếng lóng, từ thô thiển hoặc các ngôn ngữ không chính thức.
- Ép/Nối nhiều từ thành một (như Housecat, JumpedOverWal).

Nhóm tác giả sau đó sử dụng một số chuyên gia cho điểm đã đào tạo trước đó để đánh giá mức độ phù hợp của đội ngũ chuyên gia chỉnh sửa. Kết quả cuối cùng chọn được 100 người chỉnh sửa chất lượng, tương ứng 57.5% số chuyên gia thực hiện chỉnh sửa ban đầu.

2.5. Bộ dữ liệu và quản lý chất lượng

Các tác giả đã chọn ngẫu nhiên 5170 tiêu đề tin tức trong bộ dữ liệu cuối cùng, sử dụng 3 chuyên gia chỉnh sửa và 5 chuyên gia cho điểm. Nhiều chỉnh sửa vi mô được áp dụng lên các tiêu đề này, theo đó mỗi câu sẽ được cả 3 chuyên gia chỉnh sửa và 5 chuyên gia cho điểm, sự chỉnh sửa vẫn tuân theo quy tắt của chỉnh sửa vi mô. Đa chỉnh sửa vi mô cho phép các chuyên gia so sánh những chỉnh sửa khác nhau trong cùng một câu có tác động thế nào đến việc sản sinh sự hài hước, tạo tiền đề cho các nghiên cứu tiếp theo trong tương lai.

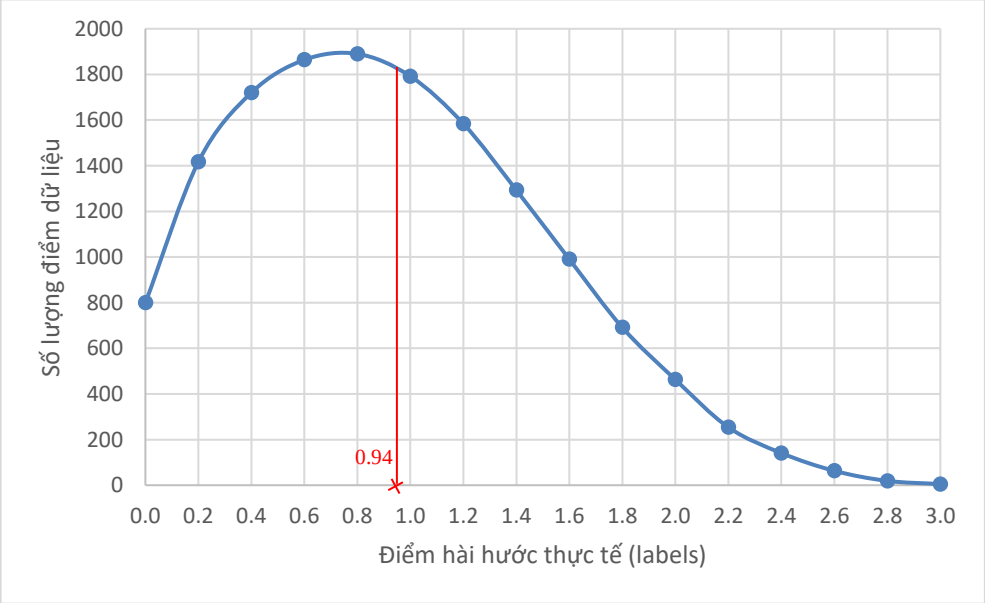
Trong quá trình chỉnh sửa và gán nhãn dữ liệu, một số công cụ giúp sửa lỗi chính tả, tìm tự bị thay thế không đúng quy tắt được sử dụng để hạn chế sai sót từ các chuyên gia. Đội ngũ chỉnh sửa và cho điểm cũng được chia ra làm việc và nghỉ ngơi để đảm bảo chất lượng dữ liệu. Trong quá trình này, những chuyên gia có hiệu suất thấp hơn tiêu chuẩn tiếp tục được loại bỏ. Từ gần 300,000 câu tiêu đề, các tác giả thu được 15,095 câu tiêu đề duy nhất đã được chỉnh sửa và đánh giá độ hài hước.

2.6. Phân tích sự hài hước

Phần này sẽ trình bày một số yếu tố cụ thể tác động đến sự hài hước. Các yếu tố còn lại được phân tích trong tương lai.

2.6.1. Ảnh hưởng của chỉnh sửa vi mô

Phần lớn các câu tiêu đề có mức độ hài hước từ nhẹ đến vừa. Một phần không nhỏ câu tiêu đề khó mà tạo được sự hài hước thông qua chỉnh sửa vi mô. Các tác giả cho biết nhóm chuyên gia gặp khó khăn khi tạo sự hài hước cho những câu tiêu đề có nội dung tiêu cực như chết chóc, bạo lực… Khi đội ngũ chỉnh sửa câu tiêu đề tập trung vào những thông tin, kiến thức ít được biết đến như những người kém nổi tiếng hay những vấn đề chính trị kém quan trọng, sự hài hước thu được cũng không cao.

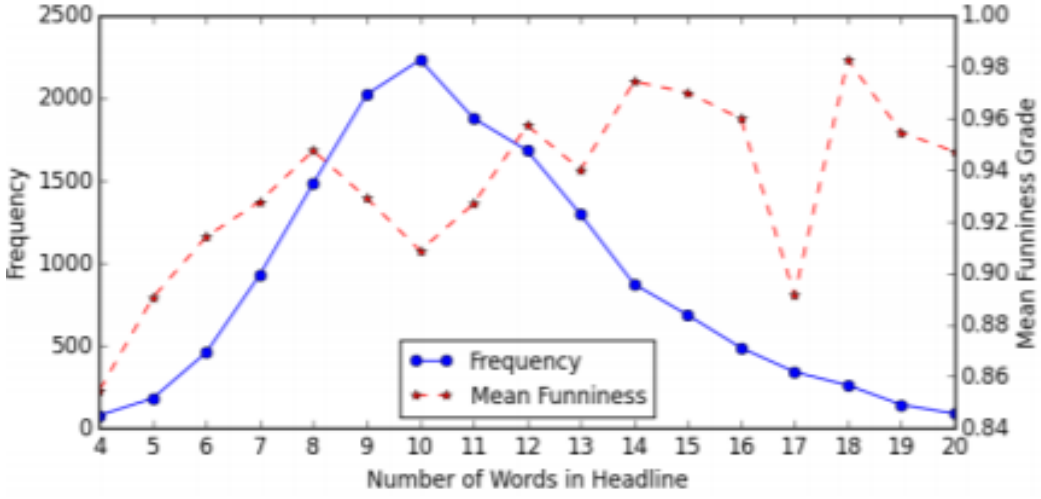


Hình 2.1: Biểu đồ thể hiện độ hài hước tiềm năng và độ hài hước trung bình của các câu tiêu đề đã chỉnh sửa trên toàn bộ dữ liệu.

Từ Hình 2.1 có thể thấy, việc tạo ra các câu tiêu đề hài hước là khó khăn. Số lượng câu tiêu đề không thể tạo ra sự hài hước dù đã áp dụng các "chỉnh sửa vi mô" vào khoảng 800 câu. Số lượng các câu tiêu đề có độ hài hước cao (chẳng hạn các câu tiêu đề có điểm hài hước trung bình là 2.4, 2.6, 2.8, 3.0) là rất ít. Các câu tiêu đề có độ hài hước "nhẹ" đến "vừa" chiếm đa số, trong đó *điểm hài hước tiềm năng* là 0.94 (đường thẳng màu đỏ). Điểm hài hước tiềm năng là trung bình cộng của các giá trị hài hước của tất cả câu tiêu đề đã chỉnh sửa tại mỗi mức điểm.

2.6.2. Ảnh hưởng của độ dài câu đến sự hài hước

Một số nhà ngôn ngữ học cho rằng lời nói đùa (jokes) nên được tạo ra với số lượng từ ít để người đọc/người nghe tư duy về nó, hiểu theo nhiều nghĩa khác nhau và từ đó nhận ra sự hài hước [9]. Một số khác nhận xét lời nói đùa nên có thêm thông tin để những lời đùa giữn này hài hước hơn [10]. Trong khi các câu tiêu đề tin tức hài hước là một dạng của những lời nói đùa, các tác giả nhận thấy các câu tiêu đề dài hơn thường có khả năng hài hước hơn, Hình 3.2.



Hình 2.2: Ảnh hưởng của độ dài câu tiêu đề đến tiềm năng của sự hài hước, ảnh trích từ [1].

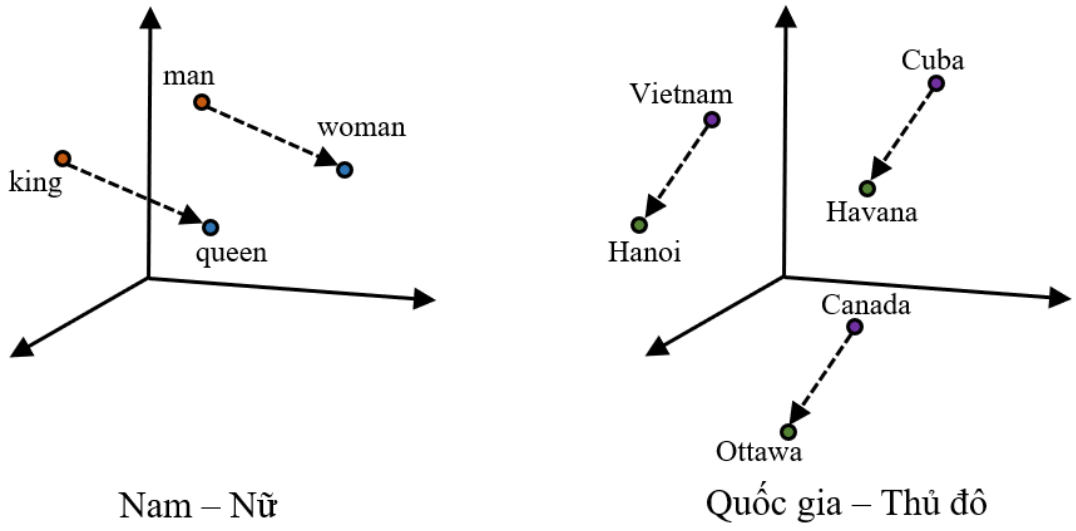
Như thể hiện ở Hình 2.2, mật độ các câu tiêu đề đạt đỉnh ở mức 10 từ cho một câu. Đường màu xanh biểu thị số lượng các câu tiêu đề theo số lượng từ. Đường màu đỏ là điểm hài hước trung bình trên từng độ dài câu. Số lượng câu tiêu đề có nhiều hơn 10 từ giảm dần khá đều. Những câu tiêu đề ít hài hước nhất là những câu ngắn nhất (4-5 từ). Điều này dễ hiểu vì câu quá ngắn thì không đủ thông tin ngữ cảnh để sự hài hước bùng nổ và các chuyên gia sẽ khó khăn trong việc chọn từ để thay thế, ngược lại, các câu có nhiều thông tin ngữ cảnh sẽ dễ dàng chọn từ thay thế hơn. Tuy nhiên, câu quá dài đôi khi cũng khó có độ hài hước cao do nó khá phức tạp để hiểu [11].

Chương 3. CƠ SỞ LÝ THUYẾT

Chương này trình bày các phương pháp, kỹ thuật được sử dụng trong khóa luận như phương pháp nhúng từ, các mô hình học sâu, một số kỹ thuật hỗ trợ quá trình huấn luyện và phương pháp đánh giá hiệu suất mô hình.

3.1. Phương pháp biểu diễn từ

Máy tính chỉ thao tác với các con số nên dữ liệu cần được xử lý và chuyển đổi về dạng thức phù hợp để máy tính có thể hiểu và thao tác được. Trong các phương pháp chuyển đổi này, nhúng từ hay biểu diễn từ là một phương pháp hiệu quả, giúp nhiều hệ thống đạt kết quả "state-of-the-art" trong lĩnh vực xử lý ngôn ngữ tự nhiên.



Hình 3.1: Minh họa phép nhúng từ.

Nhúng từ hay biểu diễn từ (word embedding/representation) là phương pháp ánh xạ các từ vào một không gian số thực nhiều chiều nhưng có kích thước nhỏ hơn rất nhiều (thường là vài trăm) so với kích thước từ điển (thường là vài trăm nghìn). Đây là tên gọi chung cho một tập hợp các mô hình ngôn ngữ (language model) và các phương pháp học đặc trưng trong xử lý ngôn ngữ tự nhiên. Phương pháp này đặc biệt quan trọng trong việc xử lý văn bản đầu vào cho các mô hình học máy, học sâu. Một phép nhúng từ tốt sẽ đem lại nhiều lợi ích về mặt tính toán và minh họa dữ liệu. Như thể hiện ở Hình 3.1, một phép nhúng từ tốt thì các vector tổng hợp của

“king – queen” (quốc vương – nữ hoàng) sẽ (gần như) song song với vector tổng hợp của “man – woman” (đàn ông – phụ nữ) trong ngữ cảnh “Nam – Nữ”. Điều tương tự cho ngữ cảnh “Quốc gia – Thủ đô”. Phần lớn các phương pháp vector hóa từ dựa vào khoảng cách hoặc góc giữa các cặp vector từ để đánh giá chất lượng nội tại của tập biểu diễn từ [12].

Một số phương pháp nhúng từ dựa trên thống kê như one-hot encoding, ma trận đồng xuất hiện (co-occurrence matrix), BoW (Bag of Word), TF-IDF (Term Frequency – Inverse Doccument Frequency). Nhược điểm của các phương pháp này là lưu trữ ít thông tin, số chiều vector tăng tuyến tính theo kích thước từ điển, cần nhiều không gian lưu trữ, gặp vấn đề biểu diễn thưa (sparsity issues), chi phí tính toán lớn, khó thêm dữ liệu mới. Do đó, các phương pháp này dần ít được sử dụng, thay vào đó là các phương pháp hiện đại hơn.

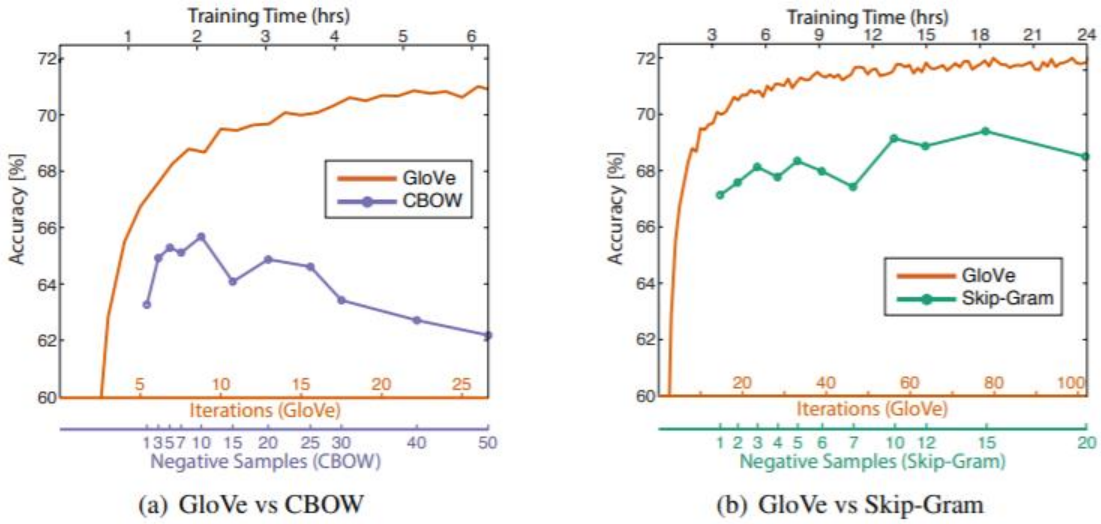
Các phương pháp nhúng từ phổ biến hiện này đều sử dụng mạng thần kinh. Chúng ta hoàn toàn có thể tự huấn luyện một tập nhúng từ, nhưng nó sẽ kém hiệu quả vì hạn chế về số lượng dữ liệu và cấu hình phần cứng. Do đó, sử dụng các tập nhúng từ đã được xây dựng sẵn (pretrained word embedding) sẽ tận dụng được các thành quả đi trước, giúp giảm chi phí tính toán và nâng cao hiệu suất mô hình. Các tập nhúng từ này đã được huấn luyện trên tập dữ liệu cực lớn bằng các siêu máy tính, sau đó được lưu lại và sử dụng cho các bài toán liên quan. Đây cũng là lý do vì sao các tập nhúng từ được huấn luyện trước được xem là một dạng của học chuyển đổi (transfer learning). Có hai hướng tiếp cận để biểu diễn từ, đó là dựa trên việc dự đoán ngữ cảnh (Prediction-based embedding) và dựa trên tần số (Frequency-based embedding). Các đại diện tương ứng là Word2vec [13] và GloVe.

3.1.1. Tập nhúng từ GloVe

Nếu Word2vec được xây dựng dựa trên việc dự đoán từ mục tiêu và ngữ cảnh thì ý tưởng đằng sau GloVe [12] là rút ra mối quan hệ giữa các từ dựa trên thống kê toàn cục (global statistics) tần suất xuất hiện của từ. Ý tưởng này xuất phát từ việc quan sát thấy rằng những từ có cùng ý nghĩa hoặc có mối quan hệ gần gũi sẽ

có xác suất xuất hiện đồng thời cao hơn. Tập nhúng từ GloVe ra đời vào năm 2014, được huấn luyện trên các bộ dữ liệu từ Wikipedia, Gigaword và Common Crawl.

GloVe là một mô hình hồi quy logarit tuyến tính hai chiều toàn cục (global log-bilinear regression) kết hợp ưu điểm của ma trận nhân tố toàn cục và cửa sổ ngữ cảnh cục bộ. Kiến trúc này tận dụng được các thông tin thống kê qua việc chỉ huấn luyện trên các phần tử khác 0 trong một ma trận từ-từ đồng xuất hiện (word-word co-occurrence matrix) hơn là sử dụng toàn bộ ma trận thưa thớt hoặc một cửa sổ ngữ cảnh riêng biệt.



(a) GloVe vs CBOW (b) GloVe vs Skip-Gram

Hình 3.2: So sánh hiệu suất trên nhiệm vụ tuần tự giữa GloVe với CBOW và Skip-gram của Word2vec, ảnh trích từ [12].

Theo Hình 3.2, Word2vec được cài đặt theo single epoch nên việc lấy mẫu được các tác giả GloVe chọn ngẫu nhiên. Tuy nhiên, sự chênh lệch về độ chính xác giữa 2 loại mô hình có khoảng cách khá lớn. Khi tăng số lần lặp thì GloVe có xu hướng tăng độ chính xác, trong khi độ chính xác của CBOW và Skip-gram thường đạt đỉnh ở lần lặp thứ 10 đến 15 vài có xu hướng giảm nếu tiếp tục lặp. GloVe cho kết quả vượt trội hơn CBOW và Skip-gram khoảng từ 2% đến 5% nếu xét theo kết quả tốt nhất của mỗi mô hình đạt được.

Về bản chất, Word2vec và GloVe thuộc về hai loại nhúng từ khác nhau nhưng đều bắt nguồn từ cửa sổ ngữ cảnh. Word2vec sử dụng cửa sổ ngữ cảnh này để tạo dữ liệu huấn luyện cho mạng thần kinh nông, trong khi GloVe sử dụng để tạo ma trận đồng xuất hiện. Do GloVe tính xác suất của từ trên toàn bộ tập dữ liệu nên sẽ mang tính toàn cục hơn Word2vec khi chỉ học dựa trên các ngữ cảnh đơn lẻ. Hình 3.2 cho thấy GloVe vượt trội hơn Word2vec trong một số tác vụ.

3.1.2. Tập nhúng từ Fasttext

Fasttext [14], [15] là tập nhúng từ được Facebook xây dựng dựa trên ý tưởng của tập nhúng từ Word2vec, được giới thiệu lần đầu vào năm 2016 và hiện đã có một phiên bản mới được giới thiệu vào năm 2018. Fasttext hỗ trợ đến 157 ngôn ngữ, có thể sử dụng mô hình đã được xây dựng để phân loại văn bản hoặc sử dụng tập nhúng từ đã được đào tạo trên các tập dữ liệu khổng lồ cho các nhiệm vụ khác.

Có hai sự lựa chọn cho tập nhúng từ của Fastext: tập nhúng từ có "subword" và tập nhúng từ không có "subword". Tập nhúng từ có sử dụng "subword" được cho là sẽ có hiệu quả khi gặp các từ hiếm hoặc không xuất hiện trong tập dữ liệu huấn luyện. Ví dụ từ "teach" sẽ được chia thành các từ con như "tea", "eac", "ach" nếu sử dụng số n-grams là 3. Các vector subword này có thể kết hợp với nhau để biểu diễn cho các từ hiếm gặp hơn, ví dụ như từ "tea" hay "approach". Tùy vào bài toán, có thể chọn các tập nhúng từ khác nhau với số lượng vector nhúng từ có thể là một triệu hoặc hai triệu trên trang chủ của Fasttext[5].

3.2. Một số hàm kích hoạt thông dụng

Với một nút trong mạng nơ-ron, các đầu vào được nhân với trọng số tương ứng rồi tính tổng. Giá trị tổng này được gọi là "tổng kích hoạt" (summed activation) của nút. Tổng kích hoạt sau đó sẽ được chuyển đổi thông qua một hàm kích hoạt thành một giá trị đầu ra cụ thể (còn được gọi là kích hoạt của một nút).

[5] https://fasttext.cc/docs/en/english-vectors.html [Online] Truy cập: 27/08/2020.

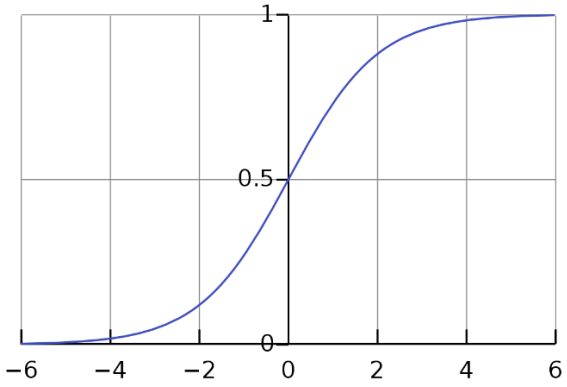
Hàm kích hoạt đơn giản nhất là hàm kích hoạt tuyến tính (linear activation), tức là không có chuyển đổi gì cả. Một mạng chỉ bao gồm các hàm kích hoạt tuyến tính rất dễ huấn luyện nhưng lại không thể học được những thứ phức tạp. Hàm kích hoạt tuyến tính vẫn còn được sử dụng trong lớp đầu ra của mạng dự đoán số lượng, ví dụ như các bài toán hồi quy. Trong khi đó, hàm kích hoạt phi tuyến cho phép các nút trong mạng học được nhiều cấu trúc phức tạp của dữ liệu. Trước đây, hàm *sigmoid* và *tanh* được sử dụng rộng rãi nhất. Hiện nay, có nhiều hàm kích hoạt mới hơn, trong đó *ReLU* dần trở nên phổ biến hơn cả[6].

3.2.1. Hàm Sigmoid

Hàm *sigmoid* còn được gọi là *logistic sigmoid*, ký hiệu σ, được sử dụng rộng rãi và là hàm mặc định cho nhiều mạng nơ-ron vào đầu thập niên 90. Công thức như sau:

$$\sigma(x) = \frac{1}{1+e^{-x}} \text{ (CT 3.1)}$$

Hàm *sigmoid* được sử dụng phổ biến, nhất là trong các tế bào (cell) LSTM hoặc tầng đầu ra của các mô hình phân loại. Có thể thấy, hàm *sigmoid* có công thức đơn giản, miền giá trị bị chặn trong khoảng (0.0, 1.0) và đạo hàm dễ tính. Nhược điểm của hàm *sigmoid* là có "vùng bão hòa", Hình 3.3.



Hình 3.3: Đồ thị hàm *sigmoid*.

[6] Nguồn: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/ [Online] Truy cập: 27/08/2020.

Hàm *sigmoid* bị chặn trong khoảng (0.0, 1.0) và có đạo hàm dễ tính nhưng có "vùng bão hòa". Các giá trị rất dương sẽ cho đầu ra tiệm cận 1.0 và rất âm sẽ cho đầu ra tiệm cận 0.0. Hàm chỉ thật sự nhạy cảm với những thay đổi xung quanh điểm giữa (mid-point) của đầu vào, ở đây là 0.5.

3.2.2. Hàm Tanh

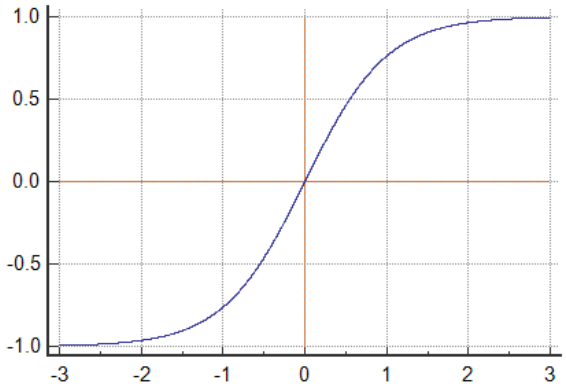
Hàm *tanh* cũng là một hàm được sử dụng phổ biến, thay thế *sigmoid* trở thành hàm kích hoạt mặc định trong nhiều mạng thần kinh vào cuối thập niên 90 và đầu những năm 2000. Công thức như sau:

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \text{ (CT 3.2)}$$

Hàm *tanh* bị giới hạn trong khoảng (-1.0, 1.0) nhưng có thể dễ dàng đưa về khoảng (0.0, 1.0) theo CT 3.3.

$$\tanh(x) = 2\sigma(2x) - 1 \text{ (CT 3.3)}$$

Hình 3.4 minh họa đồ thị hàm *tanh*.



Hình 3.4: Đồ thị hàm *tanh*.

Hàm *tanh* bị chặn trong đoạn [-1.0, 1.0] và cũng có "vùng bão hòa" tương tự hàm *sigmoid*. Các giá trị đầu vào rất dương được đưa về 1.0 và rất âm được đưa về -1.0. Hàm chỉ thật sự nhạy cảm với các thay đổi xung quanh điểm giữa của đầu vào, ở đây là 0.0.

Độ nhạy và độ bão hòa giới hạn của hàm xảy ra bất kể giá trị kích hoạt tổng hợp từ nút được cung cấp làm đầu vào có chứa thông tin hữu ích hay không. Khi đã

bão hòa, việc tiếp tục điều chỉnh các trọng số để cải thiện hiệu suất của mô hình trở nên khó khăn hơn. Đặc biệt khi sức mạnh phần cứng của GPU tăng lên, mạng càng sâu thì khả năng học khi sử dụng hàm sigmoid và tanh càng giảm do không nhận được các thông tin gradient bổ ích. Độ lỗi được lan truyền ngược trở lại mạng và được sử dụng để cập nhật trọng số sẽ giảm đáng kể với mỗi lớp mà nó truyền qua theo đạo hàm của hàm kích hoạt đã chọn. Đây là vấn đề độ dốc biến mất (vanishing gradients), ngăn các mạng học hiệu quả. Mặc dù việc sử dụng các hàm kích hoạt phi tuyến cho phép mạng nơ-ron học các hàm ánh xạ phức tạp, nhưng chúng cũng ngăn chặn hiệu quả thuật toán học làm việc với các mạng sâu.

Để sử dụng phép suy giảm độ dốc ngẫu nhiên với sự lan truyền các lỗi để đào tạo mạng nơ-ron học sâu thì cần có một hàm kích hoạt trông và hoạt động giống như một hàm tuyến tính, nhưng trên thực tế phải là một hàm phi tuyến cho phép học được các mối quan hệ phức tạp trong dữ liệu. Hàm này cũng phải cung cấp độ nhạy hơn cho đầu vào tổng kích hoạt và tránh dễ bão hòa. Trong những năm gần đây, ReLU nổi lên như là một giải pháp hữu hiệu.

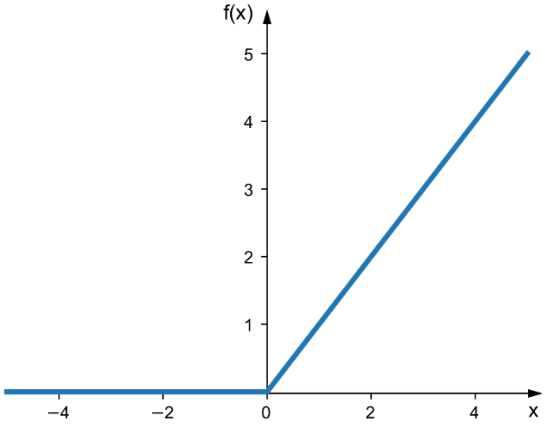
3.2.3. Hàm ReLU

Hàm kích hoạt tuyến tính chỉnh lưu (***Re****ctified* ***L****inear activation function– ReL*) trở nên phổ biến nhờ các nghiên cứu được công bố vào những năm 2009 và 2011. Một nút hoặc đơn vị (unit) thực hiện chức năng kích hoạt này được gọi là đơn vị kích hoạt tuyến tính chỉnh lưu (***Re****ctified* ***L****inear* ***U****nit – ReLU*). Thông thường, các mạng sử dụng chức năng chỉnh lưu cho các lớp ẩn được gọi là mạng chỉnh lưu.

Hàm ReLU trở thành hàm kích hoạt mặc định cho nhiều loại mạng nơ-ron vì có công thức đơn giản, có khả năng biểu diễn thưa khi có thể xuất ra giá trị 0 thật đối với các đầu vào âm (một số hàm khác như sigmoid và tanh học cách ước tính đầu ra: một giá trị rất gần với 0 nhưng không phải là 0) giúp đẩy nhanh quá trình học, có đặc tính của hàm tuyến tính hoặc gần tuyến tính giúp các mạng nơ-ron dễ tối ưu hơn, dễ tránh được hiện tượng vanishing gradients và thường cho hiệu suất tốt trong thực tế với cả các mạng rất sâu, đồ thị hàm ReLU như Hình 3.5.

Việc chấp nhận *ReLU* có thể được coi là một trong số ít các cột mốc quan trọng trong cuộc cách mạng học sâu, ví dụ: các kỹ thuật hiện cho phép phát triển thường xuyên các mạng nơ-ron rất sâu. Công thức tính đơn giản như sau:

$$relu(x) = \max(0, x) \text{ (CT 3.4)}$$



Hình 3.5: Đồ thị hàm *ReLU*.

Hàm *ReLU* là tuyến tính đối với các giá trị lớn hơn 0, có nghĩa là nó có rất nhiều đặc tính của một hàm kích hoạt tuyến tính khi huấn luyện một mạng nơ-ron bằng cách sử dụng lan truyền ngược. Tuy nhiên, nó là một hàm phi tuyến vì các giá trị âm luôn được xuất ra bằng 0.0. Về mặt kỹ thuật, chúng ta không thể tính đạo hàm khi đầu vào là 0.0, do đó, chúng ta có thể giả sử nó bằng 0. Đây không phải là một vấn đề có ảnh hưởng lớn trong thực tế.

Hàm *ReLU* có thể được sử dụng làm hàm kích hoạt mặc định cho các mạng nơ-ron, sử dụng với MLPs, CNNs nhưng nên cẩn thận khi dùng với RNNs vì có thể làm bùng nổ giá trị đầu ra. Một số "mẹo" được khuyến khích khi sử dụng *ReLU*: thử một hệ số bias nhỏ cùng với đầu vào, sử dụng "He weights initialization", sử dụng "weight penalty", scale dữ liệu…

Hạn chế của *ReLU* là trường hợp một nơ-ron âm khó có thể khôi phục được vì luôn xuất ra giá trị kích hoạt là 0.0. Điều này được gọi là *ReLU* đang chết (*dying ReLU*). Một số biến thể của *ReLU* như LeakyReLU, *ELU*… có các hệ số giúp giảm

ảnh hưởng của các đầu vào là số âm. Cả hai biến thể này đều nhằm tránh đi hiện tượng "*dying ReLU*".

Công thức cho hàm *LeakyReLU* như sau:

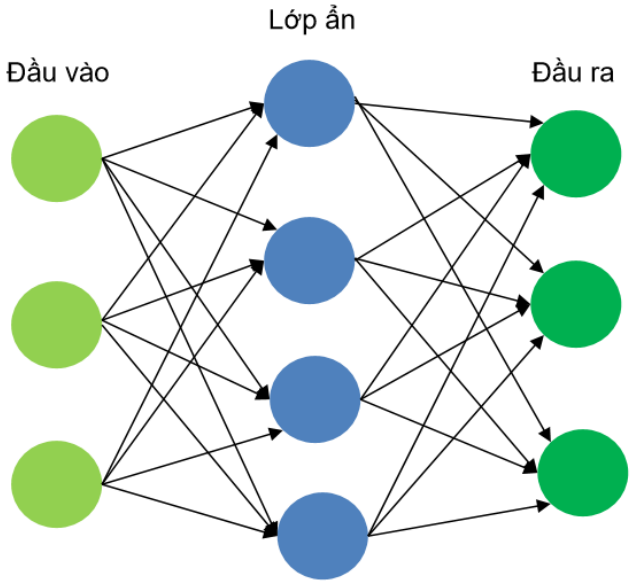
$$leakyrelu(x) = (x < 0)(\alpha x) + (x \geq 0)x \text{ (CT 3.5)}$$

Công thức cho hàm *ELU* như sau:

$$elu(x) = (x < 0)(\alpha(e^x - 1)) + (x \geq 0)x \text{ (CT 3.6)}$$

3.3. Các mô hình học sâu

Mạng nơ-ron nhân tạo (Artificial Neural Network - ANN) hay mạng nơ-ron, mạng thần kinh là một mô hình tính toán dựa trên sự mô phỏng hoạt động của hệ thần kinh con người. Kiến trúc của một mạng bao gồm các nơ-ron (còn gọi là các nút) được kết nối với nhau như Hình 3.6.



Hình 3.6: Kiến trúc mạng nơron nhân tạo.

Cấu tạo của một mạng nơ-ron nhân tạo gồm 3 lớp:

- Lớp đầu vào: nhận các vector đặc trưng và chuyển các giá trị này đến lớp ẩn.
- Lớp ẩn: tính toán ma trận trọng số từ các giá trị đầu vào trước đó tại các kết nối. Ma trận này sẽ được tối ưu sau mỗi lần huấn luyện. Kết quả tính toán tại mỗi nút ở lớp ẩn sẽ là đầu vào cho lớp tiếp theo.

- Lớp đầu ra: chứa kết quả dự đoán của mô hình. Tùy từng bài toán cụ thể là lớp đầu ra có thể có một hoặc nhiều nút.

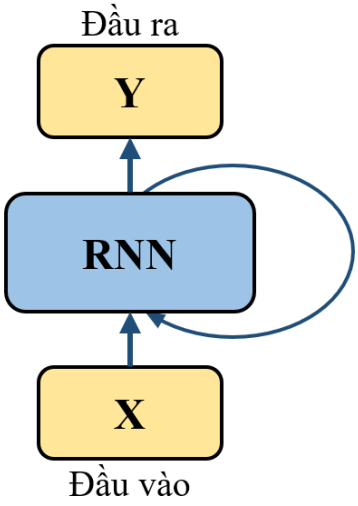
Mạng nơ-ron nhân tạo học bằng cách cho nhiều vector đầu vào đi qua, tính toán độ dốc (gradient) và độ lỗi giữa giá trị dự đoán và giá trị thực tế, sau đó áp dụng cơ chế lan truyền ngược để cập nhật trọng số của mô hình sao cho độ lỗi là nhỏ nhất.

Mô hình ANN có nhiều hơn một lớp ẩn được gọi là mạng nơ-ron/thần kinh học sâu (Deep Neural Network - DNN). Các mô hình học sâu hiện đại đều được phát triển từ đây.

3.3.1. Mạng nơ-ron hồi quy

Là một trong hai kiến trúc mô hình học sâu tiêu biểu, mạng nơ-ron hồi quy (Recurrent Neural Network - *RNN*) [16] đặc biệt hiệu quả đối với dữ liệu dạng chuỗi (sequence/time series) có các thành phần phụ thuộc lẫn nhau, chẳng hạn như ngôn ngữ của con người. Có thể nói, việc áp dụng *RNN* (còn gọi là *Vanilla RNN*) là một bước đột phá trong lĩnh vực xử lý ngôn ngữ tự nhiên nói chung cũng như trí tuệ nhân tạo nói riêng. Ngoài ra, các kiến trúc từ *RNN* cũng cho kết quả cao khi xử lý video, vì mỗi video là một tập hợp các hình ảnh được nối lại với nhau theo thứ tự thơi gian và có tính liên tục. Thông thường, mỗi giây sẽ có từ 24 đến 30 khung hình (frame).

Mạng nơ-ron hồi quy dựa trên kiến trúc của *DNN*, có bản chất là một hàm đệ quy. Trạng thái đầu ra được tính dựa vào trạng thái trước đó và đầu vào hiện tại. Hình 3.4 mô tả kiến trúc tổng quát của một mạng *RNN* đơn giản.



Hình 3.7: Kiến trúc của mạng nơ-ron hồi quy với vòng lặp.

Trong kiến trúc như hình 3.7, ***X*** là một chuỗi vector đầu vào. Khối *RNN* màu xanh sẽ áp dụng "công thức hồi quy" lên vector đầu vào và trạng thái trước đó của nó để tính toán trạng thái hiện tại. Công thức tổng quát cho trạng thái tại thời điểm ***t*** như sau:

$$h_t = f(x_t, h_{t-1}) \text{(CT 3.7)}$$

Với: $\boldsymbol{h_t}$ là trạng thái (ẩn) hiện tại, $\boldsymbol{h_{t-1}}$ là trạng thái ẩn trước đó, $\boldsymbol{x_t}$ là đầu vào hiện tại ở thời điểm (time step) ***t***, ***f*** là hàm kích hoạt.

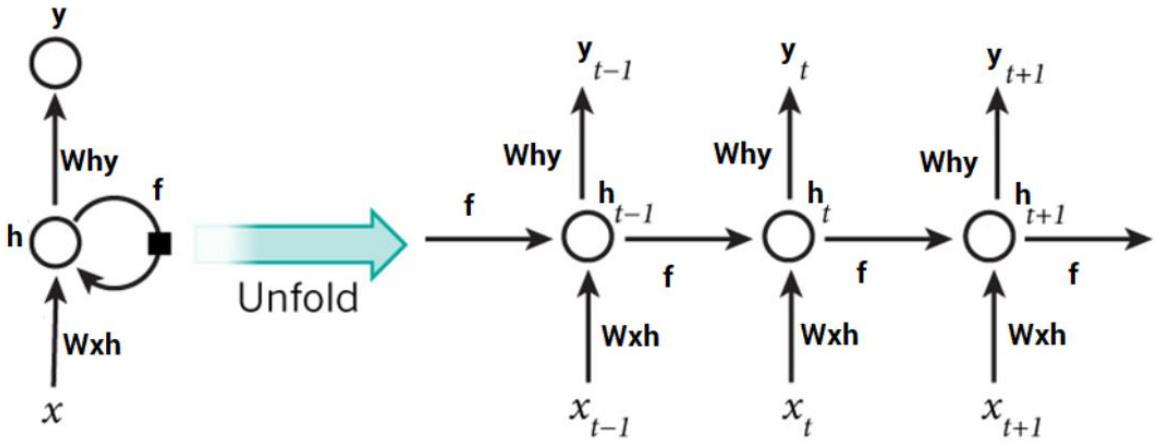
Công thức hồi quy này sẽ được áp dụng vào mạng tại mọi thời điểm. Mạng thần kinh hồi quy thường dùng hàm ***tanh*** làm hàm kích hoạt (activation function), ma trận trọng số cho đầu vào và trạng thái ẩn lần lượt là $\boldsymbol{W^{xh}}$ và $\boldsymbol{W^{hh}}$, trạng thái ẩn hiện tại được tính như sau:

$$h_t = \tanh(W^{xh} * x_t + W^{hh} * h_{t-1} + bias) \text{ (CT 3.8)}$$

Hàm ***tanh*** sẽ giúp điều chỉnh độ lớn của các giá trị tính toán sau khi qua mạng về đoạn [-1, 1], tránh hiện tượng một giá trị quá lớn được "thiên vị" so với các giá trị còn lại làm giảm tính hiệu quả của toàn mạng. Trạng thái ẩn cuối cùng tại mỗi timestep sẽ được sử dụng để tính giá trị đầu ra, trong đó $\boldsymbol{W^{hy}}$ là ma trận trọng số đầu ra:

$$y_t = W^{hy} * h_t \text{ (CT 3.9)}$$

Giá trị đầu ra dự đoán sẽ được so sánh với giá trị thực tế để tính toán độ lỗi. Cơ chế lan truyền ngược (backpropagated) được áp dụng trên toàn mạng để điều chỉnh các trọng số nhờ vào độ lỗi này. Thông thường, giá trị đầu ra tại mỗi timestep chỉ được tính và sử dụng khi xây dựng các mô hình ngôn ngữ (language modeling) dự đoán "từ tiếp theo". Đối với các bài toán phân loại hay hồi quy, chỉ có đầu ra cuối cùng của mạng được sử dụng. Hình 3.8 mô tả chi tiết kiến trúc mạng *RNN* qua từng bước.



Hình 3.8: Kiến trúc trải phẳng của mạng nơ-ron hồi quy khi[7].

Lan truyền tiến (feed forward) là quá trình các vector đầu vào lần lượt đi qua mạng để tính toán giá trị dự đoán. Lan truyền ngược (backpropagation) là quá trình đi ngược lại toàn mạng để cập nhật các ma trận trọng số nhằm làm giảm tối đa độ lỗi đã được tính toán trước đó. Nếu $\mathbb{y}_t$ là giá trị thực tế, $\boldsymbol{y_t}$ là giá trị dự đoán, giá trị lỗi tại bước thứ $\boldsymbol{t}$ với hàm mất mát cross-entropy là:

$$E_t(\mathbb{y}_t, \boldsymbol{y_t}) = -\mathbb{y}_t \log(\boldsymbol{y_t}) \text{ (CT 3.10)}$$

Thông thường, một mẫu dữ liệu huấn luyện bao gồm cả một câu hoặc một từ. Do đó, độ lỗi cho một mẫu sẽ bằng tổng độ lỗi trên từng bước thời gian $\boldsymbol{t}$, tương ứng từng từ trong câu hoặc từng ký tự trong từ. Công thức tính như sau:

$$E(\mathbb{y}, y) = -\sum \mathbb{y}_t \log(y_t) \text{ (CT 3.11)}$$

[7] Nguồn ảnh: https://machinelearning.mipa.ugm.ac.id/2018/07/01/recurrent-neural-network-rnn/ [Online] Truy cập: 27/08/2020.

Ở mỗi thời điểm ***t***, độ dốc (gradient) cũng được tính tương ứng với trọng số. Vì trọng số là giống nhau cho mọi thời điểm nên độ dốc cũng được kết hợp lại với nhau ở tất cả thời điểm. Sau cùng, trọng số sẽ được cập nhật ở cả nơ-ron hồi quy và các lớp kết nối dày đặc (dense layers).

Mạng thần kinh hồi quy dạng đơn giản có nhược điểm là kém hiệu quả trong việc xử lý những chuỗi dài. Nguyên nhân là do đạo hàm bị triệt tiêu (vanishing gradients) vì phải đạo hàm nhiều lần dẫn đến các thông tin ở xa không được truyền tới ngữ cảnh hiện tại. Để khắc phục nhược điểm nói trên của RNN, một số biến thể đã ra đời, tiêu biểu là kiến trúc Long-Short Term Memory và Gated-Recurrent Unit.

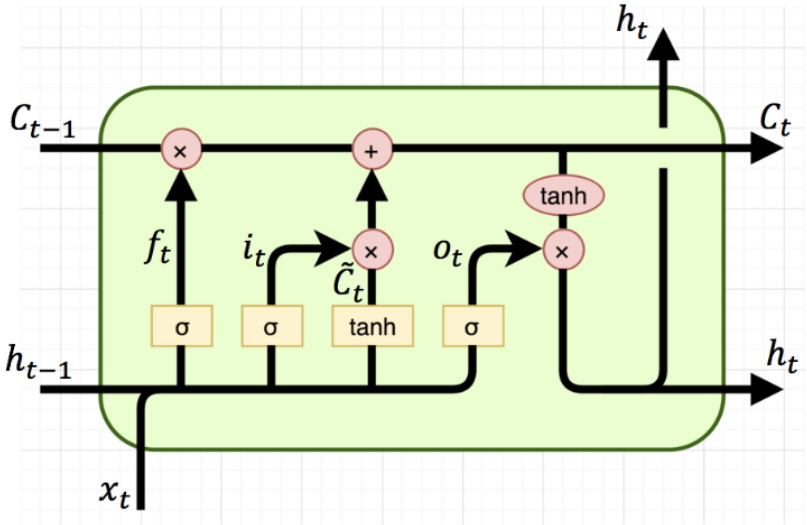
3.3.2. Bộ nhớ dài hạn-ngắn hạn (Long-Short Term Memory)

Với mạng nơ-ron hồi quy, đạo hàm tại trạng thái ***t*** so với trạng thái rất xa trước đó, giả sử như ***t-30*** sẽ xấp xỉ bằng 0 hay $\frac{\partial L}{\partial W} \approx 0$. Các trạng thái càng xa trạng thái hiện tại ***t*** thì đạo hàm sẽ càng bị triệt tiêu, dẫn đến trọng số gần như không được cập nhật, hay nói cách khác là mạng không học được gì mới. Về mặt lý thuyết, *RNN* có thể mang thông tin từ các lớp trước đến các lớp sau, nhưng thực tế thì lượng thông tin chỉ có thể được truyền sang một số trạng thái nhất định – bộ nhớ ngắn hạn (short-term memory). Xuất phát từ lý do này, một biến thể của mạng nơ-ron hồi quy là bộ nhớ ngắn hạn - dài hạn (*Long-Short Term Memory - LSTM*) [16] ra đời. Biến thể mới này kế thừa các ưu điểm và khắc phục phần nào các hạn chế mà mô hình mạng hồi quy truyền thống gặp phải.

Đặc điểm nổi bật giúp bộ nhớ ngắn hạn-dài hạn có khả năng ghi nhớ lâu hơn đó là nhờ vào việc áp dụng cơ chế "cổng" (gates). Các cổng này có nhiệm vụ quyết định thông tin nào cần thêm mới, thông tin nào có thể "quên", từ đó giúp tăng khả năng ghi nhớ và lưu trữ những thông tin quan trọng. Điểm khác biệt của *LSTM* so với *RNN* là ở cấu trúc của các tế bào ô nhớ (cell).

Trong khi mạng thần kinh hồi quy truyền thống chỉ sử dụng hàm **tanh** với dữ liệu đầu vào là trạng thái hiện tại x_t và thông tin lưu trữ từ timestep trước đó h_{t-1} thì

LSTM sử dụng thêm hàm **sigmoid (*σ*)** và thông tin lưu trữ từ trạng thái tế bào ô nhớ (cell state) trước đó c_{t-1}. Như vậy, một *cell* của *LSTM* có ba đầu vào, sử dụng hai loại hàm kích hoạt và có hai đầu ra. Hình 3.9 mô tả một *cell* trong kiến trúc bộ nhớ dài hạn – ngắn hạn.



Hình 3.9: Một *cell* của kiến trúc mô hình bộ nhớ dài hạn – ngắn hạn[8].

Có thể thấy, cấu trúc *cell* của *LSTM* được hình thành từ 3 cổng: cổng quên (forget gate), cổng đầu vào (input gate) vào và cổng đầu ra (output gate).

- Cổng quên: có nhiệm vụ quyết định lượng thông tin cần giữ lại ở trạng thái trước đó. Thông tin từ đầu vào hiện tại x_t và trạng thái ẩn trước đó h_{t-1} được nối lại với nhau bằng một phép *concatenation*, tạm gọi kết quả của phép nối này là vector $[x_t, h_{t-1}]$. Công thức tính:

$$f_t = \sigma\left(U^f x_t + W^f{}_f h_{t-1} + b^f\right) \text{(CT 3.12)}$$

- Cổng đầu vào: có nhiệm vụ cập nhật thông tin mới vào ô trạng thái hiện tại. Cổng này tương tự với cổng quên, sử dụng hàm *sigmoid* lên vector $[x_t, h_{t-1}]$ để đánh giá lượng thông tin cần thêm vào, chỉ khác ma trận trọng số được sử dụng. Công thức tính như sau:

$$i_t = \sigma(U^i x_t + W^i h_{t-1} + b^i) \text{(CT 3.13)}$$

[8] Nguồn ảnh: https://towardsdatascience.com/grus-and-lstm-s-741709a9b9b1 [Online] Truy cập: 27/08/2020.

- Cổng đầu ra: có nhiệm vụ tính giá trị trạng thái ẩn hiện tại để làm đầu vào cho timestep tiếp theo. Giống như hai cổng phía trên, cổng đầu ra cũng áp dụng hàm *sigmoid* lên vector $[x_t, h_{t-1}]$ để đánh giá lượng thông tin cần chuyển tiếp từ trạng thái hiện tại nhưng sử dụng khác ma trận trọng số:

$$o_t = \sigma(U^o x_t + W^o h_{t-1} + b^o) \text{(CT 3.14)}$$

Lưu ý: U, W là các ma trận trọng số và b là bias tương ứng từng cổng.

Nhận xét: vì hàm kích hoạt sigmoid được sử dụng nên f_t, i_t, o_t có giá trị nằm trong khoảng [0, 1]. Theo đó, nếu các giá trị này gần với 1 thì lượng thông tin sẽ được giữ lại càng nhiều, ngược lại, thông tin được xem là không cần thiết và bị loại bỏ.

Mỗi một *cell* trong kiến trúc mô hình bộ nhớ dài hạn – ngắn hạn có hai đầu ra và được tính như sau:

- Trạng thái tế bào hiện tại $\boldsymbol{c_t}$ (cell state):

$$c_t = f_t c_{t-1} + i_t \tilde{c}_t \text{ (CT 3.15)}$$

với

$$\tilde{c}_t = \tanh(U^c x_t + W^c h_{t-1} + b^c) \text{ (CT 3.16)}$$

Có thể thấy, "cổng quên" quyết định cần lấy (hay quên) bao nhiêu thông tin từ *cels state* trước đó và "cổng đầu vào" sẽ quyết định số lượng thông tin cần thêm vào từ đầu vào hiện tại và trạng thái ẩn của lớp trước đó.

- Trạng thái ẩn hiện tại $\boldsymbol{h_t}$:

$$h_t = o_t \tanh(c_t) \text{(CT 3.17)}$$

Tại đây, "cổng đầu ra" quyết định xem cần lấy bao nhiêu thông tin từ cell state hiện tại (sau khi đã thực hiện tính toán) làm trạng thái ẩn cho trạng thái hiện tại. Ngoài ra $\boldsymbol{h_t}$ cũng được dùng để tính giá trị dự đoán $\mathbf{y_t}$ tại trạng thái thứ **t**.

Nhận xét: có thể thấy rằng, $\boldsymbol{h_t}$ và $\tilde{\boldsymbol{c}}_{\boldsymbol{t}}$ khá giống với kiến trúc mô hình mạng hồi quy nên *LSTM* có "*short-term memory*". Trong khi đó, $\boldsymbol{c_t}$ giống như một băng chuyền mang những thông tin cần thiết được gửi vào và truyền đi qua các timestep, đây chính là "long-term memory". Vì thế, *LSTM* có cả "long-term" và "short-term".

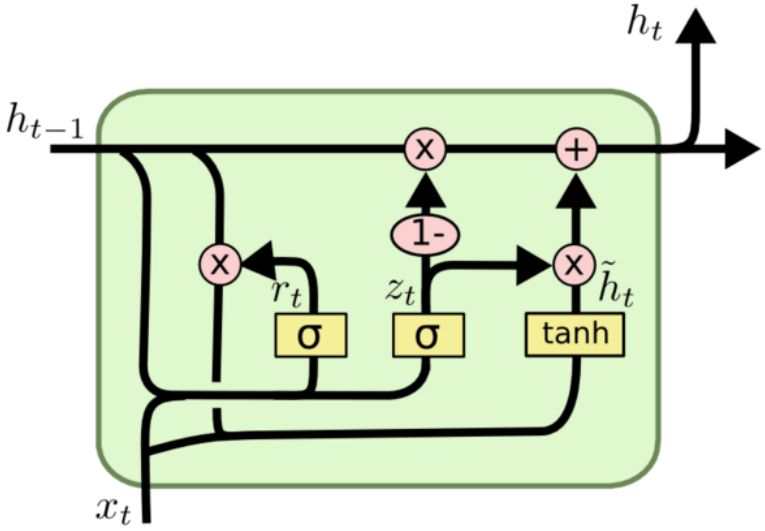
Vậy, *LSTM* có thật sự ghi nhận được tất cả thông tin ở xa? Giống như mạng *RNN*, bộ nhớ dài hạn – ngắn hạn cũng áp dụng thuật toán lan truyền ngược cho để cập nhật trọng số.

$$\frac{\partial c_t}{\partial c_{t-1}} = f_t \text{(CT 3.18)}$$

Do $0 < f_t < 1$ nên về cơ bản, *LSTM* vẫn có hiện tượng bị triệt tiêu đạo hàm nhưng ít hơn nhiều khi so với *RNN*. Hơn nữa, thông tin được lưu trữ trên cell state thì ít khi cần phải quên giá trị cell cũ nên $f_t \approx 1$, từ đó hạn chế được phần nào hiện tượng đạo hàm bị triệt tiêu (vanishing gradient). Vì những lý do này, *LSTM* được sử dụng phổ biến hơn cả.

3.3.3. Đơn vị cổng tái phát (Gated Recurrent Unit)

Ngoài *LSTM*, mạng nơ-ron hồi quy còn có một biến thể khác, đó là đơn vị cổng tái phát (*Gated Recurrent Unit – GRU*) [17]. Kiến trúc này có nguyên tắc hoạt động tương tự *LSTM* nhưng cấu tạo đơn giản hơn nhiều. *GRU* kết hợp trạng thái tế bào hiện tại (cell state) và trạng thái ẩn (hidden state) thành một nên chỉ có hai đầu vào và một đầu ra.



Hình 3.10: Kiến trúc mô hình GRU[9].

[9] Nguồn ảnh: https://technopremium.com/blog/rnn-talking-about-gated-recurrent-unit/ [Online] Truy cập: 27/08/2020.

Hình 3.10 mô tả kiến trúc mô hình *GRU*. Kiến trúc này có nhiều điểm tương đồng với mô hình *LSTM* nhưng đơn giản hơn. Để tránh hiện tượng đạo hàm bị triệt tiêu, *GRU* sử dụng hai cổng là cổng cập nhật (update gate) và cổng cài đặt lại (reset gate) để quyết định việc lưu trữ và loại bỏ thông tin.

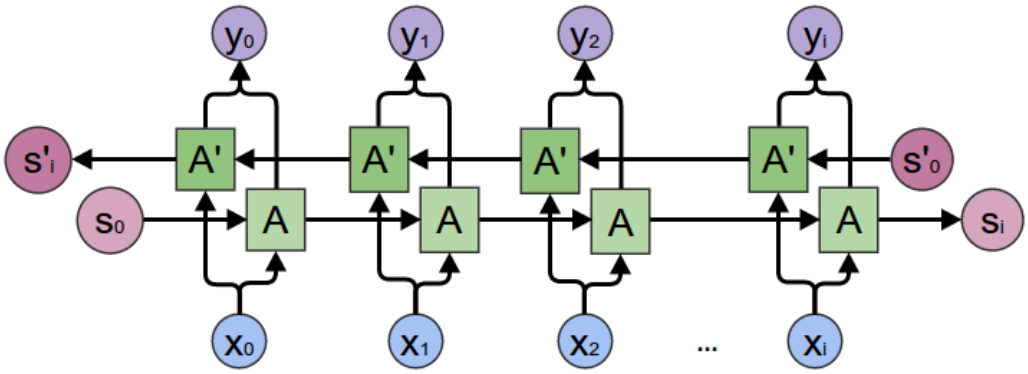
- Cổng cập nhật - $\mathbf{z_t}$: nhận vào hai thành phần là đầu vào tại timestep hiện tại $\boldsymbol{x_t}$ và giá trị trạng thái ẩn trước đó (hay đầu ra của timestep liền trước) $\boldsymbol{h_t}$. Hai thành phần này sau đó được áp dụng phép *concatenation* và đi qua hàm *sigmoid*. Các giá trị sẽ được đưa về đoạn [0, 1] để quyết định lượng thông tin sẽ thêm vào trạng thái hiện tại.
- Cổng cài đặt lại - $\boldsymbol{r_t}$: tương tự như cổng cập nhật nhưng tác dụng là để quyết định lượng thông tin cần giữ lại hay cần loại bỏ.

Vì mất đi trạng thái của *cell* (so với *LSTM*) nên tác dụng của hai cổng trên khá khó để phân biệt rõ ràng. Có thể nói, cả hai cổng đều nhằm chắt lọc thông tin từ các đầu vào của *cell* để tạo đầu ra thỏa mãn cả hai tiêu chí: lưu giữ thông tin quá khứ và có khả năng đưa ra quyết định hiện tại một cách chính xác nhất. Do ít phức tạp hơn *LSTM* nên *GRU* có thời gian tính toán nhanh hơn đôi chút. Ngược lại, *LSTM* lưu trữ được nhiều thông tin và mang đi xa hơn. Xét về hiệu quả thì rất khó để đưa ra kết luận chính xác nên khi thực nghiệm cần áp dụng và đánh giá trên cả hai kiến trúc này [11].

3.3.4. Mạng hồi quy hai chiều

Đặc điểm chung của mạng nơ-ron hồi quy và hai biến thể là *LSTM* và *GRU* là đều hoạt động theo một chiều nhất định (forward direction). Nói cách khác, các mạng này chỉ mang hay "thấy" thông tin từ thời điểm đầu tiên đến thời điểm hiện tại. Tuy nhiên, trong nhiều bài toán NLP như dịch máy, nhận dạng giọng nói, nhận dạng chữ viết tay… thì việc biết trước thông tin của các timestep tiếp theo sẽ giúp cải thiện kết quả một cách đáng kể. Trong trường hợp này, chúng ta có thể sử dụng mạng hồi quy hai chiều (*Bidirectional RNN*) với việc xử lý thông tin theo cả hai chiều, từ trái

sang phải và từ phải sang trái (forward và backward). Nguyên lý hoạt động của mạng thần kinh hồi quy hai chiều có thể được mô tả như Hình 3.11.

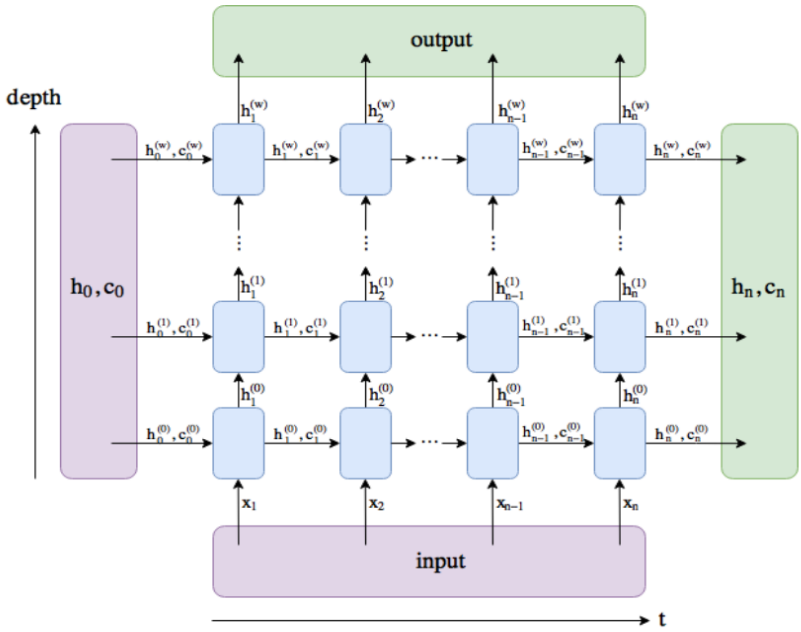


Hình 3.11: Kiến trúc mạng hồi quy hai chiều[10].

Mạng hồi quy hai chiều gồm hai mạng hồi quy con A (chiều từ trái sang phải – forward) và A’ (chiều từ phải sang trái - backward) xếp chồng lên nhau. Mỗi khối hay cell này có nguyên tắc hoạt động tương tự như các mạng hồi quy đã trình bày ở trên. Tuy nhiên, đầu ra dự đoán tại mỗi timestep y_t được tính dựa trên giá trị đầu ra của cả hai khối A và A’, tức là kết hợp cả hai trạng thái ẩn $\boldsymbol{h_t}$ từ hai hướng forward và backward. Khối A và A’ có thể cùng là *Vanilla RNN*, *LSTM* hoặc *GRU*.

[10] Nguồn ảnh: https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66 [Online] Truy cập: 27/08/2020.

3.3.5. Mạng hồi quy đa lớp (Multi-layers RNNs)



Hình 3.12: Kiến trúc mạng hồi quy đa lớp[11].

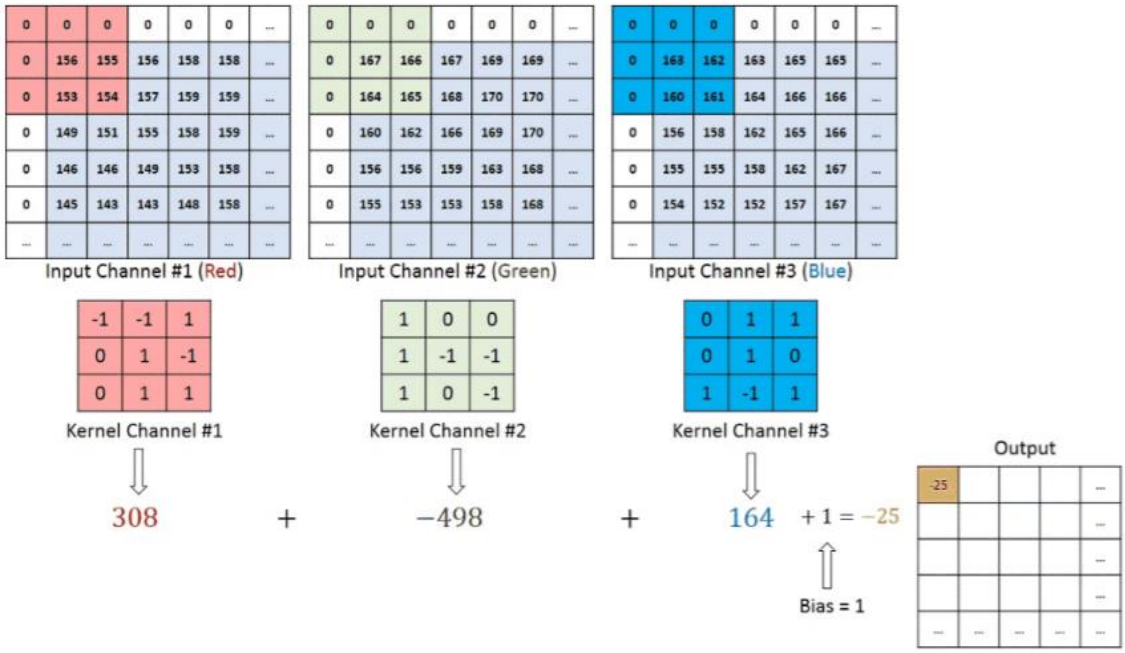
Có thể kết hợp nhiều mạng hồi quy để tạo thành một mạng nơ-ron hồi quy đa lớp (*multi-layer RNN*) với đầu ra của lớp này trở thành đầu vào của lớp tiếp theo. Hình 3.12 minh họa cho kiến trúc mạng hồi quy đa đa lớp. Mỗi lớp trong mạng đa lớp này có thể là mô hình mạng một chiều hoặc hai chiều. Kiến trúc mạng càng "sâu" càng giúp mô hình học được nhiều đặc trưng, tuy nhiên cũng dễ gây nên hiện tượng quá khớp (overfitting). Trong quá trình xây dựng và huấn luyện mô hình cần chú trọng đến độ sâu, nếu mô hình quá phức tạp so với dữ liệu thì sẽ không thể mô tả dữ liệu một cách tổng quát, dẫn đến việc kết quả rất tốt trên tập dữ liệu huấn luyện nhưng lại cho kết quả không tốt trên tập dữ liệu kiểm thử. Lúc này, mô hình cần phải sử dụng thêm một số kỹ thuật giảm độ phức tạp.

3.3.6. Mạng nơ-ron tích chập (Convolutional Neural Networks)

Mạng thần kinh tích chập (*Convolutional Neural Networks – CNN*) là một mạng nơ-ron dựa trên kiến trúc *DNN*. Mô hình này đặc biệt mạng lại nhiều thành tựu to lớn

[11] Nguồn ảnh: https://stackoverflow.com/questions/48302810/whats-the-difference-between-hidden-and-output-in-pytorch-lstm [Online] Truy cập: 27/08/2020.

cho lĩnh vực xử lý ảnh và thị giác máy tính. Quan sát thấy được, một bức ảnh màu thường có ba kênh màu (**R**ed – **G**reen - **B**lue) nên số lượng pixel (điểm ảnh) trên một bức ảnh càng lớn nếu kích thước của ảnh tăng lên. Nếu dùng một *DNN* thông thường thì số lượng tham số sẽ rất lớn. Tuy nhiên, các pixel gần nhau trong ảnh thường có "liên kết" với nhau hơn là các pixel ở xa. Hơn nữa, nếu dùng phép tích chập (convolution) thì chỉ một ma trận chập (kernel) được sử dụng chung trên toàn bộ bức ảnh, nói cách khác là các pixel ảnh chia sẻ hệ số với nhau. Do đó, nếu áp dụng phép tích chập lên ảnh thì sẽ giảm được số lượng tham số mà vẫn đảm bảo mô hình học được các đặc trưng quan trọng. Mỗi kernel ở đây sẽ phụ trách học một đặc trưng.

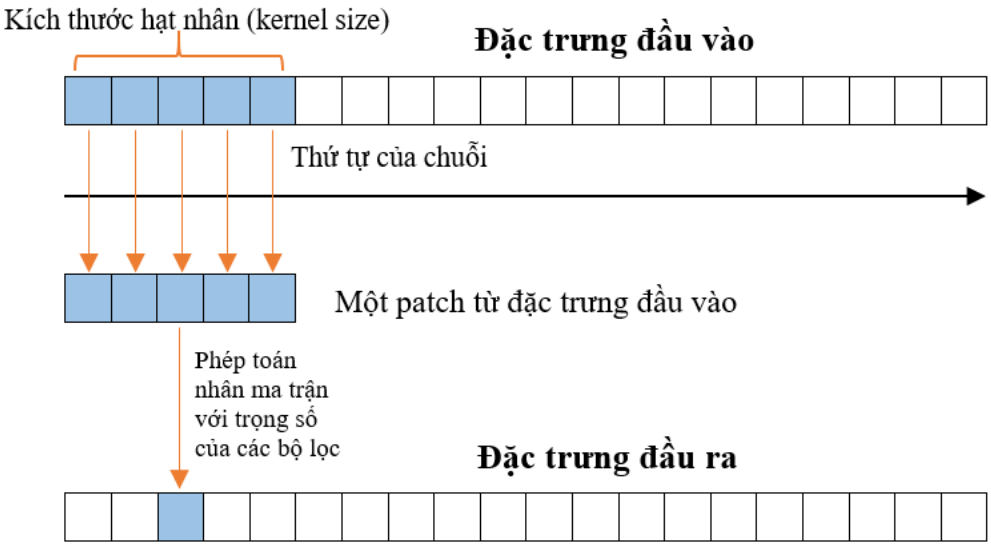


Hình 3.13: Minh họa việc thực hiện phép tính chập trên một ảnh màu có ba kênh ảnh, ảnh trích từ [37].

Hình 3.13 cho thấy quá trình tính toán tích chập trên một bức ảnh màu. Kết quả đầu ra của một phép chập trên ảnh màu là một ma trận, có một hệ số bias được cộng vào sau khi tính tổng các phần tử tương ứng của từng ma trận chập trên từng kernel.

Mô hình *CNN* được giới thiệu lần đầu tiên bởi Fukushima [18] từ nhiều năm trước nhưng chưa được sử dụng rộng rãi do giới hạn phần cứng máy tính [6]. Sau đó, có nhiều công trình nghiên cứu cải tiến, điển hình là nghiên cứu của *LeCun* và các cộng sự [19]. Nghiên cứu đã áp dụng thuật toán gradient-based vào mô hình *CNN* và đã đạt được thành tựu vượt trội cho bài toán nhận diện chữ viết tay. Kể từ đây, *CNN* được sử dụng phổ biến và thể hiện sự vượt trội của mình trong nhiều bài toán khác nhau, không chỉ dừng lại trong thị giác máy tính mà còn mở rộng sang các lĩnh vực khác như xử lý ngôn ngữ, âm thanh.

Trong xử lý ngôn ngữ tự nhiên, mạng nơ-ron tích chập cũng dần được áp dụng, điển hình phải kể đến nhóm của *Collobert* [20] hay *Yoon Kim* [21]. *CNN* hoạt động với dữ liệu đầu vào là những ma trận số. Với đầu vào dạng văn bản, kỹ thuật nhúng từ được giới thiệu ở phần 3.1 được áp dụng để chuyển đổi văn bản thành các vector. Khi làm việc với dữ liệu tuần tự như văn bản, chúng ta đang làm việc với các tích chập một chiều, nhưng ý tưởng vẫn giống như khi làm việc trên ảnh. Hình 3.14 mô tả cách thức hoạt động của ma trận chập một chiều trên dữ liệu tuần tự.



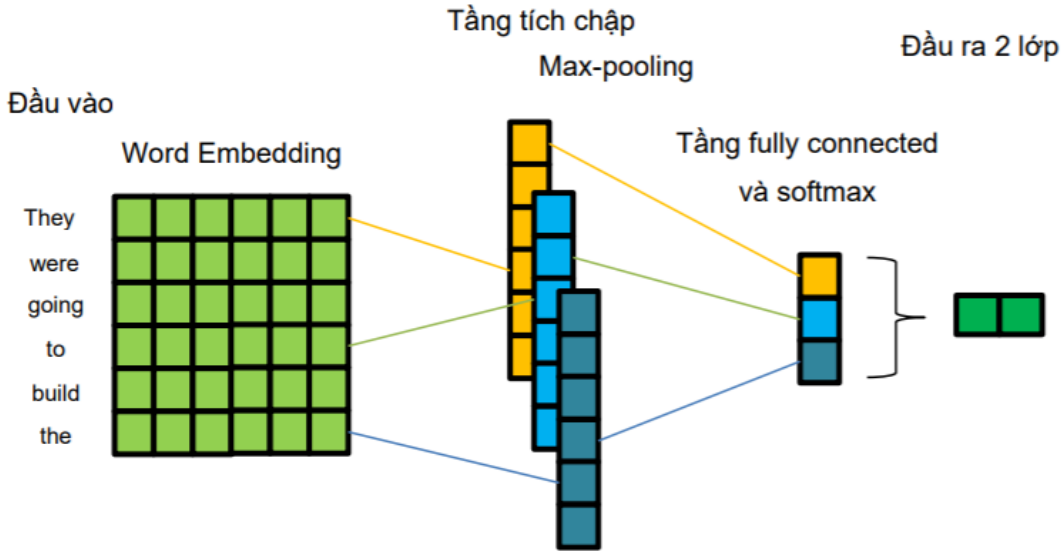
Hình 3.14: Cách tích chập hoạt động với dữ liệu tuần tự.

Thuật toán chập một chiều bắt đầu bằng cách lấy một vùng (patch) hay cửa sổ (window) của các đặc trưng đầu vào bằng với kích thước của bộ lọc (filter

kernel). Các giá trị trong patch này được nhân chập với các trọng số tương ứng của bộ lọc, tính tổng và trở thành giá trị đặc trưng đầu ra. *CNN* một chiều là bất biến đối với các bản dịch, có nghĩa là các chuỗi nhất định có thể được nhận ra ở một vị trí khác nhau. Điều này có thể hữu ích cho các bài toán nhận diện mẫu trong văn bản.

Một mạng thần kinh tích chập thường gồm bốn thành phần chính: đầu vào, tầng tích chập, tầng kết nối dày đặc (fully connected) và đầu ra. Chi tiết mô hình mạng cho bài toán phân loại nhị phân đơn giản được mô tả như Hình 3.15.

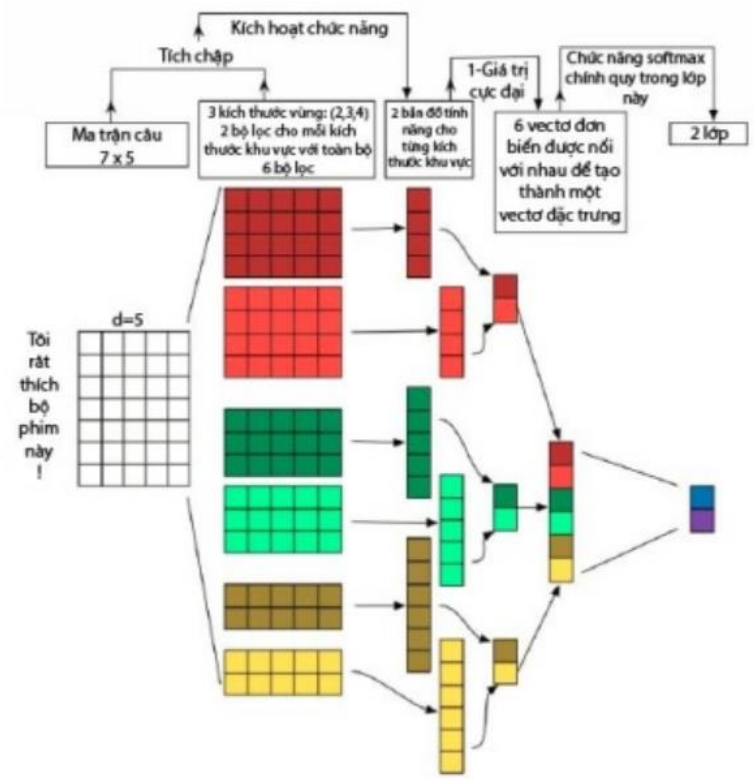
- Đầu vào: dữ liệu đã tiền xử lý được đưa vào lớp nhúng từ để ánh xạ các từ sang một không gian vector ngữ nghĩa tương ứng.
- Tầng tích chập: thực hiện việc nhân tích chập để rút trích đặc trưng. Ở đây, các hàm kích hoạt như *ReLU* và *max-pooling* được sử dụng để loại bỏ các đặc trưng không tốt nhằm giảm thời gian tính toán và tránh hiện tượng mô hình quá khớp (over-fitting).
- Tầng fully connected: Các đặc trưng được nối kết với nhau qua phép concatenation. Hàm kích hoạt như softmax được áp dụng để dự đoán kết quả.
- Đầu ra: trả về kết quả dự đoán.



Hình 3.15: Kiến trúc mạng thần kinh tích chập cho bài toán phân loại nhị phân đơn giản, ảnh trích từ [37].

Giống như các mô hình mạng thần kinh khác, *CNN* cũng áp dụng cơ chế lan truyền ngược để cập nhật trọng số. Các trọng số cần cập nhật ở đây chính là các ma trận tích chập rút trích đặc trưng.

Về cơ bản, kiến trúc này là cách tiếp cận dựa trên mô hình ngôn ngữ **n-grams** để trích xuất các tính năng cấp cao hơn. Số lượng **grams** hay **n** ở đây chính là kích thước của bộ lọc (kernel size). Hình 3.16 là ví dụ minh họa cho sự kết hợp 2-3-4-grams với hai bộ lọc cho mỗi gram trong mô hình *CNN*. Các kết quả tích chập được đưa qua một lớp *pooling* để lấy ra những đặc trưng nổi bật, như ví dụ trên lấy *max-pooling* cho mỗi gram. Sau đó, các đặc trưng của mỗi gram được nối lại bằng phép *concatenation* tạo thành một vector đặc trưng rồi tiếp tục áp dụng hàm softmax để dự đoán đầu ra.



Hình 3.16: Kiến trúc *CNN* với các kernel 2, 3, 4 tương ứng 2-3-4-grams, ảnh trích từ [37].

Nhìn chung, mạng thần kinh tích chập có hiệu quả trong nhiều bài toán NLP nhờ vào khả năng khai thác các manh mối ngữ nghĩa trong các cửa sổ theo ngữ

cảnh. Tuy nhiên, mô hình này vẫn gặp khó khăn khi xử lý các văn bản dài và dữ liệu có yêu cầu cao về tính thứ tự.

3.3.7. Mô hình Transformer và cơ chế chú ý

Các mạng nơ-ron hồi quy có hiệu quả khi xử lý với dữ liệu tuần tự nhưng với những chuỗi dài thì hiệu suất giảm rõ rệt do vấn đề đạo hàm bị triệt tiêu như đã trình bày ở mục 3.3.1. Vấn đề đặt ra là cần có một kỹ thuật có thể xử lý các chuỗi dài mà vẫn đảm bảo tính hiệu quả. Phần này, sinh viên sẽ trình bày cơ chế chú ý (còn gọi là cơ chế tập trung) với mô hình Transformer loại bỏ hoàn toàn các mạng hồi quy mà vẫn đảm bảo hiệu quả.

3.3.7.1. Cơ chế chú ý

Cơ chế chú ý hay cơ chế tập trung (*Attention mechanism*) [22] sử dụng trong mô hình hóa chuỗi lần đầu được giới thiệu vào năm 2015 cho bài toán dịch máy với mô hình đề xuất thuộc họ mô hình mã hóa – giải mã (*encoder – decoder*). Bộ mã hóa đọc toàn bộ chuỗi đầu vào và mã hóa chuỗi này thành một vector có kích thước cố định (fixed-length vector) chứa toàn bộ thông tin của chuỗi. Bộ giải mã sẽ dự đoán kết quả dịch nghĩa hoàn toàn dựa vào vector mã hóa này. Theo đó, mạng nơ-ron cần phải "nén" tất cả các thông tin từ câu đầu vào thành một vector đặc trưng. Tuy nhiên, điều này là bất khả thi, đặc biệt với những chuỗi dài và dài hơn chuỗi dài nhất trong tập huấn luyện. ***Cho*** và cộng sự [23] đã chỉ ra rằng hiệu suất của bộ mã hóa - giải mã cơ bản giảm xuống nhanh chóng khi độ dài của câu đầu vào tăng. Dù có sử dụng *LSTM* hay *GRU* nhưng hai mô hình này vẫn chưa đủ khả năng cải thiện hiệu suất tính toán.

Để giải quyết vấn đề này, một mô hình mở rộng từ kiến trúc mã hóa – giải mã được đề xuất. Cụ thể, mỗi khi tạo từ mới, mô hình sẽ tìm kiếm mềm dẻo (soft search) và chú ý vào những từ có liên quan nhất từ câu nguồn để dự đoán chính xác từ cần tạo ra. Cơ chế chú ý được áp dụng trong quá trình này đã giúp cải thiện đáng kể kết quả dự đoán. Điểm đặc biệt ở đây là cách tiếp cận này không mã hóa cả câu đầu vào thành một vector cố định. Thay vào đó, mô hình sẽ mã hóa câu đầu vào

thành một chuỗi các vector và chọn ra một tập con các vector có ảnh hưởng đến quá trình giải mã nhờ vào cơ chế chú ý.

Ý tưởng chính của cơ chế chú ý: chỉ tập trung vào những thành phần nhất định của dữ liệu đầu vào - những phần có tầm quan trọng đối với kết quả dự đoán tại một thời điểm nhất định.

Không chỉ được sử dụng trong các bài toán NLP như dịch máy, phân tích cảm xúc, tóm tắt văn bản, cơ chế chú ý trong những năm gần đây còn được sử dụng trong nhiều bài toán thị giác máy tính như đặt tiêu đề cho ảnh hay mô tả ảnh… Đối với NLP, cơ chế chú ý có chi phí tính toán không quá nhiều ở cấp độ từ, nhưng sẽ bùng nổ tham số ở cấp độ ký tự. Ngoài ra, so với sự chú ý bằng thị giác của con người hoặc động vật, sự chú ý trong học sâu là khá phản trực giác (counter-intuitive). Sự chú ý của con người sẽ chỉ tập trung vào một thứ và bỏ qua những thứ khác. Nhưng các mô hình học máy cần xét tất cả đầu vào một cách chi tiết trước khi đưa ra quyết định nên tập trung vào phần nào. Có thể thấy, vì đã xét tất cả dữ liệu đầu vào rồi mới quyết định, điều này giống với việc truy cập vào bộ nhớ hơn là sự chú ý. Mặc dù vậy, việc áp dụng cơ chế chú ý vào các mạng học sâu đã và đang đem đến những kết quả đáng mong đợi ở nhiều lĩnh vực vài bài toán khác nhau [24].

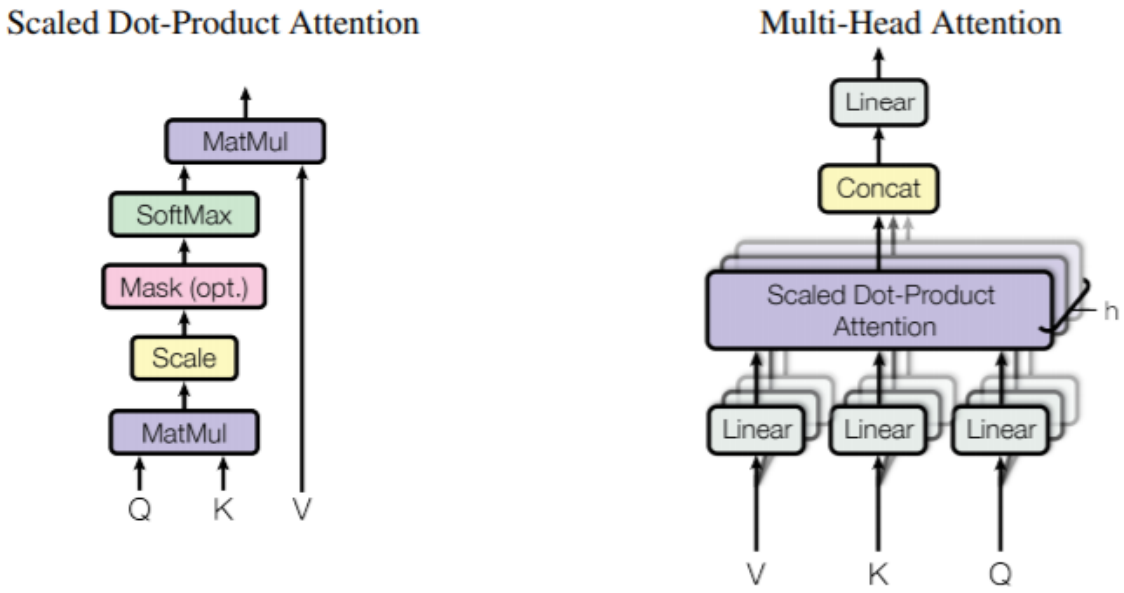
3.3.7.2. Kiến trúc mô hình Transformer

Năm 2017, *Vaswani* và các cộng sự [25] đã tiếp tục đóng góp cho sự thành công của việc áp dụng cơ chế chú ý vào các mô hình học sâu. Các tác giả đã định nghĩa lại khái niệm "chú ý" một cách tổng quát hơn khi dựa vào các khái niệm *keys, query, values*. Họ cũng cung cấp thêm khái niệm *multi-head attention* – một sự cải tiến từ cơ chế chú ý trước đó.

Trước hết hãy định nghĩa thế nào là *"self attention"*. Theo *Cheng* [26], tự chú ý hay chú ý nội bộ – *self attention,* là cơ chế chú ý liên quan đến các vị trí khác nhau trong chuỗi hoặc câu đơn để sự biểu diễn của chuỗi được sinh động hơn.

Trước đó, sự chú ý của một từ w_t được tính bằng cách lấy *dot product* giữa biểu diễn của từ w_t và trạng thái ẩn của từ trước đó w_{t-1}. Trong mô hình *Transformer* này, sự chú ý của từ w_t được tính bằng cách lấy *dot product* của biểu diễn từ cho w_t và tất cả các từ đã thấy trước đó $w_0, w_1 \ldots w_{t-1}$.

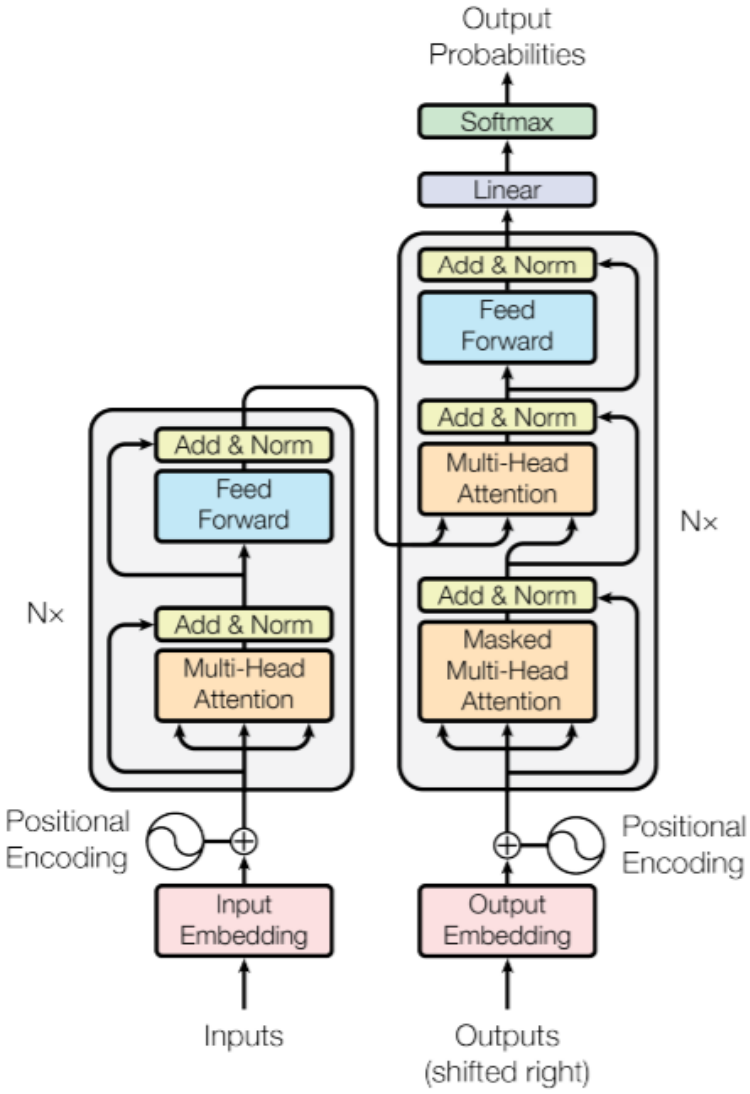
Theo định nghĩa tổng quát, mỗi biểu diễn (embedding) của một từ phải có ba vector riêng, tương ứng là *key*, *query* và *value*. Các vector này chính là các vector nhúng từ trừu tượng ở các không gian nhúng con (subspace) khác nhau. Nếu kích thước của các vector nhúng từ là (D, 1) và muốn vector key có kích thước (D/3, 1) thì nhân hệ số nhúng với ma trận W_k có kích thước (D/3, D). Vì vậy, vector key trở thành $K = W_k * E$. Tương tự, đối với vector *query* và *value*, các phương trình lần lượt là $Q = W_q * E$ và $V = W_v * E$, trong đó E là vector nhúng của bất kỳ từ nào.



Hình 3.17: Scaled Dot-Product Attention (trái) và Multi-Head Attention (phải) gồm vài attention layers chạy song song, ảnh trích từ [25].

Một hàm chú ý được mô tả là sự ánh xạ một *query* và một tập các cặp *key-value* đến đầu ra *output*, trong đó *query*, *key*, *value* và *output* đều là vector. Vector *output* là tổng có trọng số của các vector *value*, trong đó các trọng số được gán cho mỗi *value* được tính bằng hàm tương thích của *query* và *key* tương ứng. Một hàm

chú ý riêng lẻ như vậy được tính qua khối Scaled Dot-Product Attention. Thay vì chỉ thực hiện tính toán một hàm chú ý riêng lẻ, các tác giả cho thấy lợi ích từ việc tính toán ***h*** (trong mô hình này h = 8) lần như vậy cho một query với sự thay đổi về số chiều của các vector này. Sau đó, các kết quả được nối với nhau và đưa ra đầu ra – *Multi-Head Attention*, Hình 3.17.



Hình 3.18: Kiến trúc mô hình *Transformer*, ảnh trích từ [25].

Kiến trúc mô hình *Transformer* dựa trên kiến trúc encoder-decoder truyền thống nên cũng gồm hai khối mã hóa và giải mã như Hình 3.15.

- Khối mã hóa: là một tập hợp N = 6 lớp giống nhau xếp chồng lên nhau, trong đó mỗi lớp lại gồm hai lớp con là một cơ chế *Multi-Head Attention* và một

mạng tiến kết nối dày đặc đơn giản (FLC). Sau mỗi lớp con sẽ là một hàm cộng và chuẩn hóa các vector. Đầu ra của mỗi lớp encoder này là một vector d_{model} = 512 chiều (trong một thí nghiệm, các tác giả đã tăng lên 1024 chiều với 4 lớp Transformer).

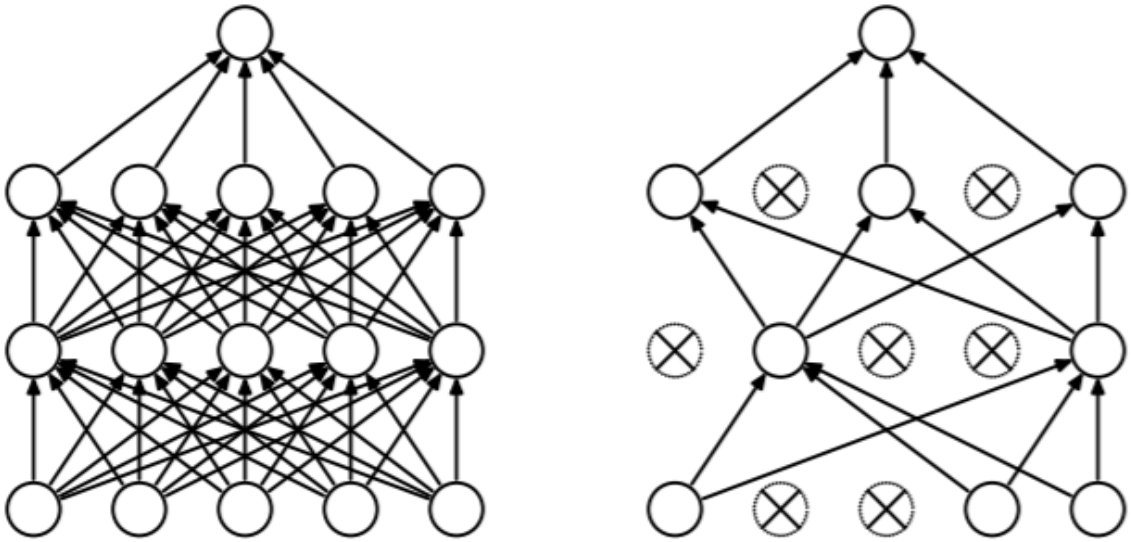
- Khối giải mã: cũng là một tập hợp N=6 lớp giống nhau xếp chồng lên nhau. Điểm khác biệt so với encoder là mỗi decoder lại có thêm một lớp con thứ ba là *Masked Multi-Head Attention*. Cơ chế tự chú này ở hàm này đã được chỉnh sửa bằng cách thêm mặt nạ vào, kết hợp với việc các đầu vào cho khối giải mã được dịch sang phải một đơn vị, sẽ đảm bảo các dự đoán cho vị trí ***i*** chỉ có thể phụ thuộc vào các đầu ra đã biết tại các vị trí nhỏ hơn ***i***.

Kiến trúc mô hình *Transformer* hoàn toàn không sử dụng bất kỳ mạng thần kinh hồi quy hay tích chập nào mà hoàn toàn chỉ dựa vào cơ chế chú ý. Để mô hình có thể tận dụng được thứ tự của chuỗi đầu vào, các tác giả đã thêm vào mỗi token (hiểu nôm na một từ có nghĩa, một từ đã được phân biệt và có một embedding riêng) một số thông tin liên quan hoặc hoàn toàn về vị trí token này, vector này gọi là *positional encoding*. Vector vị trí này có số chiều bằng với số chiều của tập nhúng từ nên có thể tính tổng hai vector này. Với bài toán dịch máy, mô hình Transformer có tốc độ train nhanh hơn và cho kết quả SOTA so với các mô hình dịch máy trước đó.

3.4. Một số kỹ thuật trong đào tạo mạng học sâu

3.4.1. Dropout

Dropout [27] trong mạng thần kinh là kỹ thuật loại bỏ một số đơn vị (*units*) dọc theo các kết nối của nó trong suốt quá trình đào tạo một cách ngẫu nhiên, Hình 3.23. Kỹ thuật này giúp mô hình tránh được hiện tượng quá khớp với dữ liệu huấn luyện vì nếu dùng FCL, các nơ-ron sẽ phụ thuộc mạnh lẫn nhau, làm giảm sức mạnh của mỗi nơ-ron và bị over-fitting. Theo đó, tại mỗi giai đoạn huấn luyện sẽ có ***p%*** số lượng units bị bỏ qua với ***p*** cho trước.



Hình 3.19: Minh họa mô hình mạng học sâu với *Dropout*.

Như thể hiện ở Hình 3.19, hình bên trái là mô hình mạng thông thường với hai lớp ẩn. Hình bên phải là mô hình mạng sau khi áp dụng *Dropout*: những đơn vị (unit) có dấu X bị vô hiệu hóa. Trước đây, *regularization* thường được sử dụng để ngăn chặn hiện tượng mô hình quá khớp bằng cách thêm yếu tố phạt (penalty) vào hàm mất mát, từ đó giúp các đặc trưng giảm sự phụ thuộc lẫn nhau. *Dropout* là một cách tiếp cận khác nhưng có cùng mục đích với *regularization*. Cụ thể, kỹ thuật này được thực hiện như sau:

- Trong pha huấn luyện: chọn ngẫu nhiên *p%* số *unit* (còn gọi là nút – node) và không sử dụng các nút này để tính toán (bỏ qua luôn hàm kích hoạt cho các nút này). Những nút này không có tác dụng trong việc cập nhật trọng số mô hình.
- Trong pha đánh giá: Sử dụng toàn bộ hàm kích hoạt nhưng giảm chúng với tỷ lệ *p* do *p%* hàm kích hoạt đã bị bỏ qua trong quá trình huấn luyện.

Dropout bắt buộc mô hình phải tìm ra các đặc trưng thật sự mạnh mẽ, có tác động quan trọng đến mô hình và phải tốt hơn ngay cả khi kết hợp với các đặc trưng khác. Kỹ thuật này đòi hỏi quá trình huấn luyện diễn ra nhiều lần hơn để mô hình hội tụ, ngược lại, thời gian đạo tạo mỗi epoch sẽ it hơn.

3.4.2. Chuẩn hóa lô (Batch Normalization)

Mạng thần kinh càng sâu thì việc đào tạo càng phức tạp do phân phối của từng lớp đầu vào thay đổi trong quá trình huấn luyện bởi sự thay đổi của các tham số ở các lớp trước đó (trọng số được cập nhật mỗi *mini-batch*[12]). Sự thay đổi trong phân phối đầu vào đến các lớp trong mạng được gọi là dịch chuyển đồng biến nội bộ (*internal covariate shift*). Điều này làm chậm quá trình đào tạo khi yêu cầu tốc độ học thấp và tham số phải được khởi tạo cẩn thận. Chuẩn hóa "lô" (*BatchNormalization* hay *BatchNorm - BN*) [28] là kỹ thuật thường được sử dụng khi huấn luyện các mạng thần kinh rất sâu. Mục đích của kỹ thuật này nhằm giúp ổn định quá trình đào tạo, tăng tốc độ học (*learning rate*) và giảm sự phụ thuộc vào các trọng số được khởi tạo ban đầu. Các bước tính toán cho một *mini-batch* như sau:

- Đầu vào: giá trị của x trên một mini batch B = $\{x_{1..m}\}$; tham số cần học là γ, β
- Đầu ra: tập giá trị $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

Thuật toán chuẩn hóa batch được thực hiện như sau:

Tính giá trị trung bình của mini batch:

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \quad \text{(CT 3.19)}$$

Tính phương sai của mini batch:

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \quad \text{(CT 3.20)}$$

Chuẩn hóa, thêm ϵ để chắc chắn mẫu không âm:

$$\widehat{x_i} \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \text{ (CT 3.21)}$$

Tính phép tỷ lệ (scale) và dịch chuyển (shift):

$$y_i \leftarrow \gamma\widehat{x_i} + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \text{(CT 3.22)}$$

Các tham số gamma và beta được tính như sau:

[12] mini-batch là một tập con thuộc dữ liệu huấn luyện, trong đó mô hình sẽ cập nhật trọng số mỗi khi một tập con này đi qua.

$$\gamma^{(k)} = \sqrt{Var[x^k]} \text{ (CT 3.23)}$$

$$\beta^{(k)} = E[x^k] \text{ (CT 3.24)}$$

BatchNormalization sẽ chuẩn hóa các đặc trưng về trạng thái *zero-mean*[13] với độ lệch chuẩn là 1. Trong đó, *non-zero mean* là hiện tượng dữ liệu không phân bố quanh giá trị 0 mà phần lớn các giá trị này lớn hơn 0 hoặc nhỏ hơn 0. Thêm vào đó, độ lệch chuẩn nếu cao sẽ dẫn đến tình trạng mất cân bằng dữ liệu và làm cho mô hình kém hiệu quả. Mất cân bằng dữ liệu rất phổ biến, cộng thêm việc khởi tạo trọng số ngẫu nhiên ở các lớp (layer), vấn đề sẽ càng nghiêm trọng nếu mô hình càng sâu (có nhiều lớp)[14].

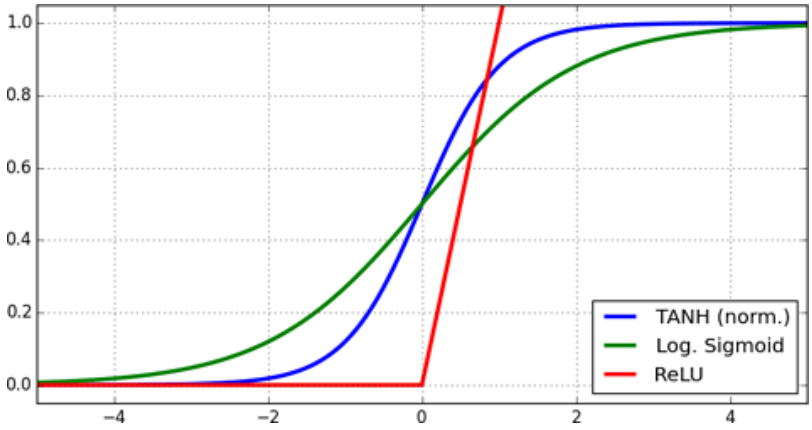
Lấy ví dụ là một công thức của lớp đơn giản như sau: y = (Wx+b). Đạo hàm của y theo w có dạng như sau: dy = dWx. Như vậy, giá trị x (x chính là giá trị của các đặc trưng) ảnh hưởng trực tiếp đến giá trị độ dốc (gradient – độ dốc trong DL không đơn giản như vậy nhưng nguyên lý hoạt động vẫn tương tự). Nếu sử dụng thuật toán giảm giá trị độ dốc (gradient descent) học mô hình thì giá trị độ dốc không nên quá lớn hoặc quá nhỏ. Cho nên, nếu các đặc trưng bị bias (thiên vị, nhiễu) thì mô hình sẽ không ổn định mà kém hiệu quả.

Hơn nữa, mô hình còn chịu ảnh hưởng của các hàm kích hoạt phi tuyến (non-linear activation functions) như sigmoid, tanh, relu. Các hàm này đều có vùng bão hoà[15] (vùng có giá trị đầu ra không đổi dù giá trị đầu vào chênh lệch lớn), Hình 3.20. Theo chiều tiến (forward), khi nhiều giá trị đầu vào rơi vào vùng bão hoà thì nhiều giá trị đầu ra sẽ giống nhau. Như vậy luồng dữ liệu sau đó của model cũng sẽ giống nhau – hiện tượng *covariance shifting*. Theo chiều ngược lại (backward), độ dốc bằng 0 tại vùng bão hoà và các trọng số mô hình không được cập nhật. Như vậy, việc giá trị đặc trưng x quá khác 0 làm cho x có khả năng cao nằm trong vùng bão hoà của hàm kích hoạt và việc học các đặc trưng không có ý nghĩa.

[13] zero-mean là trạng thái khi trung bình của các giá trị bằng 0.

[14] https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/ [Online] Truy cập: 27/08/2020.

[15] https://machinelearningmastery.com/how-to-accelerate-learning-of-deep-neural-networks-with-batch-normalization/ [Online] Truy cập: 27/08/2020.



Hình 3.20: Minh họa miền giá trị của một số hàm kích hoạt phi tuyến tính.

Ở Hình 3.20, tiếp tuyến hyperbol được chuẩn hóa thành [0, 1][16]. Hàm *sigmoid* và *tanh* có vùng bão hòa ở cả hai miền giá trị rất âm và rất dương. Riêng hàm *ReLU*, các đầu vào dương là tuyến tính, các đầu vào âm luôn xuất ra giá trị 0.

BatchNormalization còn được xem như một dạng regularization[17] dùng để phạt (penalty) các giá trị có trọng số gây ảnh hưởng xấu tới quá trình huấn luyện, tránh mô hình quá khớp với dữ liệu huấn luyện (overfitting). Khi dùng BatchNormalization, giá trị các đầu vào sẽ được quy về trong một miền giá trị, làm giảm hiện tượng bùng nổ hoặc biến mất đạo hàm (exploding/vanishing gradient). Chúng ta cũng sẽ không cần phải dùng quá nhiều Dropout, không lo bị mất quá nhiều thông tin. Vì vậy, các giá trị đầu ra của các đơn vị ẩn (hidden unit) trong mạng nơ-ron được đối xử bình đẳng hơn, mô hình không đi theo hướng gây ảnh hưởng tiêu cực cho quá trình học và hàm mất mát (loss function) được kiểm soát tốt hơn.

Có nghiên cứu chỉ ra rằng, việc sử *Dropout* trước *BatchNormalization* sẽ cho kết quả rất tệ, cả trong lý thuyết và thực nghiệm [29]. Theo đó, *Dropout* sẽ thay đổi phương sai của chuyển trạng thái huấn luyện sang kiểm thử nhưng

[16] Nguồn ảnh: https://www.researchgate.net/publication/281289234_Semantic_Image_Segmentation_Combining_Visible_and_Near-Infrared_Channels_with_Depth_Information [Online] Truy cập: 27/08/2020.
[17] https://machinelearningmastery.com/how-to-reduce-overfitting-in-deep-learning-with-weight-regularization/ [Online] Truy cập: 27/08/2020.

BatchNormalization vẫn tích luỹ đầy đủ thông tin trong quá trình huấn luyện. Do *Dropout* làm thay đổi phương sai nên sẽ xảy ra hiện tượng không đồng nhất về phương sai, dẫn đến hành vi suy luận không chắc chắn và không chính xác. Các tác giả đã đề xuất cách kết hợp *Dropout* và *BatchNormalization* bằng cách sử dụng *Dropout* sau khối *BatchNormalization* cuối cùng để tránh hiện tượng "*scaling*" các "*feature-map*" trước các lớp *BatchNormalization*. Cách thứ hai là tinh chỉnh lại công thức của *Dropout* sao cho ít nhạy cảm với phương sai. Vì vậy, khi thực nghiệm cần sử dụng và đánh giá các trường hợp trên bộ dữ liệu đang có để thu được mô hình tốt nhất.

3.5. Phương pháp huấn luyện và đánh giá mô hình

3.5.1. Độ đo mô hình

Theo như cuộc thi yêu cầu, độ đo *Root Mean Square Error* (RMSE)[18] sẽ được sử dụng để đo tính hiệu quả của mô hình dự đoán so với các giá trị thực. Công thức tính như sau:

$$RMSE_{model} = \sqrt{\frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{N}} \quad \text{(CT 3.25)}$$

Trong đó:

- $\hat{y}_i$ là giá trị dự đoán cho mẫu dữ liệu thứ i.
- y_i là giá trị thật của mẫu dữ liệu thứ i.
- N là tổng số lượng mẫu cần đánh giá.

3.5.2. Hàm mất mát

Khóa luận này sẽ sử dụng hai hàm mất mát (loss function) được sử dụng phổ biến nhất trong các bài toán hồi quy để so sánh tính hiệu quả của hai hàm này, đó là *trung bình bình phương lỗi* và *trung bình giá trị tuyệt đối lỗi*.

[18] https://www.tensorflow.org/api_docs/python/tf/keras/metrics/RootMeanSquaredError [Online] Truy cập: 27/08/2020.

- **Trung bình bình phương lỗi** – *Mean Squared Error* (MSE)[19] là tổng bình phương khoảng cách giữa giá trị mục tiêu và giá trị dự đoán, được tính như sau:

$$MSE = \frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{N} \text{ (CT 3.26)}$$

Với:

- $\hat{y}_i$ là giá trị dự đoán cho mẫu dữ liệu thứ i.
- y_i là giá trị thật của mẫu dữ liệu thứ i.
- N là tổng số lượng mẫu cần đánh giá.

- **Trung bình giá trị tuyệt đối lỗi** – *Mean Absolute Error* (MAE)[20] là tổng của sự khác biệt tuyệt đối giữa các giá trị mục tiêu và giá trị dự đoán. MAE đo lường mức độ trung bình của các lỗi trong một tập hợp các dự đoán mà không xét đến hướng, công thức cụ thể như sau:

$$MAE = \frac{\sum_{i=1}^{N}|\hat{y}_i - y_i|}{N} \text{ (CT 3.27)}$$

Với:

- $\hat{y}_i$ là giá trị dự đoán cho mẫu dữ liệu thứ i.
- y_i là giá trị thật của mẫu dữ liệu thứ i.
- N là tổng số lượng mẫu cần đánh giá.

Nhận xét:

- MSE sẽ đánh trọng số cho các dữ liệu ngoại lệ (outliers) cao hơn vì giá trị lỗi ($\hat{y}_i - y_i$ = e)^2 sẽ tăng càng lớn nếu e>1, hay e^2 >> |e|. Vì thế, hiệu suất tổng thể của mô hình sẽ giảm nếu có nhiều ngoại lệ.

[19] https://www.tensorflow.org/api_docs/python/tf/keras/losses/MeanSquaredError [Online] Truy cập: 27/08/2020.
[20] https://www.tensorflow.org/api_docs/python/tf/keras/losses/MeanAbsoluteError [Online] Truy cập: 27/08/2020.

- MAE thì ngược lại, trọng số cho các ngoại lệ được đánh giá tương đương với các dữ liệu còn lại. Tuy nhiên, nhược điểm của MAE là gradient giống nhau trong suốt quá trình huấn luyện, tức là gradient sẽ lớn cho dù giá trị hàm mất mát là nhỏ. Cách giải quyết là sử dụng tốc độ học động: tốc độ học giảm dần khi tiến đến giá trị mục tiêu. Trong khi đó, MSE vẫn có thể hội tụ được dù tốc độ học là cố định. Giá trị gradient của MSE sẽ lớn khi giá trị mất mát lớn và giảm khi giá trị mất mát tiến về 0 khiến cho MSE chính xác hơn vào cuối quá trình huấn luyện.

3.5.3. Hàm tối ưu

Hàm tối ưu *Adam* [30] có độ phức tạp vừa phải, ít yêu cầu bộ nhớ, thích hợp với các bài toán đó độ biến thiên không ổn định và dữ liệu phân mảnh, các siêu tham số biến thiên hiệu quả và yêu cầu ít điều chỉnh. Thêm vào đó, hàm tối ưu này còn kế thừa ưu điểm của các hàm tối ưu khác như *Adadelta*, *RMSprop* và hiện được sử dụng rất rộng rãi trong các mô hình học sâu. Ngoài ra, sinh viên còn sử dụng thêm các hàm *RMSprop*, *amsgrad*, *Nadam*, *Adamax* nhằm mục đích so sánh.

3.5.4. Một số kỹ thuật tối ưu hỗ trợ

Khóa luận này có sử dụng hai kỹ thuật là *EarlyStopping*[21] và *ModelCheckpoint*[22] nhằm giúp mô hình tránh được hiện tượng quá khớp với dữ liệu huấn luyện (overfitting) và ghi nhận lại các trọng số ở thời điểm mô hình cho kết quả tối ưu nhất.

- **EarlyStopping** sẽ theo dõi giá trị được chỉ định (*monitor*) với chế độ theo dõi tùy chọn (*mode*) và chờ trong một số lần lặp nhất định (*patience*). Theo đó, mô hình sẽ cập nhật và ghi nhận giá trị *monitor* theo *mode* và chờ trong một số lần lặp *patience*. Nếu qua số lần lặp *patience* mà *monitor* không cải

[21] https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping [Online] Truy cập: 27/08/2020.
[22] https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ModelCheckpoint [Online] Truy cập: 27/08/2020.

thiện được gì thì sẽ dừng quá trình huấn luyện, tránh cho mô hình quá phức tạp so với dữ liệu cần biểu diễn.

- **ModelCheckpoint** giúp lưu lại các trọng số ở thời điểm mô hình cho kết quả tối ưu nhất. Giống như *EarlyStopping*, kỹ thuật này cũng có các tham số *monitor* và *mode* lần lượt chỉ định giá trị cần quan tâm và chế độ tùy chọn tương ứng. Ngoài ra, *ModelCheckpoint* còn có tham số *save_best_only* cho phép lựa chọn việc chỉ lưu lại mô hình ở thời điểm tốt nhất.

Chương 4. THÍ NGHIỆM VÀ KẾT QUẢ

Trong chương này, sinh viên trình bày quá trình tiền xử lý dữ liệu, phương pháp đánh giá mô hình và tiến hành cài đặt các mô hình học sâu như đã giới thiệu ở chương 3, cụ thể là các mô hình Vanilla RNN, LSTM, GRU, BiLSTM, BiGRU. Tiếp theo đó, sinh viên cài đặt mô hình kết hợp tự đề xuất CNN-BiLSTM-BiGRU và đánh giá các mô hình này trên tập dữ liệu kiểm thử của bộ dữ liệu Humicroedit. Cuối cùng, sinh viên phân tích và đánh giá kết quả của các phương pháp kể trên cho bài toán đánh giá độ hài hước trên câu tiêu đề tin tức đã chỉnh sửa.

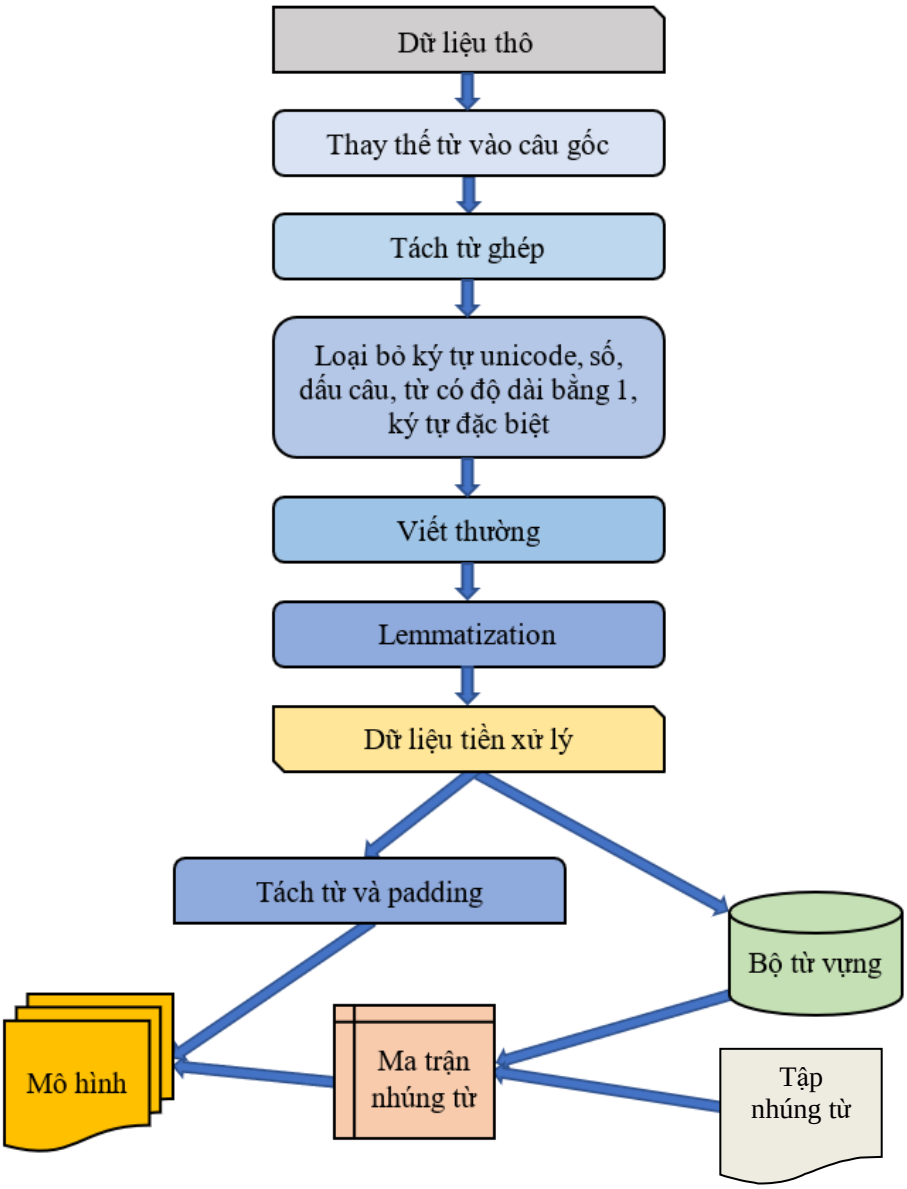
4.1. Thư viện và công cụ sử dụng

Khóa luận này sử dụng *WordNetLemmatizer*, *averaged_perceptron_tagger*, *punct*, *wordnet*, *stopword* của thư viện *nltk*[23] nhằm mục đích loại bỏ đi các dấu câu, một số *stopword* và đưa một từ về dạng từ gốc của nó (*lemmatization*). Riêng đối với *stopword*, đề tài này không sử dụng hết danh sách các từ được thư viện cung cấp. Chi tiết các từ *stopword* được sử dụng ở phần phụ lục A. Ngoài ra, quá trình cài đặt còn sử dụng biểu thức chính quy (*regular expression*) được hỗ trợ bởi thư viện *re* của Python để loại bỏ đi các ký tự unicode nếu có. Để thực hiện việc tách từ (*tokenization*) và thống nhất độ dài câu cho toàn bộ dữ liệu (*padding*), công cụ tách từ cung cấp bởi thư viện *Keras*[24] của *Tensorflow* được sử dụng vì khả năng xử lý mạnh mẽ và tiện lợi. Cuối cùng, các mô hình được xây dựng trên framework của thư viện này.

4.2. Tiền xử lý dữ liệu

Để có thể đưa vào đào tạo mô hình, dữ liệu thô từ bộ dữ liệu **Humicroedit** mà ban tổ chức **SemEval-2020** cung cấp phải được xử lý phù hợp. Các bước thực hiện của quá trình tiền xử lý dữ liệu được mô tả như Hình 4.1.

[23] https://www.nltk.org/ [Online] Truy cập: 27/08/2020.
[24] https://www.tensorflow.org/api_docs/python/tf/keras/ [Online] Truy cập: 27/08/2020.



Hình 4.1: Quá trình tiền xử lý dữ liệu.

Đầu tiên, theo như Hình 4.1, từ được chỉ định ở câu tiêu đề gốc sẽ được thay thế bằng một từ cho sẵn tương ứng (thay từ được đánh dấu bởi cặp dấu **<…/>** trong cột *original* bằng từ tương ứng ở cột *edit* trong tập dữ liệu). Bước kế tiếp là thực hiện tách các từ ghép thành những từ riêng lẻ, ví dụ như từ "*eye-witness*" sẽ được tách thành từ "*eye*" và từ "*witness*". Sau đó, thực hiện loại bỏ các ký tự unicode, con số, dấu câu, từ có độ dài bằng 1, ký tự đặc biệt. Vì các tiêu đề trong bộ dữ liệu được viết hoa-thường một cách tùy ý nên quá trình tiền xử lý sẽ thống nhất chuyển tất cả

bộ dữ liệu về dạng viết thường. Tiếp theo, những từ là *stopword* sẽ được loại bỏ, danh sách sách *stopword* được đính kèm ở phần phụ lục A. Cuối cùng, quá trình chuyển một từ về dạng từ gốc của chính nó (*lemmatization*) sẽ được thực hiện. Kết thúc quá trình này sẽ thu được tập dữ liệu gồm các câu tiêu đề đã được chỉnh sửa và xử lý.

Mô hình học sâu hiện tại không làm việc trực tiếp với từ nên bộ dữ liệu sau khi xử lý sẽ phải được chuyển về dạng vector. Theo đó, bộ từ vựng sử dụng cho mô hình sẽ được xây dựng từ cả hai tập dữ liệu huấn luyện (training set) và đánh giá (development/validation set). Kế tiếp, thực hiện tách từ (*tokenization*) trên mỗi câu tiêu đề và chuyển các từ này thành các số nguyên tương ứng số thứ tự trong tập từ vựng. Mỗi tiêu đề sẽ tiếp tục được chuẩn hóa về cùng một độ dài (*padding* cuối câu) với độ dài tối đa cho mỗi câu là 20 – đây cũng là độ dài tối đa của một câu tiêu đề trong bộ dữ liệu. Bộ từ vựng thu được cũng sẽ tham gia xây dựng một ma trận nhúng từ với tập pretrained **GloVe** và **Fasttext 300** chiều. Đối với các từ không xuất hiện trong tập từ vựng thì sẽ được thay thế bằng một vector 300 chiều với các phần tử bằng 0. Ma trận nhúng này sẽ được sử dụng trong lớp nhúng (embedding layer) của mô hình.

4.3. Độ đo đánh giá mô hình

Cuộc thi **SemEval-2020** yêu cầu sử dụng độ đo **Root Mean Square Error** (RMSE) để đánh giá hiệu suất của mô hình được xây dựng cho bài toán hồi quy của bài toán *Đánh giá độ hài hước trên câu tiêu đề tin tức đã chỉnh sửa*. Đây cũng chính là độ đo được sử dụng trong khóa luận này.

Ngoài ra, mỗi mô hình với mỗi lần cấu hình khác nhau sẽ được huấn luyện 05 lần để đánh giá tính ổn định và tìm ra các tham số phù hợp nhất cho mô hình đó. Hiệu suất của mô hình tương ứng mỗi cấu hình được tính bằng trung bình cộng của 05 lần đào tạo và độ lệch chuẩn tương ứng (bên cạnh độ lệch chuẩn, người ta còn sử dụng độ lỗi lề - margin error, trong khóa luận này sinh viên sử dụng độ lệch chuẩn vì tính đơn giản).

4.4. Cài đặt các mô hình thí nghiệm

Trong quá trình chọn lựa mô hình, sinh viên đã cài đặt nhiều mô hình khác nhau, trong đó có thể chia ra làm ba loại mô hình là *mô hình RNN đơn, mô hình đa kênh CNN kết hợp RNN* và *Transformer*. Các mô hình *RNN đơn* và *mô hình đa kênh CNN kết hợp RNN* có sử dụng chung một số siêu tham số để phục vụ cho việc đánh giá và so sánh hiệu suất, chi tiết thể hiện trong Bảng 4.1.

Siêu tham số	**Giá trị**
Max sequence length	20
Embedding dim	300
Embedding trainable	True
Epoch	50
Batch size	16, 32, 64
Learning rate	0.001, 0.0001, 0.0002, 0.0005
Loss function	MSE, MAE
Dropout	0.5
Activation	ReLU
RNN units	128
Optimized function	Adam

Bảng 4.1: Các siêu tham số sử dụng chung trong các mô hình học sâu.

Các *mô hình RNN đơn* và *mô hình đa kênh CNN-LSTM-GRU* (sẽ được trình bày trong phần tiếp theo) lần lượt được đào tạo với cấu hình là sự kết hợp của các siêu tham số trong Bảng 4.1. Lớp nhúng từ với số chiều là 300 cho mỗi vector sẽ được phép huấn luyện lại cho phù hợp với bộ dữ liệu **Humicroedit** (*trainable=True*). Hàm tối ưu *Adam* được sử dụng với các tốc độ học (learning rate) như mô tả ở bảng trên, các tham số khác còn lại là *beta_1=0.9*, *beta_2=0.999*,

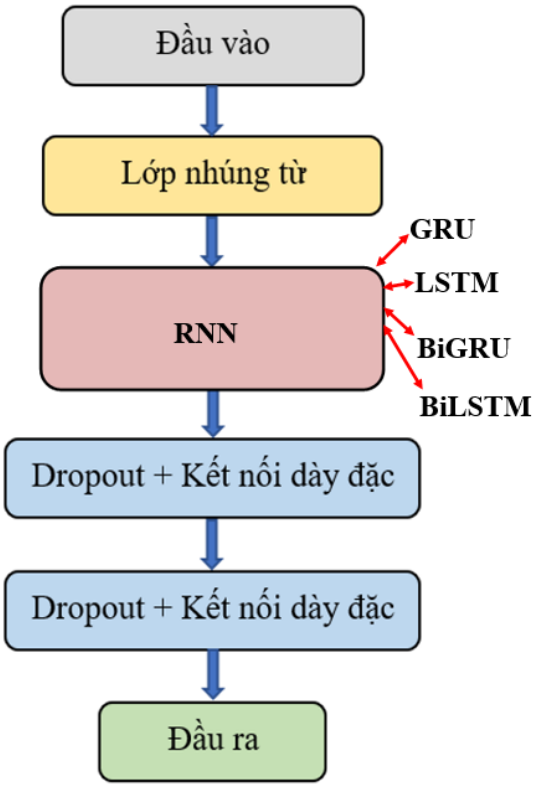
epsilon=1e-07. Hàm kích hoạt được sử dụng ở các lớp là *ReLU* ngoại trừ lớp đầu ra là *linear*.

Số lần lặp (*epoch*) mặc định là 50 nhưng có thể mô hình sẽ dừng sớm hơn do việc sử dụng kỹ thuật dừng sớm như đã trình bày ở mục 3.4.4. *EarlyStopping* sẽ theo dõi giá trị hàm mất mát trên tập dữ liệu đánh giá *monitor=val_loss* với chế độ theo dõi là nhỏ nhất *mode=min* và chờ trong 20 epochs *patient=20*. Kỹ thuật *ModelCheckpoint* cũng sẽ được áp dụng với các thông số như sau: *monitor=val_rmse, mode=min, save_best_only=True*.

Nhận xét: giá trị *val_rmse* chỉ là một thước đo thô về hiệu suất của mô hình ở phiên đào tạo trong khi val_loss cho nhiều thông tin hơn về việc mô hình có đang cải thiện hiệu suất hay không. Chính vì thế, đối với bài toán hồi quy, quá trình đào tạo sẽ dừng khi giá trị hàm mất mát trên tập dữ liệu đánh giá không còn giảm nữa. Trong khi đó, bài toán cần tìm mô hình cho kết quả tốt nhất trên tập dữ liệu kiểm thử và giá trị RMSE trên tập dữ liệu đánh giá là thước đo hiệu quả. Vì thế, mô hình sẽ lưu lại các trọng số nếu giá trị *val_rmse* là tốt nhất ở một thời điểm huấn luyện xác định (với bài toán hồi quy thì RMSE càng nhỏ càng tốt).

4.4.1. Các mô hình RNN đơn

Các *mô hình RNN đơn* gồm có ***Vanilla RNN***, ***GRU***, ***LSTM***, ***BiGRU***, ***BiLSTM***. Kiến trúc mô hình như Hình 4.2.



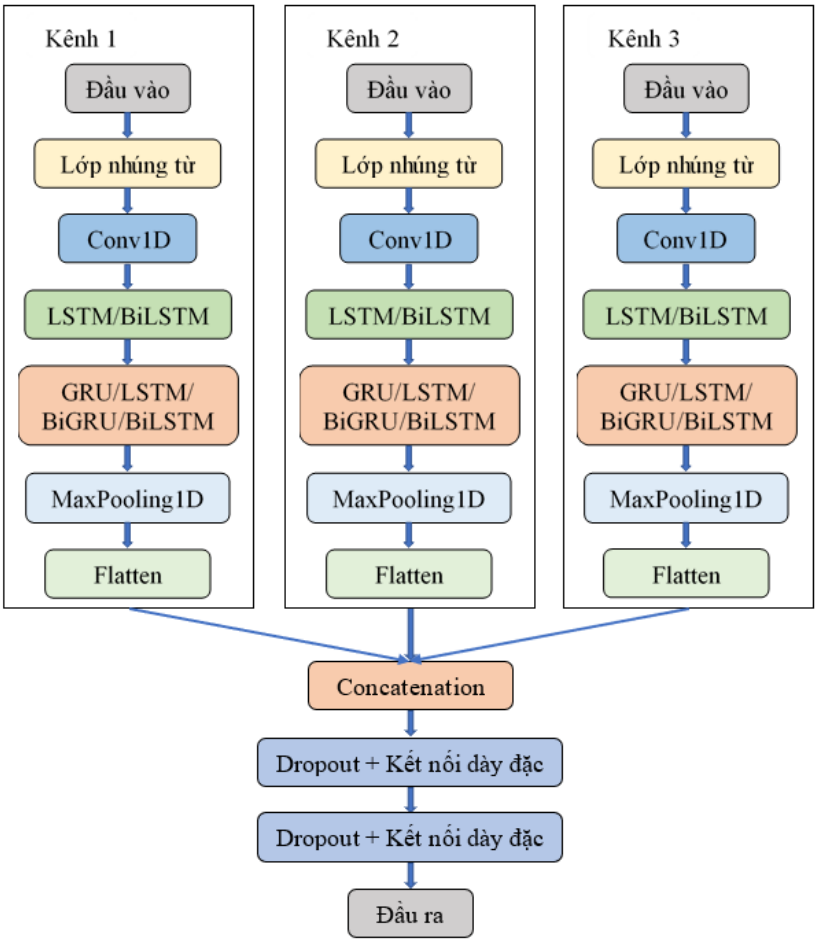
Hình 4.2: Kiến trúc các mô hình Vanilla RNN, GRU, LSTM, BiGRU, BiLSTM.

Các khối ***Vanilla RNN***, ***GRU***, ***LSTM***, **BiGRU**, **BiLSTM** có số đơn vị ẩn (units) là 128. Lớp kết nối dày đặc đầu tiên sử dụng *kernel_initializer='he_uniform'*, trong khi hai lớp còn lại sử dụng giá trị mặc định của thư viện cung cấp là *kernel_initializer='glorot_uniform'*.

4.4.2. Mô hình đa kênh CNN kết hợp RNN

Lấy cảm hứng từ các nghiên cứu liên quan như đã trình bày ở mục 1.6 và một số hướng dẫn sử dụng mô hình đa kênh (multi channels) trong NLP[25,26], sinh viên đề xuất mô hình đa kênh kết hợp giữa *CNN* và các biến thể của *RNN* thành một mô hình có ba kênh, mỗi kênh gồm một lớp *CNN* trích xuất các đặc trưng n-grams theo sau đó là hai lớp *RNN* như Hình 4.3.

[25] https://www.kaggle.com/thebrownviking20/n-gram-multichannel-bilstm-cnn-embed1h [Online] Truy cập: 27/08/2020.
[26] https://machinelearningmastery.com/develop-n-gram-multichannel-convolutional-neural-network-sentiment-analysis/ [Online] Truy cập: 27/08/2020.



Hình 4.3: Kiến trúc mô hình đề xuất kết hợp giữa CNN và các biến thể của RNN.

Theo như mô tả ở Hình 4.3, có bốn mô hình được đề xuất: ***CNN-LSTM-GRU***, ***CNN-LSTM-LSTM***, ***CNN-BiLSTM-BiGRU***, ***CNN-BiLSTM-BiLSTM***. Các mô hình đề xuất này gồm có ba kênh với đầu vào giống nhau. Đầu tiên, mô hình ***CNN-LSTM-GRU*** sử dụng các siêu tham số ở Bảng 4.1 và 4.2 dưới đây để huấn luyện mô hình nhằm so sánh với các *mô hình RNN đơn*.

Như thể hiện trong Bảng 4.2, ***mô hình đa kênh CNN-LSTM-GRU*** sử dụng số bộ lọc (*filters*) cho cả ba kênh là 8, hàm kích hoạt cho lớp *CNN* là *ReLU* và *kernel size* lần lượt là 1, 2, 3 cho mỗi kênh để trích xuất các đặc trưng 1, 2, 3 grams tương ứng. Mỗi lớp *LSTM* và *GRU* tiếp theo sẽ có 128 đơn vị ẩn. Sau đó, mỗi kênh sẽ đi qua lớp *MaxPooling1D* để làm giảm số chiều và chọn ra những đặc trưng nổi bật nhất với pool size là 2, 4, 6 tương ứng với 1, 2, 3 grams. Tiếp theo, các vector

đặc trưng sẽ được làm phẳng khi qua lớp *Flatten* trước khi được gộp lại bằng phép *Concatenation*. Cuối cùng, các đặc trưng đi qua thêm hai lớp kết nối dày đặc trước khi đến lớp dự đoán kết quả.

Siêu tham số	Giá trị
CNN filters	8
CNN activation	ReLU
CNN kernel size	[1, 2, 3]
Pool size	[2, 4, 6]

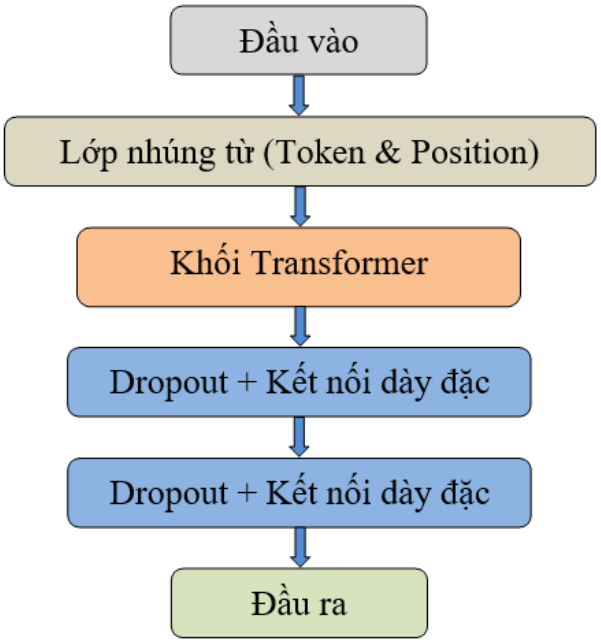
Bảng 4.2: Các siêu tham số sử dụng trong mô hình CNN-LSTM-GRU.

Sau khi đào tạo trên mô hình ***CNN-LSTM-GRU***, sinh viên đã cải tiến bằng cách thay đổi một số tầng, cụ thể là được ba mô hình mới gồm: ***CNN-LSTM-LSTM***, ***CNN-BiLSTM-BiGRU*** và ***CNN-BiLSTM-BiLSTM***. Chi tiết các kết quả cải tiến sẽ được trình bày ở phần phân tích và nhận xét ở mục 4.6.2 và 4.6.3.

4.4.3. Transformer

Bên cạnh các mô hình đã trình bày, sinh viên còn sử dụng thêm mô hình ***Transformer***. Mô hình ***Transformer*** này được chỉnh sửa lại cho phù hợp với bài toán phân loại văn bản và được hướng dẫn cụ thể trên trang chủ của thư viện Keras[27]. Sinh viên thực hiện một số chỉnh sửa nhỏ để áp dụng vào bài toán hồi quy trong khóa luận này. Kiến trúc mô hình được mô tả như Hình 4.4.

[27] https://keras.io/examples/nlp/text_classification_with_transformer/ [Online] Truy cập: 27/08/2020.



Hình 4.4: Kiến trúc mô hình Transformer được chỉnh sửa cho bài toán hồi quy.

Hình 4.4 mô tả kiến trúc mô hình đề ***Transformer*** được chỉnh sửa lại cho phù hợp với bài toán hồi quy trên văn bản. Mô hình ***Transformer*** này có sử dụng thêm một số siêu tham số. Cụ thể, mô hình sử dụng số lớp *attention* (*num_heads*) lần lượt là 2 và 3 kết hợp với số chiều cho lớp ẩn (*ff_dim*) là [16, 32, 64].

4.5. Kết quả thí nghiệm

Ở phần này, sinh viên trình bày kết quả đánh giá trung bình và tốt nhất mà mỗi mô hình đạt được trên tập dữ liệu kiểm thử (test set). Các siêu tham số giúp mô hình cho ra kết quả tốt nhất có thể khác nhau, sinh viên sẽ làm rõ ở mục 4.6. Giá trị RMSE và độ lệch chuẩn (ĐLC) được làm tròn đến 05 chữ số thập phân.

Mô hình	$\mathbf{RMSE_{test}}$ trung bình $\pm$ ĐLC	$\mathbf{RMSE_{test}}$ tốt nhất
Vanilla RNN	0.57086 ± 0.00003	0.57082
GRU	0.54904 ± 0.00216	0.54606
LSTM	0.54702 ± 0.00317	0.54329
BiGRU	0.55286 ± 0.00236	0.55043
BiLSTM	0.54148 ± 0.00015	0.54119
CNN-LSTM-GRU	$\mathbf{0.54079} \pm 0.00091$	0.53966
CNN-LSTM-LSTM	0.53905 ± 0.00048	0.53858
CNN-BiLSTM-BiGRU	0.54078 ± 0.00148	0.53921
CNN-BiLSTM-BiLSTM	$\mathbf{0.53760 \pm 0.00092}$	**0.53630**
Transformer	0.53959 ± 0.00048	0.53925

Bảng 4.3: Kết quả đánh giá của các mô hình đã cài đặt trên tập dữ liệu kiểm thử.

Bảng 4.3 trình bày kết quả tốt nhất của các mô hình đã cài đặt sau khi tối ưu các siêu tham số, trong đó mô hình ***CNN-BiLSTM-BiLSTM*** cho kết quả tốt nhất với giá trị RMSE trên tập dữ liệu kiểm thử là **0.53760**, độ lệch chuẩn là **0.00092**. Trong năm lần huấn luyện, có một lần mô hình thu được kết quả kiểm thử tốt nhất là **0.53630**. Chi tiết kết quả và các thông số đã sử dụng để huấn luyện các mô hình trong Bảng 4.3 được trình bày ở phần phụ lục B và C.

Mô hình đa kênh ***CNN-BiLSTM-BiLSTM*** đã giúp sinh viên đạt được một trong những mục tiêu quan trọng của khóa luận này, đó là có điểm RMSE tốt hơn mô hình "BASELINE" của ban tổ chức và nằm trong nhóm 30 đội có thành tích tốt nhất. Vì lý do không đủ thời gian thực hiện nên sinh viên không được công nhận kết quả ở phiên đánh giá chính thức của cuộc thi (kết thúc vào đầu tháng 3 năm 2020). Tuy nhiên, cuộc thi có một phiên đánh giá phụ (post-evaluation) để giúp các đội tham gia đối chiếu kết quả. Cụ thể, ở phiên đánh giá phụ, kết quả dự đoán từ mô

hình đa kênh ***CNN-BiLSTM-BiLSTM*** của sinh viên xếp hạng thứ **13** trên tổng số **62** đội.

Sub-Task 1 Results (ignore when viewing results for sub-task 2)

#	User	Entries	Date of Last Entry	Team Name	RMSE ▲	RMSE@10 ▲	RMSE@20 ▲	RMSE@30 ▲	RMSE@40 ▲
1	oyx	1	03/23/20		0.50157 (1)	0.80027 (58)	0.68195 (59)	0.60493 (59)	0.54541 (59)
2	Pramodith	20	04/26/20	lmml	0.51695 (2)	0.82450 (57)	0.69790 (58)	0.61897 (58)	0.56023 (58)
3	ZiyangLin	17	07/18/20		0.52054 (3)	0.84302 (52)	0.71518 (54)	0.63145 (56)	0.56844 (57)
4	ttzhang	1	03/12/20	ECNU7	0.52187 (4)	0.85769 (48)	0.72911 (51)	0.64118 (52)	0.57365 (55)
5	YuWang17	21	07/23/20		0.52773 (5)	0.83383 (54)	0.71366 (55)	0.63123 (57)	0.57274 (56)
6	Ferryman	1	03/09/20		0.52776 (6)	0.83953 (53)	0.72031 (53)	0.63722 (53)	0.57503 (54)
7	SparkersTroy	2	08/26/20		0.52811 (7)	0.85553 (49)	0.72805 (52)	0.64118 (51)	0.57747 (52)
8	rtdesetty	50	04/27/20	TCS Research	0.53259 (8)	0.86191 (47)	0.73269 (48)	0.64732 (49)	0.58267 (51)
9	docekal	2	07/08/20	BUT-FIT	0.53336 (9)	0.82605 (56)	0.71130 (56)	0.63497 (54)	0.57611 (53)
10	Farah	3	03/24/20	JUST_Farah	0.53396 (10)	0.90910 (41)	0.76645 (43)	0.66720 (45)	0.59243 (45)
11	theoliao	3	04/26/20		0.53466 (11)	0.85477 (50)	0.73183 (50)	0.64650 (50)	0.58343 (50)
12	Mjason	1	03/11/20	Lunex	0.53518 (12)	0.87481 (46)	0.74673 (46)	0.65606 (47)	0.58769 (49)
13	VoMinhTam	3	08/27/20	NLP@UITVNU	0.53630 (13)	0.88644 (45)	0.74976 (45)	0.65843 (46)	0.58955 (46)
14	kdehumor	13	07/07/20	KdeHumor	0.54084 (14)	0.85344 (51)	0.73185 (49)	0.64800 (48)	0.58798 (48)

Hình 4.5: Kết quả ở phiên đánh giá phụ cho bài toán hồi quy (task1) trên trang chủ CodaLab[28] (ảnh chụp màn hình).

Như thể hiện ở hình 4.5, kết quả này trùng khớp với kết quả mà sinh viên ghi nhận như trong Bảng 4.3. Kết quả ở các cột RMSE@10, RMSE@20... chỉ là các tiêu chí đánh giá bổ sung, không tính vào kết quả xếp hạng.

Nếu so sánh với kết quả ở phiên đánh giá chính thức, kết quả của sinh viên xếp thứ **18** trên tổng số **49** đội, trong đó mô hình BASELINE của ban tổ chức xếp hạng **33**. Cuộc thi này thu hút tổng cộng **379** đội đăng ký tham gia. Bảng 4.4 trình bày kết quả mô hình của sinh viên, mô hình đạt kết quả tốt nhất ở phiên đánh giá chính thức và mô hình BASELINE.

[28] CodaLab là trang web đưa thông tin và ghi nhận kết quả của cuộc thi SemEval-2020. https://competitions.codalab.org/competitions/20970#results [Online] Truy cập: 27/08/2020.

Tên đội	Giá trị $RMSE_{test}$	Hạng	Ghi chú
Hitachi	**0.49725**	1	Phiên chính thức
NLP@UITVNU	**0.53630**	18	Phiên phụ
BASELINE	0.57471	33	Phiên chính thức

Bảng 4.4: Kết quả cuối cùng hệ thống CodaLab ghi nhận.

Theo kết quả từ Bảng 4.4, đội *Hitachi* có kết quả tốt nhất và xếp hạng **1** trong phiên chính thức, mô hình mẫu của ban tổ chức với tên đội là *BASELINE* xếp hạng **33**. Mô hình ***CNN-BiLSTM-BiLSTM*** của sinh viên xếp hạng **18** ở phiên đánh giá phụ.

4.6. Phân tích kết quả thí nghiệm

Ở phần này, sinh viên trình bày chi tiết các thí nghiệm với các mô hình đã cài đặt, đồng thời phân tích tác động của các phương pháp cải tiến đến kết quả của mô hình.

4.6.1. Các mô hình RNN đơn, CNN-LSTM-GRU và Transformer

Sinh viên tiến hành huấn luyện các mô hình cơ bản là ***Vanilla RNN***, ***GRU***, ***LSTM***, ***BiGRU***, ***BiLSTM***, mô hình đa kênh ***CNN-LSTM-GRU*** và mô hình ***Transformer***. Các siêu tham số ở Bảng 4.1 được sử dụng cho các mô hình này. Riêng mô hình ***CNN-LSTM-GRU*** còn sử dụng thêm các siêu tham số ở Bảng 4.2 và mô hình Transformer sử dụng riêng các siêu tham số như mô tả ở phần phụ lục B.

Kết quả từ Bảng 4.3 cho thấy mô hình đa kênh ***CNN-LSTM-GRU*** cho kết quả tốt hơn các mô hình cơ bản kể trên. Điều này là dễ hiểu vì mô hình đa kênh này có kiến trúc phức tạp khi sử dụng kết hợp *CNN* và các biến thể của *RNN*, cho phép học được nhiều đặc trưng tốt hơn. Bên cạnh đó, thí nghiệm còn tìm ra được giá trị tối ưu của một số siêu tham số dùng chung, cụ thể như sau:

- *Learning rate* là **0.001** cho kết quả tốt hơn các thông số còn lại (0.0001, 0.0002, 0.0005).

- *Batch size* bằng **16** giúp mô hình đạt kết quả tốt khi sử dụng kích thước batch bằng 32 hay 64.
- *Loss function* là **MSE** cho kết quả tốt hơn MAE trong tất cả các thí nghiệm.

Chi tiết kết quả các thí nghiệm được trình bày ở phần Phụ lục B. Nhằm giới hạn lại các thí nghiệm về sau, khóa luận sẽ sử dụng các tham số tối ưu hiện tại: *Learning rate = 0.001*, *batch size = 16*, *loss function = Mean Squared Error* cho các thí nghiệm tiếp theo.

4.6.2. Cải tiến từ mô hình CNN-LSTM-GRU

Đầu tiên, sinh viên đánh giá tác động của các giá trị *Dropout* và số lượng *filters* trên mô hình ***CNN-LSTM-GRU*** khi đã cố định các thông số tốc độ học, kích thước batch và hàm mất mát như đã giới hạn ở mục 4.6.1.

- Các giá trị *Dropout* được thay đổi từ **0.5** thành 0.4, 0.3, 0.2, 0.1 cho cả hai lớp. Kết quả thí nghiệm cho thấy việc thay đổi giá trị *Dropout* thường giúp cải thiện chất lượng mô hình. *Dropout* thường giúp mô hình tránh overfitting và tìm ra các đặc trưng mạnh mẽ nhưng nếu tỷ lệ quá cao có thể làm mô hình không học được nhiều đặc trưng. Cụ thể, *Dropout* là **0.2** giúp mô hình cải thiện chất lượng hơn các tỷ lệ còn lại, giảm từ $\boldsymbol{0.54079 \pm 0.00091}$ còn $\mathbf{0.53938 \pm 0.00099}$. Các giá trị *Dropout* còn lại đôi khi làm mô hình tệ hơn, nhưng sự chênh lệch không nhiều. Vì thế, trong thí nghiệm tiếp theo ở phần này, sinh viên sẽ sử dụng cả năm giá trị *Dropout*.
- Sinh viên tiến hành tăng số lượng bộ lọc (filters) của lớp tích chập, từ 8 lên 16, 64 và 128. Kết quả cho thấy hiệu suất mô hình hầu như tăng khi tăng số lượng bộ lọc. Với số bộ lọc bằng 128, mô hình có chỉ số RMSE trung bình là $\boldsymbol{0.53996 \pm 0.00205}$. Nhận thấy rằng việc tăng số lượng filter có xu hướng cải thiện kết quả nhưng không đáng kể, ở thí nghiệm tiếp theo sinh viên sẽ sử dụng số lượng filter là 128.

Tiếp theo, sinh viên tiến hành cải tiến kiến trúc mô hình bằng cách thay đổi các tầng *LSTM* và *GRU*, được ba mô hình mới là ***CNN-LSTM-LSTM***, ***CNN-BiLSTM- BiGRU*** và ***CNN-BiLSTM- BiLSTM***.

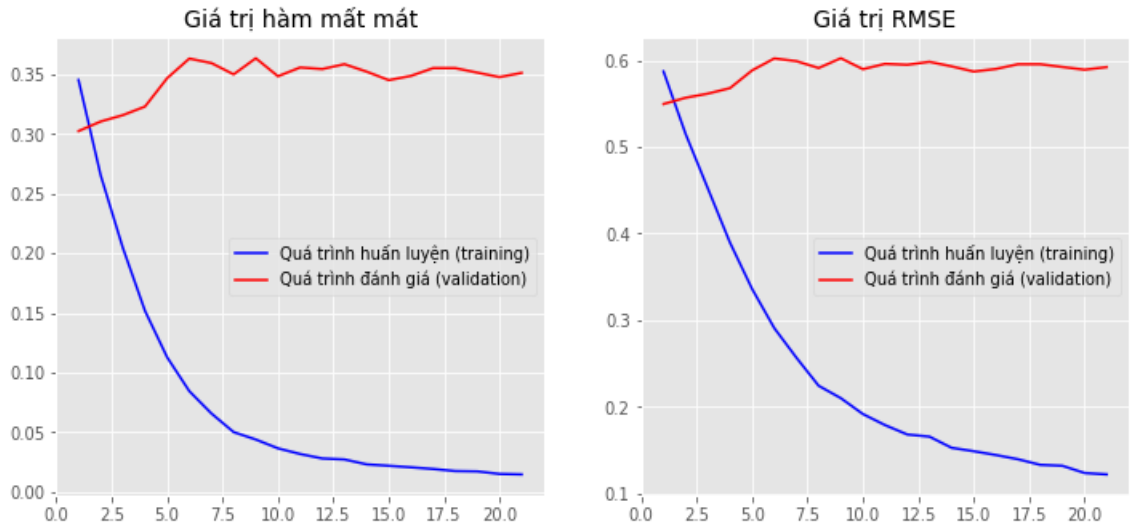
Kết quả thí nghiệm từ Bảng 4.3 cho thấy mô hình có sử dụng *GRU* hoặc *BiGRU* cho kết quả thấp hơn khi sử dụng *LSTM* hoặc *BiLSTM*. Điều này là dễ hiểu vì *LSTM* có kiến trúc phức tạp với số lượng cổng và số đầu vào, đầu ra của một cell nhiều hơn nên sẽ nắm bắt và lưu giữ nhiều thông tin hơn so với *GRU*. Thêm vào đó, *LSTM* hai chiều tỏ ra vượt trội hơn khi đạt $RMSE_{test}$ trung bình tốt nhất là **0.53760** với độ lệch chuẩn là 0.00092. Trong 5 lần đào tạo thì có một lần mô hình ***CNN-BiLSTM-BiLSTM*** đạt RMSE tốt nhất là **0.53630**.

Sau khi thực hiện tăng số lượng filters và thay đổi kiến trúc mô hình như đã trình bày ở trên, sinh viên nhận thấy giá trị *Dropout* bằng 0.2 cho kết quả tốt và ổn định nhất. Để giới hạn lại các thí nghiệm phía sau, sinh viên sẽ chỉ dùng *Dropout* bằng 0.2 cho mô hình tốt nhất hiện tại là ***CNN-BiLSTM-BiLSTM***.

4.6.3. Một số thí nghiệm trên mô hình CNN-BiLSTM-BiLSTM

Từ thí nghiệm ở mục 4.6.2, sinh viên nhận thấy mô hình ***CNN-BiLSTM-BiLSTM*** gặp phải vấn đề overfitting sớm như thể hiện trong Hình 4.5.

Dù gặp vấn đề overfitting sớm nhưng nhờ sử dụng hai kỹ thuật là *EarlyStopping* và *ModelCheckpoint* (mục 3.5.4) nên mô hình vẫn ghi nhận lại được các trọng số tốt nhất. Các mô hình sinh viên đã sử dụng cũng có xu hướng tương tự: giá trị hàm mất mát và RMSE thường đạt giá trị tốt nhất trong khoảng 5 epoch đầu tiên trên tập dữ liệu đánh giá.



Hình 4.6: Giá trị hàm mất mát và RMSE trong quá trình huấn luyện và đánh giá của mô hình ***CNN-BiLSTM-BiLSTM***.

Từ Hình 4.6 có thể thấy rằng, giá trị hàm mất mát giảm dần qua các epoch trong phiên huấn luyện nhưng lại đạt giá trị nhỏ nhất ngay từ epoch đầu tiên và không có cải thiện ở các epoch tiếp theo trong phiên đánh giá, tương tự cho giá trị RMSE.

Để cải thiện chất lượng mô hình, sinh viên tiến hành áp dụng thêm một số phương pháp cải tiến vào mô hình ***CNN-BiLSTM-BiLSTM*** và đánh giá tác dụng của phương pháp này đối với mô hình trên bộ dữ liệu **Humicroedit**. Kết quả thí nghiệm như trình bày trong Bảng 4.5.

- Scale dữ liệu. Các đầu là các vector nhúng từ vào được scale về đoạn [-1, 1] và đầu ra được scale về đoạn [0, 1] với mong muốn quá trình tính toán dễ dàng hơn và các giá trị lớn sẽ ít có tác động đến mô hình so với các giá trị nhỏ. Kết quả sau khi đào tạo không những không được cải thiện mà còn tệ hơn, từ **0.53760** tăng lên *0.54551*.

CNN-BiLSTM-BiLSTM		**$RMSE_{test}$ trung bình**
	Scale đầu vào và đầu ra	0.54551 ± 0.00332
Thay đổi hàm kích hoạt	ReLU → $LeakyReLU_{alpha=0.1}$	0.54083 ± 0.00093
	ReLU → $ELU_{alpha=0.1}$	0.54081 ± 0.00195
Thay đổi hàm tối ưu	Adam → amsgrad	**0.53801** ± 0.00083
	Adam → RMSprop	0.55697 ± 0.00291
	Adam → Nadam	0.56447 ± 0.00502
	Adam → Adamax	0.54826 ± 0.00261
Thêm BatchNormalization	Trước hàm kích hoạt	**0.54037** ± 0.00053
	Sau hàm kích hoạt	0.54253 ± 0.00176
Thay tập nhúng từ	GloVe → Fasttext	0.54467 ± 0.00217
	GloVe → Fasttext subword	0.55576 ± 0.00373

Bảng 4.5: Kết quả một số thí nghiệm trên mô hình CNN-BiLSTM-BiLSTM.

- Thay đổi hàm kích hoạt *ReLU*. Hàm kích hoạt *ReLU* có nhược điểm "dying *ReLU*. Các biến thể như *LeakyReLU*, *ELU* tạo một độ dốc nhỏ ở miền giá trị âm giúp mô hình vẫn học được từ những đầu vào có giá trị nhỏ hơn 0. Ở đây, sinh viên sử dụng hệ số alpha lần lượt là 0.1, 0.2, 0.3, 0.4. Kết quả thí nghiệm cho thấy hai biến thể này không phải lúc nào cũng cho kết quả tốt hơn hàm *ReLU*. Đây cũng là một trong những lý do giúp hàm ReLU được sử dụng phổ biến ở thời điểm hiện tại, không chỉ vì tính đơn giản của hàm số mà còn bởi hiệu quả mà nó mang lại trong nhiều bài toán khác nhau. Có thể thấy rằng, giá trị alpha = 0.1 mang lại kết quả tốt nhất trong số bốn hệ số

alpha đã dùng, nghĩa là alpha càng nhỏ càng tốt, nếu alpha bằng 0 thì nó trở thành *ReLU*. Tuy nhiên, theo hiểu biết của sinh viên, hiện nay chưa có chứng minh nào khẳng định *ReLU* tốt hơn các biến thể của nó và ngược lại. Có thể tùy vào loại mô hình hay bộ dữ liệu mà kết quả mang lại sẽ khác nhau. Cách tốt nhất là sử dụng các hàm này và ghi nhận lại kết quả.

- Thay đổi hàm tối ưu *Adam*. Bên cạnh *Adam* – hàm tối ưu được sử dụng phổ biến hiện nay, các hàm khác như *amsgrad, RMSprop, Nadam, Adamax* cũng được sử dụng và mang lại hiệu quả cao. Sinh viên thử cài đặt các hàm này để so sánh với hàm *Adam*. Learning rate được sử dụng là **0.01**, đây cũng là giá trị mặc định cho các hàm tối ưu này trong thư viện của Tensorflow. Kết quả huấn luyện cho thấy hàm *Adam* vượt trội hơn hẳn khi có RMSE là **0.53630** trong khi kết quả tốt nhất của các hàm được thí nghiệm là amsgrad đạt *0.53801* RMSE. Hàm *amsgrad* chính là một cải tiến từ hàm *Adam*, được giới thiệu vào năm 2018 sau bốn năm hàm *Adam* được công bố. Hiện tại, hàm *Adam* vẫn đang cho kết quả tốt trên nhiều bài toán khác nhau, cụ thể là bài toán hồi quy trong khóa luận này.

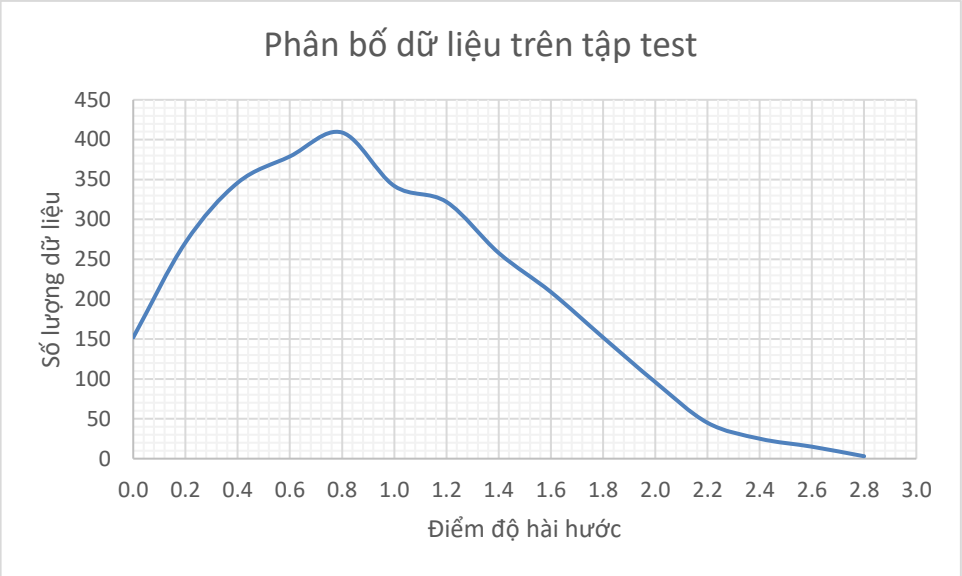
- Thêm *BatchNormalization*. Đây là phương pháp giúp chuẩn hóa các đặc trưng về trạng thái "zero-mean" với phương sai là 1. *BatchNormalization* có tác dụng giúp cho việc tính đạo hàm được ổn định, sử dụng learning rate lớn hơn làm tăng tốc quá trình huấn luyện, tránh hiện tượng các đặc trưng rơi vào khoảng bão hòa và còn giúp giảm overfitting. Theo tác giả của *BatchNormalization*, lớp chuẩn hóa này nên được sử dụng ngay trước hàm kích hoạt ReLU của lớp hiện tại. Tuy nhiên, có một cuộc tranh luận diễn ra khi kết quả huấn luyện nếu sử dụng *BatchNormalization* sau hàm kích hoạt là cao hơn[29]. Ở đây, sinh viên cài đặt cả hai trường hợp. Kết quả thí nghiệm cho thấy đặt *BatchNormalization* trước hàm kích hoạt ReLU cho kết quả tốt hơn nhưng nó vẫn không tốt bằng việc không sử dụng *BatchNormalization*!

[29] https://github.com/gcr/torch-residual-networks/issues/5 [Online] Truy cập: 27/08/2020.

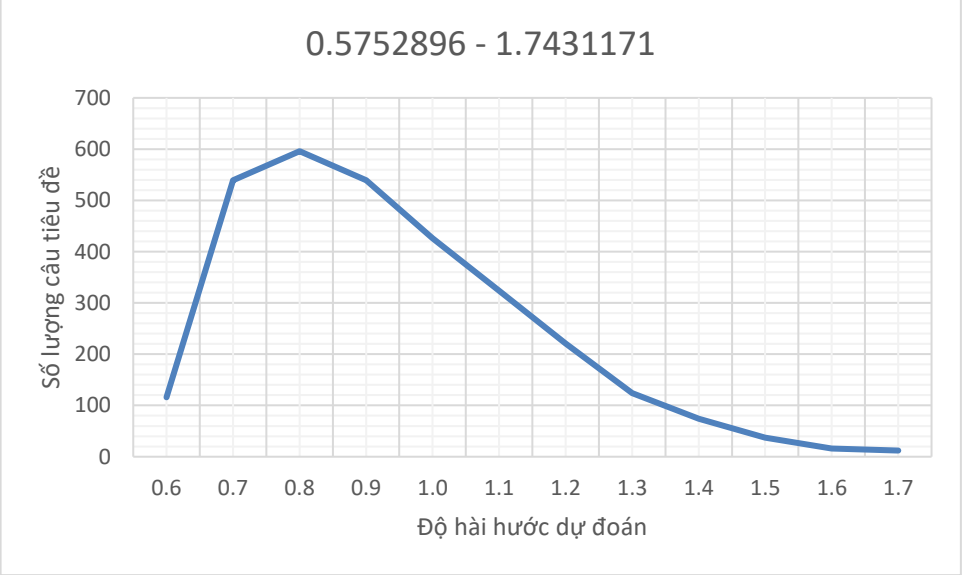
Nguyên nhân có thể là do lớp nhúng từ *GloVe* đã được scale về trạng thái *zero-mean* và bộ dữ liệu **Humicroedit** phù hợp với việc sử dụng hàm *ReLU* mà không cần chuẩn hóa thêm. Thông thường nên đặt batch size từ 32 trở lên khi sử dụng BatchNormalization. Tuy nhiên, với bộ dữ liệu này batch size là 32 hoạt động không hiệu quả.

- Thay tập nhúng từ *GloVe* bằng *Fasttext*. Ở thí nghiệm này, sinh viên sử dụng tập nhúng từ được xây dựng sẵn *Fasttext* trong hai trường hợp là có và không có huấn luyện với subword. Kết quả thí nghiệm cho thấy tập nhúng từ *Fasttext* không được đào tạo với subword cho kết quả tốt hơn tập nhúng từ có đào tạo với subword. Tuy nhiên nó vẫn không tốt bằng tập *GloVe* với kết quả ban đầu. Kết quả này khá bất ngờ khi *GloVe* được giới thiệu vào năm 2014 và được xây dựng đựa trên ma trận đồng xuất hiện, các bộ dữ liệu sử dụng được thu thập từ các năm trước. Trong khi đó, *Fasttext* sử dụng trong khóa luận này được công bố vào năm 2018 và được huấn luyện các nguồn dữ liệu từ năm 2017. Bộ dữ liệu **Humicroedit** được thu thập vào năm 2017-2018. Xét về miền giá trị dữ liệu thì tập *Fasttext* phải cho kết quả tốt hơn. Có thể bộ dữ liệu **Humicroedit** phù hợp với phương pháp xây dựng của *GloVe* hơn nên kết quả tốt nhất mà mô hình đạt được lại trên tập *GloVe*.

Tóm lại, mô hình đa kênh **CNN-BiLSTM-BiLSTM** là mô hình cho kết quả tốt nhất với RMSE trung bình là **0.53806 ± 0.00164**, trong đó có một lần mô hình đạt RMSE **0.53603** cũng là kết quả tốt nhất mà sinh viên ghi nhận được. Tuy nhiên, mô hình này vẫn còn hạn chế khi tình trạng mất cân bằng dữ liệu chưa được giải quyết triệt để.



Hình 4.7: Biểu đồ phân bố độ hài hước trên tập dữ liệu kiểm thử.



Hình 4.8: Biểu đồ phân bố độ hài hước dự đoán của mô hình CNN-BiLSTM-BiLSTM.

Từ Hình 4.7 có thể thấy rằng độ hài hước dự đoán của mô hình ***CNN-BiLSTM-BiLSTM*** có biểu đồ tương tự như biểu đồ độ hài hước trên toàn bộ dữ liệu (Hình 2.1). Tuy nhiên, giá trị dự đoán bị co lại, tập trung vào độ hài hước tiềm năng. Đặc biệt, ở miền giá trị độ hài hước cao (khoảng từ 2 đến 3) đã bị co lại hoàn

toàn, độ hài hước cao nhất mà mô hình dự đoán chỉ gần 1.75. Các giá trị dự đoán được làm tròn về một chữ số thập phân để việc minh họa được dễ dàng hơn.

Chương 5. XÂY DỰNG ỨNG DỤNG DEMO

Phần demo này nhằm đánh giá chất lượng mô hình ***CNN-BiLSTM-BiLSTM*** đã xây dựng. Để chạy được chương trình, máy tính cần cài đặt:

- Python phiên bản 3.7.5 trở lên;
- Tensorflow phiên bản 2.3.0 trở lên;
- Thư viện nltk phiên bản 3.5 trở lên;
- Thư viện PyQt5 phiên bản 5.15.1 trở lên;
- Thư viện numpy phiên bản 1.19.2.

Để mở chương trình demo, người dùng chạy trực tiếp file Demo.py đã built bằng lệnh sau trong command line:

python Demo.py **hoặc** *Demo.py*

Sau đó, một chương trình có giao diện như Hình 5.1 sẽ mở ra.

Hình 5.1: Demo kiểm tra mô hình.

Người dùng ***nhập vào một câu tiêu đề đã chỉnh sửa*** ở ô nhập dữ liệu, sau đó ***nhấn nút Đánh giá/Assess***. Kết quả dự đoán độ hài hước cho câu tiêu đề này sẽ được hiển thị ngay bên cạnh. Trong ví dụ trên Hình 5.1, câu tiêu đề đã được chỉnh sửa "Google Cancels Town Hall After Employee 's Controversial comedy Memo" được mô hình dự đoán có điểm hài hước là **0.6545325**, khá gần với điểm thật là **0.6**. Người dùng có thể nhập vào một câu khác và bắt đầu quá trình dự đoán. Một số mẫu dữ liệu dùng cho chương trình demo như trong Bảng 5.1.

Câu tiêu đề đã chỉnh sửa	**Điểm dự đoán của mô hình**	**Điểm thực**
Google Cancels Town Hall After Employee 's Controversial comedy Memo	0.6545325	0.6
Trump looks at partial eclipse … without toupee .	1.7431171	1.8
CNN 's Jake Tapper to wrestle Paul Ryan following retirement announcement	1.1651349	2.8
Syrian regime driver vows to drive out US from country , state media says	0.6303517	0

Bảng 5.1: Mẫu dữ liệu demo.

Lưu ý: khi chương trình mới chạy lần đầu có thể mất nhiều thời gian để cài đặt thư viện và nạp mô hình. Những lần dự đoán sau đó chương trình thực hiện nhanh hơn và hầu như không có độ trễ. Source code chương trình sẽ được đính kèm theo khóa luận này.

Chương 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Chương này tổng kết lại kết quả đã đạt được và nêu lên một số hướng phát triển, cải thiện cho khóa luận trong tương lai.

6.1. Kết luận

Ở khóa luận này, sinh viên tập trung tìm hiểu các khái niệm, tính chất của sự hài hước và các phương pháp để giải quyết bài toán đánh giá sự hài hước trên câu tiêu đề tin tức đã chỉnh sửa. Thông qua kết quả được trình bày ở chương 4, khóa luận đã đạt được mục tiêu đề ra khi mô hình sinh viên đề xuất cho kết quả tốt hơn mô hình BASELINE của ban tổ chức cuộc thi và xếp thứ 18 nếu tính theo kết quả chính thức, xếp thứ 13 theo phiên đánh giá phụ.

Ngoài ra, trong quá trình xây dựng và đánh giá các mô hình, sinh viên đã tìm hiểu một số mô hình học sâu phổ biến hiện nay, tiến hành cài đặt và thí nghiệm nhiều lần để đánh giá sự ổn định của các mô hình này. Song song đó, sinh viên còn thay đổi nhiều siêu tham số và thuộc tính để đánh giá tác động của những siêu tham số, thuộc tính này đến hiệu suất của mô hình.

Cuối cùng, sinh viên đã xây dựng một chương trình ứng dụng demo trên desktop phục vụ cho bài toán đánh giá độ hài hước trên câu tiêu đề tin tức đã chỉnh sửa.

6.2. Hướng phát triển

Mặc dù khóa luận này đã đạt mục tiêu đề ra nhưng sinh viên nhận thấy vẫn còn tiềm năng phát triển thêm.

- Trong quá trình giải quyết bài toán hồi quy đánh giá độ hài hước trên câu tiêu đề tin tức đã chính sửa, sinh viên chỉ sử dụng câu tiêu đề đã chỉnh sửa để xây dựng mô hình. Theo thông báo từ ban tổ chức, việc chỉ sử dụng câu đã chỉnh sửa đôi khi mang lại kết quả tốt hơn nếu dùng cả hai câu. Tuy nhiên, việc áp dụng cả hai câu cho phép mô hình có khả năng so sánh và học được thêm nhiều đặc trưng tốt hơn.

- Khóa luận này cho kết quả tốt nhất trên mô hình kết hợp giữa *CNN* và các biến thể của *RNN*. Mặc dù sinh viên có sử dụng kiến trúc mô hình Transformer, chỉnh sửa lại cho phù hợp với bài toán hồi quy nhưng mô hình này không cải thiện được kết quả. Tuy nhiên, nhiều cải tiến từ mô hình *Transformer* như *BERT* [31], *RoBERTa* [32], *XLNet* [33] hay gần đây nhất là *DeeBERT* [34] hứa hẹn sẽ cải thiện kết quả bài toán vì các mô hình này được huấn luyện trên các bộ dữ liệu lớn và trong thời gian gần đây, phù hợp với miền dữ liệu của bộ dữ liệu Humicroedit. Các mô hình này là những thể hiện của phương pháp học chuyển đổi (Transfer learning) – vốn mang lại nhiều kết quả SOTA cho nhiều bài toán, không chỉ trong NLP mà còn ở nhiều lĩnh vực khác như thị giác máy tính. Bên cạnh đó, các mô hình học tập đối nghịch (Adversarial Networks) [35] có nguồn gốc từ *GANs* (Generative Adversarial Networks) cũng có khả năng cải thiện hiệu suất mô hình cho bài toán hồi quy trên văn bản.
- Bộ dữ liệu Humicroedit có sự mất cân bằng khá lớn. Ví dụ, trong toàn bộ dữ liệu với hơn 15095 câu tiêu đề thì chỉ có 5 câu đạt điểm tuyệt đối là 3.0, các giá trị thể hiện cho độ hài hước càng về hai cực (0.0 và 3.0) càng ít, chủ yếu tập trung xung quanh giá trị hài hước tiềm năng (potential meanGrade) là 0.94. Các tác giả của bộ dữ liệu nhận định rằng khó tạo ra những câu tiều đề có độ hài hước lớn, thông thường chỉ có thể chỉnh sửa vào tạo ra câu tiêu đề có mức độ hài hước vừa phải. Giải quyết vấn đề mất cân bằng dữ liệu trong các bài toán hồi quy bao giờ cũng khó hơn các bài toán phân loại, đặc biệt khi rất khó để thêm dữ liệu mới. Một số phương pháp có thể hỗ trợ giải quyết phần nào tình trạng này là *Weighted MSE* [36] hoặc *SMOGN* [37] đáng để xem xét.
- Cuối cùng, việc xây dựng một bộ dữ liệu chuẩn cho tiếng Việt có liên quan đến sự hài hước cũng là điều cần thiết. Việc này không những thúc đẩy các nghiên cứu về xử lý ngôn ngữ mà còn có tính ứng dụng cao trong nhiều lĩnh vực của đời sống.

TÀI LIỆU THAM KHẢO

Danh mục tài liệu tham khảo tiếng Anh

[1] J. K. M. G. Nabil Hossain, ""President Vows to Cut< Taxes> Hair": Dataset and Analysis of Creative Text Editing for Humorous Headlines," in *Proceedings of NAACL-HLT*, Minneapolis, Minnesota, 2019.

[2] T. C. Zsolt Bitvai, "Non-linear text regression with a deep convolutional neural network," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, Beijing, China, 2015.

[3] M. S. Neşat Dereli, "Convolutional Neural Networks for Financial Text Regression," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, Florence, Italy, 2019.

[4] T. R. L. Ø. Stig Johan Berggren, "Regression or classification? Automated Essay Scoring for Norwegian," in *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, Florence, Italy, 2019.

[5] A. R. A. R. Peter Potash, "SemEval-2017 Task 6: #HashtagWars: Learning a Sense of Humor," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017) @2017 Association for Computational Linguistics (ACL)*, Vancouver, Canada, 2017.

[6] M. Z. Rutal Mahajan, "SVNIT @ SemEval 2017 Task-6: Learning a Sense of Humor Using Supervised Approach," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017) - @2017 Association for Computational Linguistics (ACL)*, Vancouver, Canada, 2017.

[7] M. S. J. B. J. F. S. B. D. M. Christopher Manning, "The Stanford CoreNLP natural language processing toolkit," in *Proceedings of 52nd Annual Meeting*

of the Association for Computational Linguistics: System Demonstrations, Baltimore, Maryland, 2014.

[8] J. K. L. V. E. H. H. K. Nabil Hossain, "Filling the Blanks (hint: plural noun) for Mad Libs Humor," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP-ACL)*, Copenhagen, Denmark, 2017.

[9] A. Tomoioagă, "A Linguistic Analysis of Jokes," in *Discourse As A Form Of Multiculturalism In Literature And Communication, Section: Language And Discourse Arhipelag Xxi Press, Tîrgu Mureş*, 2015.

[10] G. Ritchie, The Linguistic Analysis of Jokes, London: Routledge, 2005.

[11] J. L. O. C. RIM Dunbar, "The complexity of jokes is limited by cognitive constraints on mentalizing," *Human Nature*, vol. 27, pp. 130-140, 2016.

[12] R. S. C. D. M. Jeffrey Pennington, "GloVe: Global Vectors for Word Representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014.

[13] K. C. G. C. J. D. Tomas Mikolov, "Efficient Estimation of Word Representations in Vector Space," in *International Conference on Learning Representations (ICLR)*, 2013.

[14] E. G. A. J. T. M. Piotr Bojanowski, "Enriching Word Vectors with Subword Information," in *Transactions of the Association for Computational Linguistics*, 2017.

[15] E. G. P. B. C. P. A. J. Tomas Mikolov, "Advances in Pre-Training Distributed Word Representations," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, 2018.

[16] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network," *Physica D: Nonlinear Phenomena,* vol. 404, p. 132306, 2020.

[17] C. G. K. C. Y. B. Junyoung Chung, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *NIPS 2014 Deep Learning and Representation Learning Workshop,* 2014.

[18] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural Network,* vol. 1, p. 119–130, 1989.

[19] "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE,* vol. 86, pp. 2278--2324, 1998.

[20] J. W. L. B. M. K. K. K. P. K. Ronan Collobert, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research,* vol. 12, pp. 2493--2537, 2011.

[21] Y. Kim, "Convolutional Neural Networks for Sentence Classification," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP),* p. 1746–1751, 2014.

[22] K. C. Y. B. Dzmitry Bahdanau, "Neural Machine Translation by Jointly Learning to Align and Translate," *International Conference on Learning Representations (ICLR),* 2015.

[23] B. v. M. C. G. D. B. F. B. H. S. Y. B. Kyunghyun Cho, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP),* p. 1724–1734, 2014.

[24] M. L. P. T. Andrea Galassi, "Attention in Natural Language Processing," *arXiv:1902.02181v2,* 2020.

[25] N. S. N. P. J. U. L. J. A. N. G. Ł. K. I. P. Ashish Vaswani, "Attention Is All You Need," *Advances in Neural Information Processing Systems (NIPS)*, pp. 5998--6008, 2017.

[26] L. D. M. L. Jianpeng Cheng, "Long Short-Term Memory-Networks for Machine Reading," *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 551-561, 2016.

[27] G. H. A. K. I. S. R. S. Nitish Srivastava, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *The journal of machine learning research*, vol. 15, pp. 1929-1958, 2014.

[28] C. S. Sergey Ioffe, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv preprint arXiv:1502.03167*, 2015.

[29] S. C. X. H. J. Y. Xiang Li, "Understanding the Disharmony between Dropout and Batch Normalization by Variance Shift," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2682--2690, 2019.

[30] J. B. Diederik P. Kingma, "Adam: A method for stochastic optimization," *International Conference on Learning Representations (ICLR)*, 2015.

[31] M.-W. C. K. L. K. T. Jacob Devlin, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, vol. Volume 1 (Long and Short Papers), p. 4171–4186, 2019.

[32] M. O. N. G. J. D. M. J. D. C. O. L. M. L. L. Z. V. S. Yinhan Liu, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," in *arXiv preprint arXiv:1907.11692*, 2019.

[33] Z. D. Y. Y. J. C. R. S. Q. V. L. Zhilin Yang, "XLNet: Generalized Autoregressive Pretraining for Language Understanding," in *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, Vancouver, Canada., 2019.

[34] R. T. J. L. Y. Y. J. L. Ji Xin, "DeeBERT: Dynamic Early Exiting for Accelerating BERT Inference," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

[35] S. L. G. Z. Suyang Zhu, "Adversarial Attention Modeling for Multi-dimensional Emotion Regression," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, 2019.

[36] A. O. D. M. Y. C. Keng-Shih Lu, "Perceptually inspired weighted MSE optimization using irregularity-aware graph Fourier transform.," in *arXiv preprint arXiv:2002.08558*, 2020.

[37] L. T. R. P. R. Paula Branco, "SMOGN: a Pre-processing Approach for Imbalanced Regression," in *Proceedings of Machine Learning Research*, 2017.

[38] Z. Q. K. D. D. X. Y. Z. H. Z. H. X. Q. H. Fuzhen Zhuang, "A Comprehensive Survey on Transfer Learning," *arXiv preprint arXiv:1911.02685*, 2019.

[39] M. N. M. I. M. G. C. C. K. L. L. Z. Matthew Peters, "Deep Contextualized Word Representations," *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, vol. Volume 1 (Long Papers), pp. 2227-2237, 2018.

[40] K. N. T. S. ,. I. S. Alec Radford, "Improving Language Understanding With Unsupervised Learning," *Technical report, OpenAI*, 2018.

[41] M. S. Z. C. Q. V. L. M. N. a. o. Yonghui Wu, "Google's Neural Machine

Translation System: Bridging the Gap between Human and Machine Translation," *arXiv preprint arXiv:1609.08144,* 2016.

Danh mục tham khảo tiếng Việt

[42] H. C. D. Nguyễn, D. H. Nguyễn, "Nhận diện cung bậc cảm xúc của bình luận tiếng việt trên mạng xã hội sử dụng phương pháp CNN", Khóa luận tốt nghiệp, Trường Đại học Công nghệ thông tin ĐHQG HCM, 2019.

[43] H. K. Phạm, "Áp dụng phương pháp học sâu vào bài toán phát hiện tin giả với bộ dữ liệu LIAR", Khóa luận tốt nghiệp, Trường Đại học Công nghệ thông tin ĐHQG HCM, 2020.

PHỤ LỤC

A. Danh sách các "stopwords" trong tiếng Anh được sử dụng

Các stopwords được cung cấp bởi thư viện **nltk** và được tuyển chọn lại. Khóa luận này sẽ sử dụng 81 (trong tổng số 179) từ sau đây làm stopwords*: a, about, am, an, at, be, been, can, are, d, did, do, does, had, has, have, he, her, hers, herself, him, himself, his, how, i, if, in, into, is, it, it's, its, itself, ll, m, ma, me, my, myself, o, of, off, on, our, ours, ourselves, re, s, same, she, she's, should, so, some, such, t, than, that, that'll, the, their, theirs, them, themselves, they, ve, very, was, we, were, will, y, you, you'd, you'll, you're, you've, your, yours, yourself, yourselves*.

B. Kết quả huấn luyện chi tiết của các mô hình Vanilla RNN, GRU, LSTM, BiGRU, BiLSTM, CNN-LSTM-GRU và Transformer

Các bảng dưới đây là kết quả chi tiết về hiệu suất của các mô hình khi kết hợp các cấu hình đã trình bài ở mục 4.4. Các từ viết tắt: LR (learning rate) – tốc độ học, BS (batch size) – kích thước một batch trong một lần đào tạo, LF (loss function) – hàm mất mát, $RMSE_{test}$ – giá trị thang đo RMSE trên tập dữ liệu kiểm thử. Giá trị trung bình và độ lệch chuẩn (ĐLC) được tính bằng các hàm *AVERAGE()* và *STDEV.P()* trong Excel. Các kết quả được làm tròn đến 05 chữ số thập phân.

LR	BS	LF	RMSE 1	RMSE 2	RMSE 3	RMSE 4	RMSE 5	Trung bình	ĐLC
0.001	16	mse	0.57477	0.57472	0.57521	0.57473	0.57471	0.57483	0.00019
0.0005	16	mse	0.57472	0.57472	0.57472	0.57472	0.57472	0.57472	0
0.0002	16	mse	**0.57082**	0.57085	0.57091	0.57089	0.57083	**0.57086**	0.00003
0.0001	16	mse	0.60901	0.60901	0.60901	0.60901	0.60901	0.60901	0
0.001	32	mse	0.57480	0.57471	0.57471	0.57471	0.57471	0.57473	0.00004
0.0005	32	mse	0.57475	0.57504	0.57471	0.57471	0.57471	0.57478	0.00013
0.0002	32	mse	0.58665	0.58589	0.60278	0.58820	0.59619	0.59194	0.00654
0.0001	32	mse	0.61998	0.60688	0.61454	0.61785	0.61448	0.61475	0.00445
0.001	64	mse	0.57472	0.57471	0.57471	0.57472	0.57470	0.57471	0.00001
0.0005	64	mse	0.57153	0.57233	0.57468	0.57130	0.57108	0.57218	0.00132
0.0002	64	mse	0.62520	0.61353	0.60702	0.60854	0.60412	0.61168	0.00742
0.0001	64	mse	0.63137	0.63024	0.63453	0.63566	0.63275	0.63291	0.00199
0.001	16	mae	0.59123	0.59123	0.59123	0.59123	0.59123	0.59123	0
0.0005	16	mae	0.59079	0.59079	0.59079	0.59079	0.59079	0.59079	0
0.0002	16	mae	0.57549	0.57549	0.57549	0.57549	0.57549	0.57549	0
0.0001	16	mae	0.62819	0.62819	0.62819	0.62819	0.62819	0.62819	0
0.001	32	mae	0.59128	0.58305	0.59143	0.59050	0.58846	0.58894	0.00313
0.0005	32	mae	0.59070	0.59100	0.59124	0.59111	0.59109	0.59103	0.00018
0.0002	32	mae	0.61099	0.60506	0.61817	0.61035	0.62493	0.61390	0.00692
0.0001	32	mae	0.65127	0.63133	0.65384	0.62540	0.63802	0.63997	0.01105
0.001	64	mae	0.58904	0.58801	0.59136	0.59016	0.59059	0.58983	0.00118
0.0005	64	mae	0.58305	0.57483	0.57111	0.57452	0.58406	0.57751	0.00511
0.0002	64	mae	0.62546	0.63876	0.62529	0.63404	0.63764	0.63224	0.00582
0.0001	64	mae	0.67559	0.65744	0.67282	0.65640	0.64185	0.66082	0.01227

Bảng B.1: Kết quả chi tiết huấn luyện mô hình Vanilla RNN.

LR	BS	LF	RMSE 1	RMSE 2	RMSE 3	RMSE 4	RMSE 5	Trung bình	ĐLC
0.001	16	mse	0.57160	0.56188	0.56038	0.56038	0.55569	0.56199	0.00524
0.0005	16	mse	0.55548	0.55146	0.55361	0.55376	0.55454	0.55377	0.00133
0.0002	16	mse	0.55357	0.54949	0.54949	0.54949	0.54949	0.55031	0.00163
0.0001	16	mse	0.55451	0.55465	0.55465	0.55465	0.55465	0.55462	0.00006
0.001	32	mse	0.55223	0.54767	0.54869	**0.54606**	0.55056	**0.54904**	0.00216
0.0005	32	mse	0.55018	0.54891	0.55018	0.55018	0.55018	0.54993	0.00051
0.0002	32	mse	0.55984	0.55984	0.55984	0.55984	0.56057	0.55999	0.00029
0.0001	32	mse	0.55741	0.55741	0.55741	0.55741	0.55741	0.55741	0
0.001	64	mse	0.55903	0.55903	0.55903	0.54969	0.55427	0.55621	0.00375
0.0005	64	mse	0.55744	0.55744	0.55744	0.55738	0.55986	0.55791	0.00097
0.0002	64	mse	0.57229	0.57145	0.57229	0.57138	0.57138	0.57176	0.00044
0.0001	64	mse	0.56726	0.56726	0.56726	0.56726	0.56678	0.56716	0.00019
0.001	16	mae	0.55102	0.57514	0.55111	0.58167	0.55387	0.56256	0.01314
0.0005	16	mae	0.54649	0.55611	0.55842	0.55360	0.54776	0.55248	0.00464
0.0002	16	mae	0.56699	0.55951	0.56529	0.56529	0.55914	0.56324	0.00326
0.0001	16	mae	0.55787	0.56162	0.55787	0.56234	0.55856	0.55965	0.00193
0.001	32	mae	0.56449	0.55460	0.57147	0.55034	0.55996	0.56017	0.00740
0.0005	32	mae	0.55146	0.55700	0.55682	0.56045	0.54922	0.55499	0.00407
0.0002	32	mae	0.56887	0.56591	0.56902	0.56451	0.56450	0.56656	0.00201
0.0001	32	mae	0.57953	0.57953	0.57953	0.57953	0.57953	0.57953	0
0.001	64	mae	0.56950	0.55891	0.56395	0.54915	0.56103	0.56051	0.00670
0.0005	64	mae	0.54803	0.55402	0.56203	0.55578	0.54803	0.55358	0.00525
0.0002	64	mae	0.57084	0.57084	0.57711	0.57084	0.57084	0.57209	0.00251
0.0001	64	mae	0.56963	0.56963	0.56963	0.56963	0.57292	0.57029	0.00132

Bảng B.2: Kết quả chi tiết huấn luyện mô hình GRU.

LR	BS	LF	RMSE 1	RMSE 2	RMSE 3	RMSE 4	RMSE 5	Trung bình	ĐLC
0.001	16	mse	0.54511	0.55082	**0.54329**	0.54506	0.55082	**0.54702**	0.00317
0.0005	16	mse	0.55069	0.55069	0.55069	0.55069	0.55069	0.55069	0
0.0002	16	mse	0.54712	0.54712	0.54944	0.54712	0.54673	0.54751	0.00098
0.0001	16	mse	0.55862	0.55731	0.55713	0.55729	0.55682	0.55743	0.00062
0.001	32	mse	0.55455	0.55455	0.55455	0.55455	0.55455	0.55455	0
0.0005	32	mse	0.54866	0.54866	0.54866	0.54866	0.54866	0.54866	0
0.0002	32	mse	0.56257	0.56257	0.56257	0.56257	0.56270	0.56260	0.00005
0.0001	32	mse	0.57146	0.57146	0.57146	0.57240	0.57248	0.57185	0.00048
0.001	64	mse	0.56029	0.56029	0.56029	0.56029	0.56029	0.56029	0
0.0005	64	mse	0.55979	0.55979	0.55912	0.55912	0.55912	0.55939	0.00033
0.0002	64	mse	0.58251	0.58251	0.58246	0.58251	0.58251	0.58250	0.00002
0.0001	64	mse	0.58196	0.58196	0.58196	0.58196	0.58196	0.58196	0
0.001	16	mae	0.55078	0.56328	0.56163	0.55986	0.55244	0.55760	0.00503
0.0005	16	mae	0.54736	0.54736	0.54736	0.54759	0.55779	0.54949	0.00415
0.0002	16	mae	0.55483	0.55483	0.55483	0.55349	0.55483	0.55456	0.00054
0.0001	16	mae	0.56749	0.56965	0.56035	0.56965	0.56165	0.56576	0.00399
0.001	32	mae	0.55623	0.55729	0.55587	0.56358	0.55757	0.55811	0.00281
0.0005	32	mae	0.56121	0.56019	0.56186	0.56121	0.55329	0.55955	0.00318
0.0002	32	mae	0.59017	0.58422	0.57824	0.58751	0.57824	0.58368	0.00482
0.0001	32	mae	0.58542	0.58373	0.57817	0.58542	0.58373	0.58329	0.00267
0.001	64	mae	0.55131	0.56967	0.57244	0.55699	0.55606	0.56129	0.00825
0.0005	64	mae	0.57996	0.57103	0.57996	0.57793	0.57848	0.57747	0.00332
0.0002	64	mae	0.58231	0.57618	0.57618	0.57356	0.57618	0.57688	0.0029
0.0001	64	mae	0.59128	0.59128	0.59128	0.59128	0.59128	0.59128	0

Bảng B.3: Kết quả chi tiết huấn luyện mô hình LSTM.

LR	BS	LF	RMSE 1	RMSE 2	RMSE 3	RMSE 4	RMSE 5	Trung bình	ĐLC
0.001	16	mse	0.55130	0.55725	0.55299	0.55242	0.55752	0.55430	0.00258
0.0005	16	mse	0.55718	0.55771	0.55478	0.55763	0.55741	0.55694	0.00110
0.0002	16	mse	0.58366	0.57367	0.57706	0.58366	0.57626	0.57886	0.00407
0.0001	16	mse	0.59771	0.59771	0.59771	0.59873	0.59830	0.59803	0.00042
0.001	32	mse	**0.55043**	0.55397	0.55695	0.55174	0.55121	**0.55286**	0.00236
0.0005	32	mse	0.56545	0.56111	0.57274	0.56754	0.57274	0.56792	0.00445
0.0002	32	mse	0.60140	0.60140	0.60208	0.60208	0.60140	0.60167	0.00033
0.0001	32	mse	0.63979	0.63979	0.63979	0.63979	0.63882	0.63960	0.00039
0.001	64	mse	0.59221	0.59523	0.59523	0.59461	0.59221	0.59390	0.00140
0.0005	64	mse	0.61368	0.63132	0.62401	0.62401	0.62401	0.62341	0.00563
0.0002	64	mse	0.63545	0.63545	0.63767	0.63513	0.63514	0.63577	0.00096
0.0001	64	mse	0.64912	0.64912	0.64912	0.64912	0.64912	0.64912	0
0.001	16	mae	0.56469	0.57699	0.56309	0.56924	0.56926	0.56865	0.00483
0.0005	16	mae	0.56920	0.56920	0.56920	0.56920	0.56920	0.56920	0
0.0002	16	mae	0.59358	0.59358	0.59358	0.59358	0.59358	0.59358	0
0.0001	16	mae	0.57047	0.57047	0.57047	0.57047	0.57047	0.57047	0
0.001	32	mae	0.59226	0.59159	0.58121	0.57107	0.58778	0.58478	0.00790
0.0005	32	mae	0.58802	0.58380	0.59520	0.59360	0.58521	0.58917	0.00451
0.0002	32	mae	0.61477	0.60860	0.59555	0.59962	0.60162	0.60403	0.00683
0.0001	32	mae	0.62978	0.62421	0.63036	0.62474	0.62905	0.62763	0.00261
0.001	64	mae	0.58267	0.58793	0.58249	0.60495	0.58181	0.58797	0.00877
0.0005	64	mae	0.60619	0.62893	0.61509	0.61763	0.62230	0.61803	0.00756
0.0002	64	mae	0.64434	0.64493	0.64426	0.63039	0.64972	0.64273	0.00649
0.0001	64	mae	0.65441	0.65441	0.65441	0.65441	0.65441	0.65441	0

Bảng B.4: Kết quả chi tiết huấn luyện mô hình BiGRU.

LR	BS	LF	RMSE 1	RMSE 2	RMSE	RMSE 4	RMSE 5	Trung bình	ĐLC
0.001	16	mse	0.54151	0.54157	0.54159	**0.54119**	0.54153	**0.54148**	0.00015
0.0005	16	mse	0.54327	0.54324	0.54319	0.54327	0.54332	0.54326	0.00004
0.0002	16	mse	0.55023	0.55033	0.55023	0.55029	0.55029	0.55027	0.00004
0.0001	16	mse	0.55306	0.55308	0.55306	0.55305	0.55305	0.55306	0.00001
0.001	32	mse	0.54774	0.54783	0.54785	0.54764	0.54764	0.54774	0.00009
0.0005	32	mse	0.55046	0.55054	0.55051	0.55046	0.55050	0.55049	0.00003
0.0002	32	mse	0.55238	0.55237	0.55237	0.55238	0.55241	0.55238	0.00001
0.0001	32	mse	0.55028	0.55026	0.55028	0.55028	0.55027	0.55027	0.00001
0.001	64	mse	0.55231	0.55240	0.55238	0.55255	0.55228	0.55238	0.00009
0.0005	64	mse	0.54779	0.54791	0.54788	0.54789	0.54784	0.54786	0.00004
0.0002	64	mse	0.55469	0.55468	0.55468	0.55471	0.55471	0.55469	0.00001
0.0001	64	mse	0.55369	0.55369	0.55370	0.55370	0.55369	0.55369	0
0.001	16	mae	0.55537	0.55523	0.55550	0.55888	0.55404	0.55580	0.00162
0.0005	16	mae	0.56603	0.56714	0.56610	0.56476	0.56809	0.56642	0.00112
0.0002	16	mae	0.56260	0.56206	0.56174	0.56251	0.56155	0.56209	0.00041
0.0001	16	mae	0.55240	0.55247	0.55239	0.55253	0.55247	0.55245	0.00005
0.001	32	mae	0.57171	0.56630	0.57119	0.57096	0.57429	0.57089	0.00258
0.0005	32	mae	0.56996	0.57042	0.56830	0.57105	0.56882	0.56971	0.00101
0.0002	32	mae	0.55642	0.55617	0.55700	0.55689	0.55724	0.55674	0.00039
0.0001	32	mae	0.56095	0.56113	0.56119	0.56104	0.56102	0.56107	0.00008
0.001	64	mae	0.56238	0.56215	0.56282	0.56261	0.56250	0.56249	0.00022
0.0005	64	mae	0.56416	0.56465	0.56594	0.56304	0.56453	0.56446	0.00093
0.0002	64	mae	0.56743	0.56704	0.56738	0.56721	0.55485	0.56478	0.00497
0.0001	64	mae	0.56658	0.56655	0.56651	0.56669	0.56659	0.56658	0.00006

Bảng B.5: Kết quả chi tiết huấn luyện mô hình BiLSTM.

LR	BS	LF	RMSE 1	RMSE 2	RMSE 3	RMSE 4	RMSE 5	Trung bình	ĐLC
0.001	16	mse	0.54244	**0.53966**	0.54064	0.54077	0.54043	**0.54079**	0.00091
0.0005	16	mse	0.55074	0.55041	0.55020	0.55092	0.54886	0.55023	0.00073
0.0002	16	mse	0.55429	0.55346	0.55275	0.55284	0.55334	0.55334	0.00055
0.0001	16	mse	0.56238	0.56066	0.56148	0.56111	0.56068	0.56126	0.00064
0.001	32	mse	0.54380	0.54819	0.54355	0.54399	0.54552	0.54501	0.00173
0.0005	32	mse	0.54768	0.54735	0.54739	0.54802	0.54659	0.54741	0.00047
0.0002	32	mse	0.56468	0.56793	0.56797	0.56806	0.56337	0.56640	0.00198
0.0001	32	mse	0.56621	0.56565	0.56661	0.56746	0.56582	0.56635	0.00065
0.001	64	mse	0.55133	0.54957	0.54897	0.54860	0.54921	0.54954	0.00095
0.0005	64	mse	0.56502	0.56401	0.56348	0.56477	0.56055	0.56357	0.00160
0.0002	64	mse	0.55962	0.55905	0.55870	0.55885	0.55952	0.55915	0.00036
0.0001	64	mse	0.56427	0.56415	0.56386	0.56374	0.56355	0.56391	0.00026
0.001	16	mae	0.54905	0.54828	0.55842	0.55742	0.55071	0.55278	0.00428
0.0005	16	mae	0.55343	0.55824	0.56313	0.55606	0.54854	0.55588	0.00486
0.0002	16	mae	0.57138	0.56590	0.57088	0.57071	0.56796	0.56937	0.00210
0.0001	16	mae	0.57824	0.56804	0.56513	0.56706	0.57646	0.57099	0.00531
0.001	32	mae	0.55171	0.56275	0.54876	0.57191	0.55350	0.55773	0.00850
0.0005	32	mae	0.55946	0.56037	0.57544	0.55571	0.55231	0.56066	0.00793
0.0002	32	mae	0.57633	0.56635	0.57244	0.57690	0.56784	0.57197	0.00429
0.0001	32	mae	0.57968	0.57727	0.58143	0.57868	0.57840	0.57909	0.00140
0.001	64	mae	0.57243	0.58172	0.56141	0.57521	0.59602	0.57736	0.01141
0.0005	64	mae	0.57498	0.57365	0.57271	0.57551	0.56883	0.57314	0.00237
0.0002	64	mae	0.57301	0.57628	0.57094	0.57867	0.57421	0.57462	0.00266
0.0001	64	mae	0.58001	0.58414	0.57642	0.58119	0.57595	0.57954	0.00306

Bảng B.6: Kết quả chi tiết huấn luyện mô hình CNN-LSTM-GRU.

Tham số	RMSE 1	RMSE 2	RMSE 3	RMSE 4	RMSE 5	Trung bình	ĐLC
ah=2, ffdim=16	0.54364	0.53986	0.54064	0.53979	0.54179	0.54114	0.00144
ah=2, ffdim=32	0.53936	**0.53915**	0.54052	0.53955	0.53937	**0.53959**	0.00048
ah=2, ffdim=64	0.54318	0.54233	0.54311	0.54276	0.54319	0.54291	0.00033
ah=3, ffdim=16	0.53957	0.53905	0.54033	0.54146	0.54087	0.54026	0.00087
ah=3, ffdim=32	0.53804	0.54012	0.54070	0.54573	0.53910	0.54074	0.00266
ah=3, ffdim=64	0.54317	0.54126	0.54241	0.54491	0.54534	0.54342	0.00153

Bảng B.7: Kết quả chi tiết mô hình Transformer với *ah* là số lượng lớp *attention*, *ff_dim* là số chiều vector ẩn.

C. Kết quả chi tiết cải tiến mô hình CNN-LSTM-GRU

Dropout	RMSE 1	RMSE 2	RMSE 3	RMSE 4	RMSE 5	Trung bình	ĐLC
0.4	0.54020	0.54181	0.54288	0.53998	0.54098	0.54117	0.00107
0.3	0.53950	0.53954	0.54648	0.54171	0.53999	0.54144	0.00264
0.2	0.53988	**0.53849**	0.53893	0.53998	0.53961	**0.53938**	0.00099
0.1	0.54019	0.53993	0.54124	0.53967	0.54017	0.54024	0.00053

Bảng C.1: Kết quả chi tiết mô hình CNN-LSTM-GRU với các tham số Dropout mới.

Số Filter	Dropout	RMSE 1	RMSE 2	RMSE 3	RMSE 4	RMSE 5	Trung bình	ĐLC
16	0.5	0.54325	0.54377	0.54260	0.54243	0.54288	0.54299	0.00048
16	0.4	0.54160	0.54585	0.54279	0.5418	0.54418	0.54324	0.00159
16	0.3	0.54025	0.54076	0.54669	0.55054	0.54001	0.54365	0.00424
16	0.2	0.53996	0.54093	0.54023	0.54113	0.54008	0.54047	0.00047
16	0.1	0.54106	0.55672	0.54220	0.54054	0.54038	0.54418	0.00630
64	0.5	0.54539	0.54780	0.54455	0.54818	0.54495	0.54617	0.00151
64	0.4	0.54335	0.54523	0.5439	0.54389	0.54287	0.54385	0.00079
64	0.3	0.53947	0.54097	0.54008	0.54078	0.54104	0.54047	0.00060
64	0.2	0.53804	0.53817	0.53871	0.54324	0.54376	0.54038	0.00256
64	0.1	0.53852	0.53916	0.53965	0.54104	0.55866	0.54341	0.00767
128	0.5	0.54415	0.54154	0.54062	0.53878	0.54034	0.54109	0.00177
128	0.4	0.54270	0.54535	0.54149	0.54183	0.54371	0.54302	0.00140
128	0.3	0.54279	0.55643	0.54934	0.54195	0.54444	0.54699	0.00537
128	0.2	0.53996	0.54356	0.53727	0.53908	0.53991	0.53996	0.00207
128	0.1	0.54246	0.54024	0.54030	0.54139	0.54108	0.54109	0.00081

Bảng C.2: Kết quả chi tiết mô hình CNN-LSTM-GRU khi tăng số lượng filter.

Dropout	RMSE 1	RMSE 2	RMSE 3	RMSE 4	RMSE 5	Trung bình	ĐLC
0.5	0.54558	0.54626	0.54087	0.53906	0.54276	0.54291	0.00273
0.4	0.53813	0.54016	0.53882	0.54355	0.53923	0.53998	0.00190
0.3	0.54007	0.54307	0.54332	0.54443	0.53930	0.54204	0.00199
0.2	0.53863	0.53955	0.53971	0.53878	0.53858	0.53905	0.00048
0.1	0.53928	0.53969	0.53892	0.54058	0.53926	0.53955	0.00057

Bảng C.3: Kết quả chi tiết mô hình CNN-LSTM-LSTM.

Dropout	RMSE 1	RMSE 2	RMSE 3	RMSE 4	RMSE 5	Trung bình	ĐLC
0.5	0.54897	0.54600	0.54625	0.54394	0.54832	0.54670	0.00179
0.4	0.54244	0.54534	0.54622	0.54201	0.54963	0.54513	0.00277
0.3	0.54554	0.54475	0.54770	0.54564	0.54835	0.54640	0.00138
0.2	0.54035	0.54114	0.53921	0.54345	0.53974	0.54078	0.00148
0.1	0.54041	0.55554	0.54246	0.54446	0.54125	0.54482	0.00553

Bảng C.4: Kết quả chi tiết mô hình CNN-BiLSTM-BiGRU.

Dropout	RMSE 1	RMSE 2	RMSE 3	RMSE 4	RMSE 5	Trung bình	ĐLC
0.5	0.54431	0.54416	0.5517	0.54365	0.54325	0.54541	0.00317
0.4	0.54120	0.54108	0.54572	0.54283	0.54485	0.54314	0.00188
0.3	0.53714	0.53989	0.54015	0.54168	0.53921	0.53961	0.00148
0.2	0.53746	0.53868	0.53860	0.53630	0.53698	0.53760	0.00092
0.1	0.54850	0.53650	0.53782	0.53763	0.54216	0.54052	0.00443

Bảng C.5: Kết quả chi tiết mô hình CNN-BiLSTM-BiLSTM.

D. Kết quả chi tiết các cải tiến từ mô hình CNN-BiLSTM-BiLSTM

Thí nghiệm	RMSE 1	RMSE 2	RMSE 3	RMSE 4	RMSE 5	Trung bình	ĐLC
Scale đầu ra	0.55769	0.54207	0.54855	0.54780	0.54575	0.54837	0.00517
Scale đầu vào và đầu ra	0.54154	0.54438	0.54422	0.54589	0.55155	0.54552	0.00332

Bảng D.1: Kết quả chi tiết khi scale dữ liệu.

Giá trị alpha	RMSE 1	RMSE 2	RMSE 3	RMSE 4	RMSE 5	Trung bình	ĐLC
0.1	0.54155	0.54210	0.53945	0.54035	0.54070	0.54083	0.00093
0.2	0.54200	0.54387	0.55052	0.54122	0.54507	0.54454	0.00329
0.3	0.54912	0.54844	0.54686	0.54249	0.54865	0.54711	0.00243
0.4	0.54886	0.55065	0.54966	0.54583	0.54839	0.54868	0.00162

Bảng D.2: Kết quả chi tiết khi thay thế hàm kích hoạt ReLU bằng LeakyReLU.

Giá trị alpha	RMSE 1	RMSE 2	RMSE 3	RMSE 4	RMSE 5	Trung bình	ĐLC
0.1	0.53923	0.54048	0.54001	0.54463	0.53970	0.54081	0.00195
0.2	0.53994	0.54143	0.54140	0.54928	0.53952	0.54231	0.00357
0.3	0.54857	0.53952	0.53885	0.53990	0.53892	0.54115	0.00373
0.4	0.54245	0.55348	0.55663	0.54224	0.53993	0.54695	0.00675

Bảng D.3: Kết quả chi tiết khi thay thế hàm kích hoạt ReLU bằng ELU.

Hàm tối ưu	RMSE 1	RMSE 2	RMSE 3	RMSE 4	RMSE 5	Trung bình	ĐLC
amsgrad	0.53757	0.53928	0.53868	0.53744	0.53710	0.53801	0.00083
RMSprop	0.55292	0.55833	0.55987	0.55406	0.55968	0.55697	0.00291
Nadam	0.57176	0.56237	0.56896	0.56059	0.55869	0.56447	0.00502
Adamax	0.54690	0.54685	0.54700	0.55349	0.54708	0.54826	0.00261

Bảng D.4: Kết quả chi tiết khi thay thế hàm tối ưu Adam bằng các hàm amsgrad, RMSprop, Nadam, Adamax.

Thí nghiệm	RMSE 1	RMSE 2	RMSE 3	RMSE 4	RMSE 5	Trung bình	ĐLC
Trước hàm kích hoạt, Batch size 16	0.54063	0.54078	0.54095	0.53993	0.53957	0.54037	0.00053
Sau hàm kích hoạt, Batch size 16	0.54083	0.54400	0.54249	0.54038	0.54493	0.54253	0.00176
Trước hàm kích hoạt, Batch size 32	0.54280	0.54193	0.54206	0.54211	0.54248	0.54228	0.00032
Sau hàm kích hoạt, Batch size 32	0.54264	0.54323	0.54399	0.5443	0.54473	0.54378	0.00075

Bảng D.5: Kết quả chi tiết khi sử dụng BatchNormalization.

fasttext	RMSE 1	RMSE 2	RMSE 3	RMSE 4	RMSE 5	Trung bình	ĐLC
Không subword	0.54749	0.54225	0.54625	0.54529	0.54207	0.54467	0.00217
subword	0.55899	0.55289	0.56025	0.55644	0.55025	0.55576	0.00373

Bảng D.6: Kết quả chi tiết khi sử dụng Fasttext thay cho GloVe.