VIETNAM NATIONAL UNIVERSITY
UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Magneto @ DS&KTLab

ASPECT-ORIENTED SENTIMENT ANALYSIS FOR VIETNAMESE E-COMMERCE REVIEWS

STUDENT SCIENTIFIC RESEARCH REPORT

Faculty of Information Technology

HANOI - 2021

VIETNAM NATIONAL UNIVERSITY
UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Le Thi Phuong
Le Minh Binh
Bui Khanh Huyen
Tran Khanh Hung

ASPECT-ORIENTED SENTIMENT ANALYSIS FOR VIETNAMESE E-COMMERCE REVIEWS

STUDENT SCIENTIFIC RESEARCH REPORT

Faculty of Information Technology

Supervisors:

MSc. Le Hoang Quynh

MSc. Can Duy Cat

HANOI - 2021

Abstract

Chúng tôi giới thiệu Graph Attention Network (GAT), mô hình mạng nơ-ron mới hoạt động trên dữ liệu dạng đồ thị, sử dụng các lớp tự quan tâm đã đặt mặt nạ để cải thiện những nhược điểm của các phương pháp trước đó dựa trên hỗn hợp đồ thị hoặc các ước lượng của chúng. Bằng cách xếp lớp trong đó các nút có thể quan tâm đến các đặc trưng của lân cận, chúng ta cho phép (ẩn dụ) chỉ định các trọng số khác nhau cho các nút trong một lân cận, mà không cần bất kỳ hoạt động ma trận chi phí (như đảo ngược) hoặc phụ thuộc vào việc biết cấu trúc đồ thị trước. Theo cách này, chúng tôi giải quyết một số vấn đề chính của các mạng nơ-ron đồ thị dựa trên tần số cùng lúc, và làm cho mô hình của chúng tôi dễ dàng áp dụng cho cả vấn đề dẫn chứng và dẫn điểm. Mô hình GAT của chúng tôi đã cho kết quả ??? trên bốn mô hình kiểm thử transductive và inductive: các tập dữ liệu Cora, Citeseer và Pubmed, cùng với tập dữ liệu tương tác protein-protein (ở đây, test graph bị ẩn trong quá trình học máy).

Keywords*: Graph Attention Networks(GAT), transductive, Vietnamese multi-aspect dataset.*

Acknowledgements

Lời đầu tiên, chúng em xin cảm ơn Cô Nguyễn Thị Cẩm Vân và các thầy cô trong lab Data Science and Knowledge Technology Laboratory at University of Engineering and Technology. Các thầy cô đã luôn giúp đỡ và hướng dẫn chúng em viết ra bài báo này.

Đồng thời em cũng xin cảm ơn khoa Công nghệ thông tin - Trường Đại học Công nghệ đã tạo điều kiện để cho chúng em có được trải nghiệm thực tế trong môi trường nghiên cứu khoa học và tiếp thu nhiều kiến thức bổ ích.

Declaration

Chúng tôi tuyên bố rằng tác phẩm của chúng tôi là do chính chúng tôi sáng tác và tác phẩm đó không được nộp cho bất kỳ bằng cấp hoặc trình độ chuyên môn nào khác. Chúng tôi xác nhận rằng tác phẩm được gửi là của riêng bạn, trừ khi tác phẩm đã tạo thành một phần của các ấn phẩm đồng tác giả đã được đưa vào. Đóng góp của chúng tôi và của các tác giả khác cho tác phẩm này đã được chỉ ra rõ ràng dưới đây. Chúng tôi xác nhận rằng tín dụng thích hợp đã được đưa ra trong bài viết này khi tài liệu tham khảo đã được thực hiện cho công việc của người khác.

Chúng tôi xác nhận rằng, theo hiểu biết tốt nhất của chúng tôi, bài viết của chúng tôi không vi phạm bản quyền của bất kỳ ai cũng như không vi phạm bất kỳ quyền sở hữu nào và rằng mọi ý tưởng, kỹ thuật, trích dẫn hoặc bất kỳ tài liệu nào khác từ tác phẩm của người khác được đưa vào luận án của tôi, đã xuất bản hoặc mặt khác, được thừa nhận đầy đủ theo các thông lệ tham chiếu tiêu chuẩn. Hơn nữa, trong phạm vi mà chúng tôi đã bao gồm tài liệu có bản quyền, chúng tôi xác nhận rằng chúng tôi đã nhận được sự cho phép bằng văn bản từ (những) chủ sở hữu bản quyền để đưa (những) tài liệu đó vào tác phẩm của chúng tôi và có đầy đủ quyền tác giả để cải thiện những tài liệu này.

Nhóm sinh viên

Đoàn Văn Nguyên
Nguyễn Đức Trọng
Nguyễn Trần Đạt

Table of Contents

Chapter 1

Giới thiệu

Mạng Nơ-ron Tích Chập (Convolutional Neural Networks - CNN) hiện đang được áp dụng để giải các bài toán phân loại hình ảnh (He et al., 2016), phân vùng ngữ nghĩa ảnh (Jegou et al., 2017) hoặc dịch máy (Gehring et al., 2016), trong đó dữ liệu nền được biểu diễn dưới cấu trúc giống mạng lưới. Các mô hình này tái sử dụng hiệu quả các bộ lọc cục bộ, với các tham số học, bằng cách áp dụng chúng cho tất cả các vị trí đầu vào.

Tuy nhiên, nhiều tác vụ có liên quan đến dữ liệu không thể biểu diễn dưới dạng mạng lưới dưới dạng bất quy tắc. Bản dựng hình 3D, mạng xã hội, mạng viễn thông, mạng sinh học hoặc mạng kết nối não bộ là các ví dụ điển hình. Những loại dữ liệu này thường được biểu diễn dưới dạng đồ thị.

Trong lĩnh vực công nghệ thông tin, đã có một số nghiên cứu về việc mở rộng mạng nơ-ron để xử lý các đồ thị . Các nghiên cứu ban đầu sử dụng mạng nơ–ron đệ quy để xử lý dữ liệu được biểu diễn dưới dạng đồ thị đường vòng được định hướng (Frasconi et al., 1998; Sperduti & Starita, 1997). Mạng Nơ-ron Đồ Thị (Graph Neural Network - GNN) được giới thiệu trong Gori et al. (2005) và Scarselli et al. (2009) là một mô hình chuẩn hóa của mạng nơ-ron đệ quy có thể trực tiếp xử lý các loại đồ thị bao quát hơn, ví dụ như đồ thị có chu trình, có hướng và không hướng. GNN bao gồm một quá trình lặp đi lặp lại, truyền trạng thái của nút cho đến khi đạt đối tượng; sau đó là một mạng nơ-ron, tạo ra một kết quả cho mỗi nút dựa trên trạng thái của nó. Ý tưởng này đã được Li et al. (2016) thừa hưởng và cải tiến bằng cách sử dụng đơn vị gated recurrent (Cho et al., 2014) trong bước truyền.

Tuy nhiên, việc mở rộng bộ lọc tổng quát cho không gian đồ thị đang càng ngày càng được quan tâm. Những tiến bộ theo hướng này thường được phân loại thành phương

pháp quang phổ và phương pháp phi quang phổ.

Một mặt, các phương pháp quang phổ hoạt động dựa trên một biểu diễn quang phổ của đồ thị và đã được áp dụng thành công trong bài toán phân loại nút. Trong Bruna et al. (2014), phép tích chập được xác định trong tầng Fourier bằng cách tính phân tích độ riêng của ma trận Laplacian của đồ thị, dẫn đến việc tính toán có thể nặng và bộ lọc không quang phổ tọa độ. Những vấn đề này đã được giải quyết bởi các công trình sau. Henaff et al. (2015) đã giới thiệu một tham số hóa của bộ lọc quang phổ với các hệ số mượt để làm cho chúng có tọa độ quang phổ. Sau này, Defferrard et al. (2016) đề xuất sử dụng phép mở rộng Chebyshev của ma trận Laplacian để ước lượng bộ lọc, loại bỏ việc tính toán các vector riêng của Laplacian và cho ra các bộ lọc có tính địa lý, Kipf & Welling (2017) đã đơn giản hoá phương pháp trước bằng cách hạn chế bộ lọc hoạt động trong một vùng lân cận xung quanh mỗi nút. Tuy nhiên, trong tất cả các phương pháp đồng bộ hoá tần số trên, các bộ lọc được học phụ thuộc vào cơ sở vector riêng của Laplacian, điều này phụ thuộc vào cấu trúc đồ thị. Do đó, một mô hình được huấn luyện trên một cấu trúc cụ thể không thể được áp dụng trực tiếp cho một đồ thị với cấu trúc khác.

Mặt khác, các phương pháp phi quang phổ (Duvenaud et al., 2015; Atwood & Towsley,2016; Hamilton et al., 2017) xác định việc tích chập trực tiếp trên đồ thị, hoạt động trên nhóm các đỉnh gần về vị trí. Một trong những thách thức của các phương pháp này là xác định một toán tử hoạt động với các lân cận có kích thước khác nhau và giữ thuộc tính chia sẻ trọng lượng của CNN. Trong một số trường hợp, điều này yêu cầu học một ma trận trọng lượng cụ thể cho mỗi bậc đỉnh (Duvenaud et al., 2015), sử dụng các lũy thừa của ma trận chuyển đổi để xác định lân cận và học trọng lượng cho mỗi kênh đầu vào và bậc lân cận (Atwood & Towsley, 2016), hoặc trích xuất và chuẩn hóa các lân cận chứa số lượng cố định của đỉnh (Niepert et al., 2016). Monti et al. (2016) đã trình bày mô hình tổ hợp CNN (MoNet), một phương pháp vị trí cho phép tổng quát hóa kiến trúc CNN sang đồ thị. Gần đây, Hamilton et al. (2017) đã giới thiệu GraphSAGE, Một phương pháp tính biểu diễn của nút theo cách tuân theo. Kỹ thuật này hoạt động bằng cách mẫu một khu vực cố định kích thước của mỗi nút, sau đó thực hiện một trọng tải chung cho nó (như là trung bình của tất cả các vectơ đặc trưng của hàng xóm được mẫu, hoặc kết quả của việc cho chúng qua mạng nơ-ron tần suất). Phương pháp này đã cho kết quả tuyệt vời trên nhiều tiêu chuẩn thu hồi lớn.

Cơ chế tập trung đã trở thành gần như một tiêu chuẩn trong nhiều tác vụ dựa trên chuỗi (Bahdanau et al., 2015; Gehring et al., 2016). Một trong những điểm mạnh của

các mạng tập trung là xử lý với dữ liệu đầu vào có kích thước biến đổi, tập trung vào các phần quan trọng nhất của đầu vào để tính toán. Khi sử dụng mạng tập trung để tính toán một biểu diễn của một chuỗi, nó thường được gọi là tập trung tự hoặc tập trung trong. Cùng với mạng nơ-ron lặp lại (RNNs) hoặc tích chập, tập trung tự đã chứng minh là hữu ích cho các tác vụ như đọc máy (Cheng et al., 2016) và học biểu diễn câu (Lin et al., 2017). Tuy nhiên, Vaswani et al. (2017) cho thấy rằng tập trung tự không chỉ có thể cải thiện một phương pháp dựa trên RNNs hoặc tích chập, mà còn đủ để xây dựng một mô hình mạnh mẽ nhận được hiệu năng tiên tiến nhất trong tác vụ dịch máy.

Lấy cảm hứng từ công trình nghiên cứu trên, chúng tôi giới thiệu một kiến trúc dựa trên chú ý để thực hiện phân loại nút của dữ liệu cấu trúc đồ thị. Ý tưởng là tính toán biểu diễn ẩn của mỗi nút trong đồ thị, bằng cách chú ý trên các nút lân cận của nó, theo một chiến lược chú ý tự. Kiến trúc chú ý có một số thuộc tính quan trọng: (1) hoạt động hiệu quả, vì nó có thể được paralllel hoá giữa các cặp nút lân cận; (2) nó có thể được áp dụng cho các nút đồ thị có số bậc khác nhau bằng cách chỉ định các trọng lượng tùy ý cho các nút lân cận; và (3) mô hình có thể trực tiếp được áp dụng cho các vấn đề học theo một cách dựa trên thực tế, Bao gồm các nhiệm vụ mà mô hình phải tổng hợp đến các đồ thị hoàn toàn chưa từng thấy. Chúng tôi xác nhận giải pháp được đề xuất trên bốn mốc thử nghiệm khó: mạng trích dẫn Cora, Citeseer và Pubmed cũng như tập dữ liệu tương tác protein-protein theo phương pháp chỉ dẫn, đạt được hoặc phù hợp với kết quả tiên tiến nhất trong lĩnh vực để nổi bật tiềm năng của các mô hình dựa trên chú ý khi đối mặt với các đồ thị tùy ý.

Nên chú ý rằng, giống như Kipf & Welling (2017) và Atwood & Towsley (2016), công trình của chúng tôi cũng có thể được tái cấu trúc như một trường hợp cụ thể của MoNet (Monti et al., 2016). Hơn nữa, cách tiếp cận chúng tôi chia sẻ tính toán mạng nơ-ron qua các cạnh giống như cách tạo dạng của các mạng quan hệ (Santoro et al., 2017) và VAIN (Hoshen, 2017), trong đó các mối quan hệ giữa các đối tượng hoặc đại diện được tổng hợp theo cặp, bằng cách sử dụng một mục đích chung. Tương tự, mô hình chú ý được đề xuất của chúng tôi có thể được kết nối với các công trình của Duan et al. (2017) và Denil et al. (2017), sử dụng một hoạt động chú ý lân cận để tính toán các hệ số chú ý giữa các đối tượng khác nhau trong một môi trường. Các phương pháp liên quan khác bao gồm nhúng tính linearly cục bộ (LLE) (Roweis & Saul, 2000) và các mạng nhớ (Weston et al., 2014). LLE chọn một số cố định của các láng giềng xung quanh mỗi điểm dữ liệu,và học một trọng số trọng tâm cho mỗi điểm lân cận để tái tạo mỗi điểm như là tổng trọng tâm của các điểm lân cận của nó. Bước tối ưu hóa thứ hai trích xuất nhúng đặc trưng của điểm. Mạng nhớ cũng chia sẻ một số kết nối với công

việc của chúng tôi, đặc biệt là nếu chúng tôi hiểu khu vực xung quanh một nút như là bộ nhớ, được sử dụng để tính toán các đặc trưng của nút bằng cách chú ý về giá trị của nó, và sau đó được cập nhật bằng cách lưu trữ các đặc trưng mới tại cùng vị trí.

Chapter 2

MÔ HÌNH GRAPH ATTENTION NETWORK

Trong phần này, chúng tôi sẽ trình bày các kiến thức nền tảng được sử dụng để xây dựng các mạng quan sát đồ thị tùy chọn (bằng cách chồng lớp này) và liệt kê các hiệu quả về mặt lý thuyết và thực tế cùng các hạn chế so với công trình liên quan đến lĩnh vực xử lý đồ thị nơ-ron trước đây.

2.1 Đồ Thị Phân Lớp Tập Trung (Graph Attentional Layer)

Chúng ta bắt đầu bằng viết một *phân lớp đồ thị tập trung* đơn lẻ, đây là lớp duy nhất được sử dụng trong toàn bộ các mô hình GAT trong các thử nghiệm của chúng tôi. Cụ thể, mô hình tập trung được chúng tôi sử dụng với công trình của Bahdanau et al. (2015) - nhưng khung làm việc vẫn có thể được áp dụng với loại cơ chế tập trung này.

Dữ liệu đầu vào của lớp là một tập hợp các đặc điểm nút, **h =** $\{\vec{h_1}, \vec{h_2}, ...\vec{h_N}\}$, $\vec{h_i} \in \mathbb{R}^F$, trong đó N là số lượng nút và F là số lượng đặc điểm trong mỗi nút. Lớp này tạo ra một tập hợp các đặc điểm nút mới (số các phần tử có thể chênh lệch F'), **h' =** $\{\vec{h'_1}, \vec{h'_2}, ...\vec{h'_N}\}$, $\vec{h'_i} \in \mathbb{R}^F$, làm dữ liệu đầu ra.

Để dữ liệu biểu diễn đủ hiệu quả để chuyển đổi dữ liệu đầu vào thành cấp cao hơn, phải cần ít nhất một ánh xạ tuyến tính learnable. Vì vậy, trước hết, một ánh xạ tuyến tính chung, được tham số hóa bởi *ma trận trọng số*, $\mathbf{W} \in \mathbb{R}^{F'XF}$, được gán cho mỗi nút. Chúng ta sau đó thực hiện *self-attention* trên các nút - một đích tập trung: $\mathbb{R}^{F'}X\mathbb{R}^F \to \mathbb{R}$ tính hệ số tập trung

$$e_{ij} = a(\mathbf{W}\vec{h_i}, \mathbf{W}\vec{h_j})$$

cho thấy sự quan trọng của đặc trưng của nút j đối với nút i. Trong biểu diễn tổng quát nhất, mô hình cho phép mỗi nút tập trung tới mọi nút khác, *loại bỏ tất cả thông tin cấu trúc*. Chúng ta tiếp cận cấu trúc đồ thị qua việc thực hiện sự tập trung được che đậy - Chỉ tính eij cho các nút $j \in N_i$, trong đó N_i là một khu vực xung quanh nút i trong đồ thị. Trong tất cả các thí nghiệm của chúng tôi, đó sẽ chính là các nút đồng bậc đầu tiên của i (bao gồm i). Để dễ dàng so sánh các hệ số giữa các nút khác nhau, chúng tôi chuẩn hóa chúng theo tất cả các lựa chọn của j bằng hàm softmax:

$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k \in N_i} exp(e_{ik})}$$

Trong các thí nghiệm của chúng tôi, cơ chế tập trung a là một mạng nơ-ron truyền thẳng đơn lớp, được tham số hóa bởi một vectơ trọng số $\vec{\mathbf{a}} \in \mathbb{R}_{2F'}$ và được áp dụng phi tuyến LeakyReLU (với đầu vào đối mạng âm $\alpha = 0,2$). Mở rộng ra, các hệ số được tính bởi cơ chế tập trung (minh họa bởi Hình 1 (trái)) có thể được biểu diễn như sau:

$$\alpha_{ij} = \frac{exp(LeakyReLU(\vec{a}^T[\mathbf{W}\vec{h_i} \| \mathbf{W}\vec{h_j}]))}{\sum_{k \in N_i} exp(LeakyReLU(\vec{a}^T[\mathbf{W}\vec{h_i} \| \mathbf{W}\vec{h_k}]))}$$

$.^T$ đại diện cho phép chuyển vị, và $\|$ là thao tác nối tiếp.

Sau khi được tính toán, các hệ số quan tâm được chuẩn hóa được sử dụng để tính tổng trọng số của các đặc trưng tương ứng với chúng, để dùng làm đặc trưng đầu ra cuối cùng cho mỗi nút (sau có thể là

Áp dụng một hàm phi tuyến, σ):

$$\vec{h_i'} = \sigma(\sum_{j \in N_i} \alpha_{ij} \mathbf{W}\vec{h_i})$$

Để stabilizr quá trình học tự quan tâm, chúng tôi đã tìm thấy mở rộng cơ chế của chúng tôi để sử dụng quan tâm đa đầu có lợi, tương tự như Vaswani et al. (2017). Cụ thể, K cơ chế quan tâm độc lập thực hiện biến đổi của Phương trình 4, và sau đó các đặc trưng của chúng được nối tiếp, dẫn đến biểu diễn đặc trưng đầu ra sau:

Trong đó, $\|$ đại diện cho việc nối tiếp, α_{ij}^k là các hệ số quan tâm đã chuẩn hóa được tính bởi cơ chế quan tâm k-th (a^k), và $\mathbf{W}^k$ là ma trận trọng số biến đổi tuyến tính đầu vào tương ứng. Lưu ý rằng, trong tình huống này, kết quả trả về cuối cùng, **h'**, sẽ bao gồm KF' đặc trưng (thay vì F') cho mỗi nút.

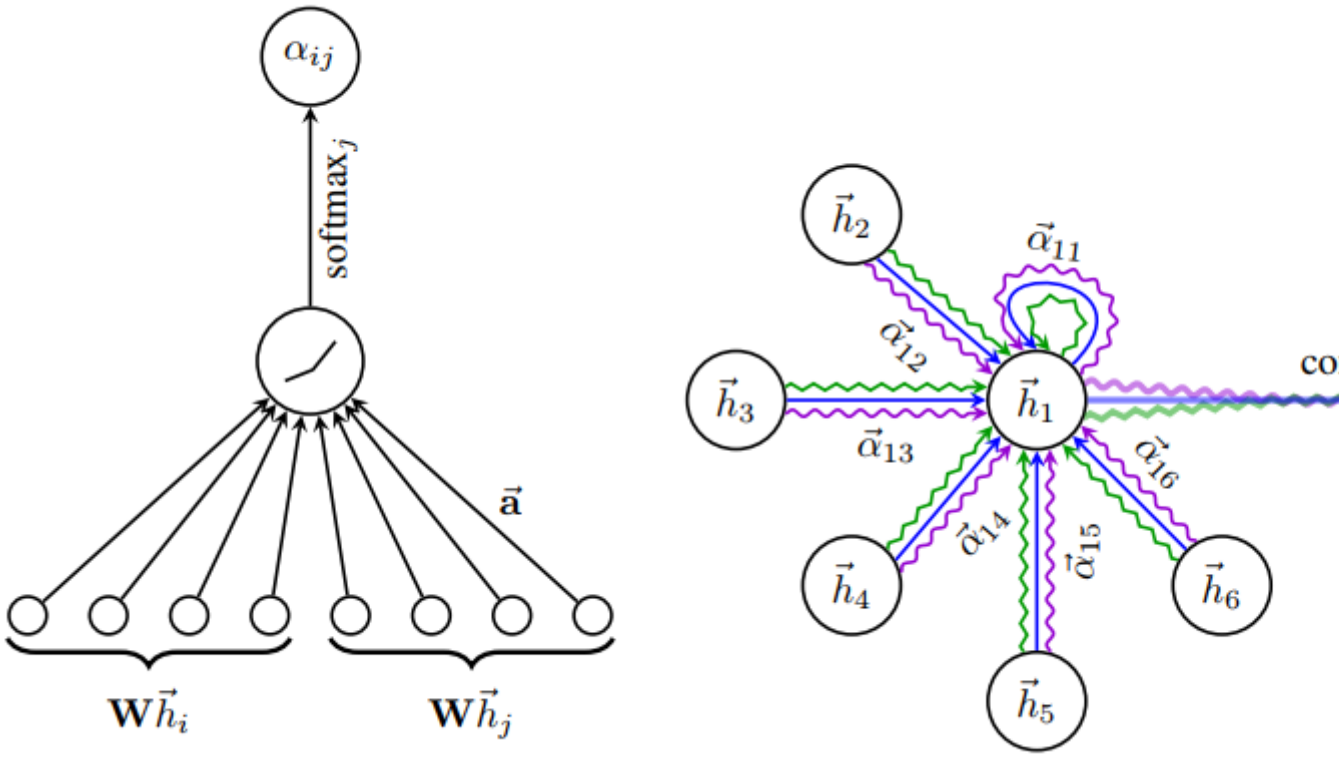

Figure 2.1: **Left**: Cơ chế quan tâm $a(\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j)$ được sử dụng bởi mô hình của chúng tôi, được tham số hóa bởi một vectơ trọng số $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$, áp dụng một hoạt động LeakyReLU. Bên phải: Một minh hoạ của quan tâm đa đầu (với $K = 3$ đầu) bởi nút 1 trên khu vực xung quanh của nó. Các kiểu mũi tên và màu sắc khác nhau đại diện cho các tính toán quan tâm độc lập. Các đặc trưng được tổng hợp từ mỗi đầu được nối tiếp hoặc trung bình để đạt được $\vec{h_1'}$,

$$\vec{h}'_i = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right)$$

Đặc biệt, nếu chúng ta thực hiện quan tâm đa đầu trên lớp cuối cùng (dự đoán) của mạng, việc nối tiếp không còn hợp lý - thay vào đó, chúng tôi sử dụng trung bình, và hoãn áp dụng nonlinearity cuối cùng (thường là một softmax hoặc sigmoid logistic cho các vấn đề phân loại) cho đến khi đó:

$$\vec{h}'_i = \sigma(\frac{1}{K}\sum_{k=1}^{K}\sum_{j \in N_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_i)$$

Quá trình tập hợp của một lớp quan tâm đồ thị đa đầu được minh họa bởi Hình 1 (bên phải).

2.2 So sánh với các nghiên cứu trước

Lớp quan sát đồ thị mô tả trong phần 2.1 trực tiếp giải quyết một số vấn đề xuất hiện trong các phương pháp trước đó về mô hình hóa dữ liệu cấu trúc đồ thị:

- Tính toán hiệu quả: hoạt động của lớp tự quan tâm có thể được đồng bộ hóa trên tất cả các cạnh và tính toán của đặc trưng đầu ra có thể được đồng bộ hóa trên tất cả các nút. Không cần phải tính toán các phép biến đổi ma trận giống như tính toán $eigendecompositions$ hoặc các phép toán ma trận tương tự chi phí cao. Độ phức tạp của một đầu quan tâm GAT tính toán đặc trưng F' có thể biểu diễn dưới dạng $O(|V|FF' + |E|F')$, F là số lượng các đặc trưng đầu vào, $|V|$ và $|E|$ là số lượng nút và cạnh trong đồ thị, tương ứng. Độ phức tạp này tương đương với các phương pháp cơ sở như Mạng Convolutional đồ thị (GCN) (Kipf & Welling, 2017). Áp dụng nhiều đầu tập trung tăng yêu cầu lưu trữ và tham số lên một yếu tố K, trong khi tính toán của các đầu riêng biệt là hoàn toàn độc lập và có thể song song hóa.

- Khác với GCNs, mô hình của chúng tôi cho phép (ẩn định) gán sự quan trọng khác nhau cho các nút trong một khu vực giống nhau, cho phép mô hình có khả năng lớn hơn. Ngoài ra, phân tích trọng số tập trung đã học có thể dẫn đến lợi ích trong khả năng hiểu biết, như trong lĩnh vực dịch máy (ví dụ, phân tích chất lượng của Bahdanau et al. (2015))

- Mechanism attention được áp dụng theo cách chung cho tất cả các cạnh trong đồ

thị và do đó không phụ thuộc vào truy cập trước tiên đến cấu trúc đồ thị toàn cầu hoặc (các đặc trưng của) tất cả nút (một hạn chế của nhiều kỹ thuật trước đó). Điều này có nhiều hậu quả mong muốn:

- Không yêu cầu đồ thị phải là không hướng (chúng ta có thể để trống tính toán ij nếu cạnh $j \longrightarrow i$ không tồn tại).
- Cách tiếp cận của chúng ta cho phép áp dụng trực tiếp cho huấn luyện tại chỗ - bao gồm các nhiệm vụ nơi mô hình được đánh giá trên các đồ thị hoàn toàn chưa từng nhìn thấy trong quá trình huấn luyện.

- Kỹ thuật tập trung được áp dụng theo cách chung đến tất cả các cạnh trong đồ thị và do đó, nó không phụ thuộc vào truy cập trước tới cấu trúc đồ thị toàn cầu hoặc tính năng của tất cả nút (giới hạn của nhiều kỹ thuật trước đó). Điều này có nhiều kết quả mong muốn: cách tiếp cận mới của Hamilton et al. (2017) lấy mẫu một vùng lưới cố định của mỗi nút để giữ cho chân dung tính toán giống nhau, điều này không cho phép truy cập tới toàn bộ vùng lưới trong quá trình suy diễn. Ngoài ra, kỹ thuật này đạt được một số kết quả mạnh mẽ nhất khi sử dụng tập trung vùng lưới dựa trên LSTM (Hochreiter & Schmidhuber, 1997). Điều này giả định tồn tại một thứ tự nút liên tục qua các vùng lưới và tác giả đã sửa chữa nó bằng cách truyền vào các dãy ngẫu nhiên được sắp xếp cho LSTM. Cách thức của chúng tôi không gặp phải bất kỳ vấn đề nào của những vấn đề đó - nó hoạt động với toàn bộ khu vực lân cận (với chi phí tính toán biến đổi, vẫn cùng mức với các phương pháp như GCN), và không giả định bất kỳ sắp xếp nào trong nó.

- GAT có thể được tái cấu trúc thành một trường hợp cụ thể của MoNet (Monti et al., 2016). Cụ thể hơn thiết lập hàm tọa độ giả là $u(x, y) = f(x) \| f(y)$, trong đó $f(x)$ là đặc trưng (có thể được chuyển đổi bằng MLP) của nút x và $\|$ là sự nối tiếp; và hàm trọng lượng là $\omega_j(u) = softmax(MLP(u))$ (với softmax được thực hiện trên toàn bộ lân cận của một nút), sẽ kiến toán tử patch của MoNet tương tự với chúng tôi. Tuy nhiên, cần lưu ý rằng, so với các phiên bản MoNet đã được xem xét trước đây, mô hình của chúng tôi sử dụng các tính năng nút để tính toán độ tương tự, thay vì các thuộc tính cấu trúc của nút (giả sử biết trước cấu trúc biểu đồ).

Chúng tôi đã có thể tạo ra một phiên bản của lớp GAT sử dụng các hoạt động ma trận rỗng, giảm độ phức tạp lưu trữ thành tuyến tính trong số lượng nút và cạnh và cho phép thực thiện các mô hình GAT trên các tập dữ liệu đồ thị lớn hơn. Tuy nhiên, khung quản lý xử lý tensor mà chúng tôi sử dụng chỉ hỗ trợ nhân ma trận rỗng cho tensor rank-2,

điều này hạn chế khả năng nhóm của lớp như hiện tại đã triển khai (đặc biệt là cho các tập dữ liệu có nhiều đồ thị). Định hướng giải quyết hạn chế này là quan trọng đối với công việc trong tương lai. Tùy thuộc vào tính chính xác của cấp trúc đồ thị, GPU có thể không cung cấp lợi ích hiệu suất lớn so với CPU trong những trường hợp rỗng. Nên tập trung rằng kích thước của "Khung nhìn" của mô hình của chúng tôi được giới hạn bởi độ sâu của mạng (tương tự cho GCN và các mô hình tương tự). Kỹ thuật như kết nối bỏ qua (He et al., 2016) có thể được áp dụng dễ dàng để mở rộng độ sâu, tuy nhiên. Cuối cùng việc lập trình song song cho tất cả các cạnh đồ thị, đặc biệt là theo cách phân tán, có thể gặp nhiều tính toán trùng lặp, vì khu vực lân cận thường rất tương đồng trong các đồ thị quan tâm.

Chapter 3

Methods

In this chapter, we firstly present the methodology for our approaches, then describe our aspect-oriented model in detail, begin with an overview of the whole architecture, followed by the explanation of how each of the components work; lastly, we introduce the parameter system used in the model.

3.1 Theoretical Basis

3.1.1 K-nearest Neighbors

KNN is one of the most straightforward supervised algorithms, but effective in dealing some problems, in both classification and regression. KNN does not study anything from training data (therefore it's considered as a lazy learning method)

In classification problem, a new data point's class is derive from k nearest data point from the training set. In general, this label can be calculated by the label of nearest data point $k = 1$ $(weights = uniform)$ or average weight of the nearest points $(weights = distance)$, or a relationship between these weight.

KNN is simple as only two criteria needed: value of k and the function to calculate distance. KNN does not require any training period because it does not derived any function from training data and only make real-time prediction when evaluating, so this make the algorithm run much faster, especially considering small dataset. New data can be also add anytime without any affection to the algorithm's accuracy.

Despite lightning performance in small dataset, as the data size enlarges, KNN is not beneficial for both memory and time. The algorithm must remember all the training

data for label calculation, so it will become extensively slow as size of dataset grows.

3.1.2 Logistics Regression

LR, in its basic form, uses a logistic function (e.g., sigmoid, tanh) to model a binary dependent variable. The prediction score of LR is calculated by formula:

$$f(x) = \theta(\mathbf{w}^T \mathbf{x})$$

where

θ: logistics function (e.g., sigmoid, tanh, etc.)

The cost function for LR is defined as:

$$L = \sum_{D} -ylog(y') - (1-y)log(1-y'))$$

where

D: dataset, containing of labeled tuples (x, y)

y: the label in a assigned example, which is either be 0 or 1.

y' is the predicted value, which is between 0 and 1, given features in x.

LR is one of the easiest ML algorithms as it is easy to implement, interpret, and very efficient to train. Traning a model with LR doesn't need high computation effort. LR also less prone to overfitting in a low dimensional dataset, and in context of a higher dimensional dataset, regularization can be used to avoid overfitting. Moreover, new data can be updated using stochastic gradient descent (SGD).

But LR also has limitations as it only address linear separable data for non-linear problems transformation is required. Features used for training model should also be carefully extracted otherwise noise will make the probabilistic predictions may be incorrect. LR requires a large dataset and sufficient training examples for all the categories it needs to identify. Lastly, each training tuples must be isolated to all others, because relationship between any of them will make model give more importance to these relative examples.

3.1.3 Support Vector Machine

A SVM model takes data points and outputs the hyperplane (a line in context of two dimensioal dataset) that best separates the classes.

The best hyperplane is the one has largest distance to neares data point of each class. In other words, SVM maximizes the margins from both class.

With nonlinear data, additional dimensions will be required. SVM can classify vectors in multidimensional space.

3.1.4 Naive Bayes

NB classifier based on applying Bayes' theorem. Bayes' theorem finds out the probability of an event if the occurence another event is probably knows, which is defined as:

$$\Pr(A|B) = \frac{\Pr(B|A)\Pr(A)}{\Pr(B|A)\Pr(A) + \Pr(B|\neg A)\Pr(\neg A)} f(x)$$

where A and B are events.

NB classifiers assumed all the features are independent and equally contributed to final result. Probability of feature y given a set of X features as $x_1, x_2, ..., x_n$ is calculated by formula:

$$P(y|x_1, x_2, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

We find the probability of all possible values of class y and choose the maximum, which can be expressed as:

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y) \tag{3.1}$$

Although completely independent features are barely exist, in practical terms NB is widely used since it's highly scalable, time-saving and suitable for categorical input variables.

3.1.5 Decision Tree

A decision tree is used to represent decisions and decision-making process. The tree can be explained as the leaves, which is final outcome, and the decision node, which is the place data splits.

For classification purpose, the outcomes (or leaves) are corresponding to features, which is selected throughout the tree. Begin with the root, or the starting node, we find out the best decision (e.g., best attribute) by applying selection measures (Information Gain, Gini Index, etc.) in the dataset, then divide dataset into subsets and recursively perform the process until the nodes can not be further divided. The leaf nodes now is the final outcome, and the total process make out the model.

3.1.6 Random Forest

RF is the combination of multiple DT that opearate as an ensemble, which results in more reliable and stable prediction. Each DT finds out a class prediction, and the most chosen becomes the total model's prediction. As those DTs are relatively independent, RF is protected from individual DT's error (unless all the DTs are wrong in the same way).

RF works well with categorical variables. Missing or imbalanced values can be handled easily. Noise or new data point also not a problem as it can only affect some of DTs but not to entire forest.

3.1.7 Convolutional Neural Network

CNN is a popular Deep Learning model. A typical CNN consists of a set of basic layers including: convolution layer with kernels, pooling layer, fully connected layer, which are linked together in a certain order.

Convolution is the first class to extract the features from the input sentence. Convolution maintains relationship between words by exploring sentence feature by using small segments of input data. It is a mathematical operation that takes two inputs such as a matrix of words and filter of kernels.

The maxpooling layer will reduce the number of parameters when the word count is too large. Spatial aggregation is also known as sub-sampling, which reduces the dimensionals of each map but retains important information.

The last layer flattens its matrix into vector and brings it to a fully connected layer like a neural network, which combines the features together to create a model.

Finally, we have a activation function like softmax or sigmoid to to classify with probability value from 0 to 1.

3.2 Model overview

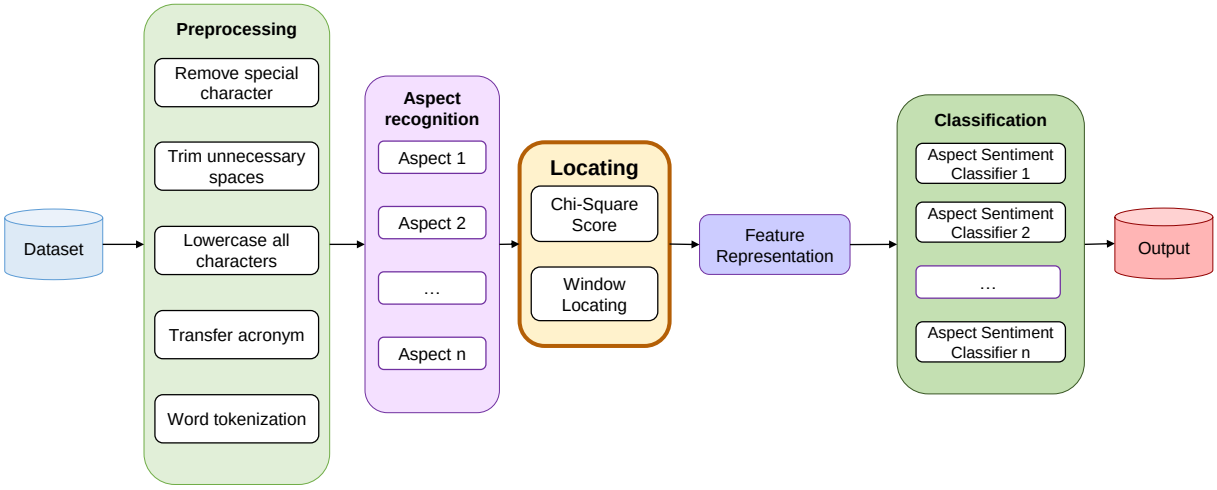

Figure 3.1: The aspect and sentiment analysis pipeline

The main objective of our model is to address the second sub-task of the ABSA problem as it must be able to perform the aspect polarity task. In this case, the system should identify which of two polarity labels - positive or negative could be assigned with the corresponding aspect. Figure 4.1 shows the position of our problem in the opinion mining overall architecture. In which aspect recognition is considered as the preceding problem of sentiment classification and is outside the scope of this study. In this report, we assume that aspect recognition is completed before.

Our aspect-oriented sentiment analysis model consists of four main phases, illustrated in Figure 4.2: reprocessing, locating, representation and classification. The first phase helps cleaning and preparing data for classification, the second one applies word-window method to find out the words which has a high ability to decide which sentimental status the aspects are. Following to that, we tested different ways to represent words with a view to extract meaningful features from data. Finally, the last phase is our application of multiple classifiers, including both classic ones like SVM, Random Forest, etc. and modern one as Deep Learning, which will be all described within this section.

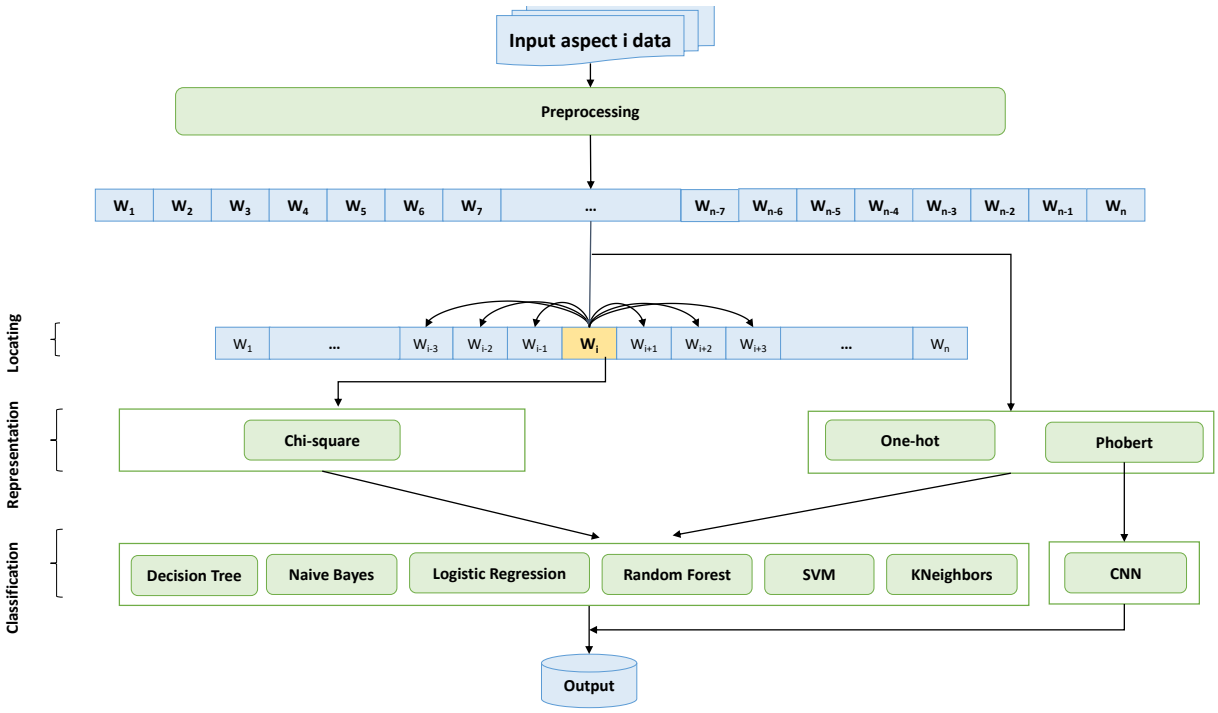

Figure 3.2: Implementation steps for a single aspect

3.3 Preprocessing

Preprocessing is one of the key steps in every natural language processing problem as it transforms data into usable one which machine can easily interprete. Since the characteristics of the input data are raw text collected from websites, the input text (from the dataset contains the specific aspect that we have gathered and processed in Chapter **??** above) can have noise which can harmful to machine learning performance such as special characters, spelling mistakes, spacing errors, etc. To standardize the data and reduce the amount of noise, preprocessing process is applied according to the following steps:

- **Step 1:** Special characters (including punctuations) were completely removed.
- **Step 2:** Trim unnecessary spaces made by spacing errors.
- **Step 3:** Lowercase all characters.
- **Step 4:** Transfer acronyms to the full form of them and translated from context.
- **Step 5:** Word tokenization. Tokenization is essentially splitting a phrase, sentence, paragraph, or an entire text document into smaller units called tokens, such as individual words or terms. By tokenization, the meaning of the text can be interpret

Input	**Bỉm mềm , chất lượng tốt , hút dc nhiều nhưng giao hàng khá chậm.**
Step 1	Bỉm mềm chất lượng tốt hút dc nhiều nhưng giao hàng khá chậm
Step 2	Bỉm mềm chất lượng tốt hút dc nhiều nhưng giao hàng khá chậm
Step 3	bỉm mềm chất lượng tốt hút dc nhiều nhưng giao hàng khá chậm
Step 4	bỉm mềm chất lượng tốt hút được nhiều nhưng giao hàng khá chậm
Step 5	**bỉm mềm chất_lượng tốt hút nhiều giao_hàng chậm**

Table 3.1: Step-by-step preprocessing illustration

easier by some analysing methods such as count the number of words appeared, the frequency of the word, and so on.

VnCoreNLP (Vu et al., 2018 [3]) is used for all pre-processing steps.

3.4 Locating

3.4.1 Chi-Squared Test for Feature Selection

In statistics, the Chi-squared test is used to determine whether two categorical variables independent or related. In feature selection, the two variables are the observations of the feature and the occurrence of the class. The outcome of the test is a test statistic that has a chi-squared distribution and can be clarified or fail to reject the assumption or null hypothesis H_0 that the observed and expected frequencies are equal.

Given a document D, we estimate the (χ^2) value and rank them by their score:

$$\chi^2 = \sum_{t=1}\sum_{c=1} \frac{(O_{t,c} - E_{t,c})^2}{E_{t,c}} = N \sum_{t,c} p_t p_c \left(\frac{(O_{t,c}/N) - p_t p_c}{p_t p_c} \right)^2$$

where

χ^2 = Pearson's cumulative test statistic, which asymptotically approaches a χ^2 distribution.

$O_{t,c}$ = the number of observations of type t in class c.

N = total number of observations.

$E_{t,c} = Np_{t,c}$ = the expected (theoretical) count of type t in class c, asserted by the null hypothesis that the fraction of type t in class c in the population is $p_{t,c}$

For each feature, a corresponding high χ^2 score indicates that the null hypothesis of independence (meaning the document class has no impact over the term's frequency) should be dismissed and the occurrence of the term and class are dependent, therefore we should select the feature for classification. In other words, using this method remove the feature that are most likely autonomous of class and consequently unessential for classification.

We use Scikit-learn (Pedregosa et al., 2011 [2]) as it provide multiple feature selection methods, including Chi-Squared Test. Scikit-learn gives a *SelectKBest* class that can be used with various statistical tests. It will rank the features with the statistical test that we've determined (Chi-Squared Test in detail) and select the top k performing ones (implying that these terms is viewed as more relevant to the task than the others). These top performing features will be used for locating and one of the representation method that we will described below.

3.4.2 Word-Window Locating

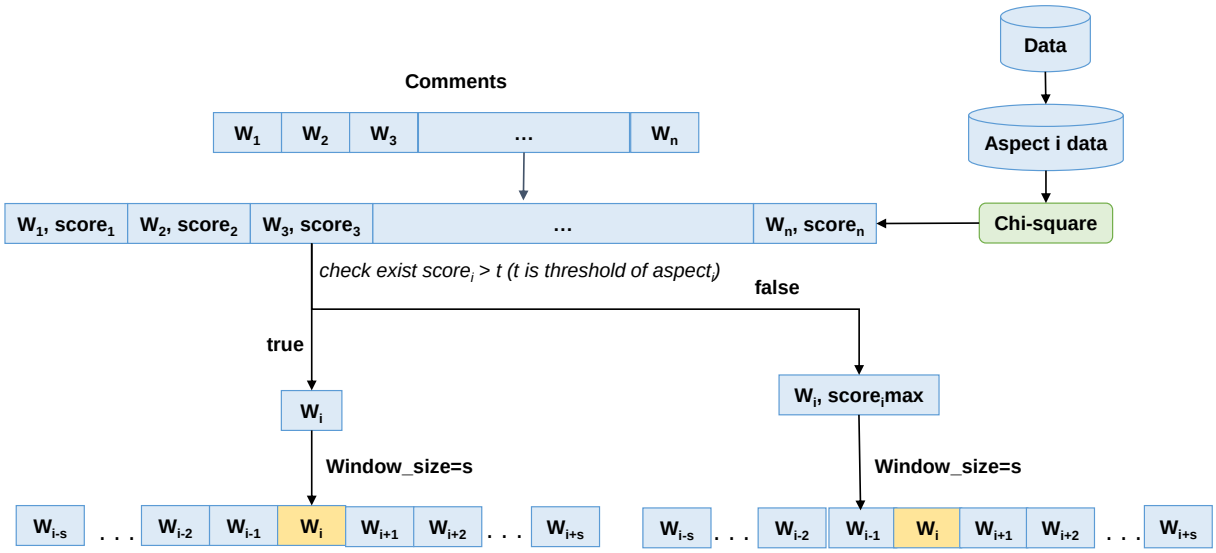

Figure 3.3: Word-Window Locating process

We use chi-squared rank as calculated above to weight the words in the comment, with a view to select out which word play the important role on determine aspect's sentiment.

Given a sentence as a set of word $W = \{w_0, w_1, w_2, ..., w_n\}$ and a set of class $C =$

$\{c_0, c_1, c_2, ..., c_m\}$, we simply define score $s_{i,j}$ as χ^2 score of the word w_i in class c_j. For each aspect c_j , we choose a threshold s_{c_j}, and ignore all the words which have lower score than that threshold. If the sentence has no word with score higher than threshold, the word with highest score will be selected. Based on calculation above, we take out 3 words before and after the high-score word w_i. This combination, or *window*, can be illustrated as: $\{w_{i-3}, w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}, w_{i+3}\}$

The results collected is helpful for classifier as WWL removed the non-related words of each aspect and its sentiment; therefore, only relevant words extracted to put in learning model.

3.5 Representation

3.5.1 One Hot encoding

Every word which are part of the text data are written in the form of vectors, constituting only of 1 and 0 (1 if word in vocabulary else 0). Although it does not highlight the importance of words in sentiment classification, but it has yielded quite surprising results. Further information will be provided in Section 5.2.

3.5.2 Chi-Squared

We use word-level χ^2 as calculated above to weight the words in the vocabulary, and use that vocabulary to represent data. Then we proceed to filter each aspect's vocabulary manually to reduce dimensions used for classfiers. The results using this data representation method shown in Section 5.2.

3.5.3 Word Embedding using PhoBERT

BERT BERT (Bidirectional Encoder Representations from Transformers) is understood as a pre-trained model, which learns vectors that represent two dimensions of words while models such as Word2vec, fastText find a vector that represents each word based on a large set of materials, so it does not represent the diversity of context. BERT has succeeded in improving recent work in finding representatives of words in digital spaces (spaces that computers can understand) through its context.

Transformer is an attention mechanism that learns the correlation between words (or part of a word) in a text. Transformer consists of two main parts: Encoder and

Decoder, encoder that reads input data and decoder makes predictions. In this sitution, BERT uses only Encoder.

BERT is used to support other problems in the field of natural language processing such as emotional classification, amantic analysis, spam filtering, news classification, etc. Our proposed work use BERT for the problem of sentment classification.

PhoBERT Pre-trained PhoBERT models are the state-of-the-art language models for Vietnamese (Dat and Anh [1]). PhoBERT uses VnCoreNLP to separate words for input data before passing encoder.

PhoBERT output consists of 2 layers: the first one is the last hidden layer created by token embedding, sentences embedding, transformer positional embedding into a 3D vector, while the second is the layer that has been pooling into a 2D vector.

In each string of words after using PhoBERT, we have two ways of expressing the word and sentence level:

- Represented by word level: each word will be represented as a 768-dimensional vector, so in total a $n*768$ matrix will represent a sentence.
- Represented by sentence level: the whole sentence is also represented by a 768-dimensional vector.

Typically, a sentence after being represented by the matrix bar will be spread through the convolution layer + nonlinear layer first (RELU), after which the calculated values will spread through the pooling layer, multiple kernels.

Specifically, the word vectors from w_i to w_{i+j-1} as $[w_i, w_{i+1}, w_{i+2}, ..., w_{i+j-1}]$ will be combine to represent one feature. Afterwards, a filter is applied to the specific k-word vector (in the proposed work we use k = 2, 3 and 4) to captured most useful features for sentiment detection.

Feature c_i is generated from a nonlinear activation function (hyperbolic tangent or ReLU)

$$c_i = f(F(w_{i,i+k} + b) \tag{3.2}$$

where b is a bias, a vector performs as a additional function to the input.

We will get 1 characteristic matrix for a specific feature: $c = [c_1, c_2, c_3, ..., c_h]$ Then, maxpooling (take out the maximum characteristic c_i among the matrix) is applied to

choose the most important features. The reason for choosing the highest c is because the input will inevitably map to a fixed output size so that it is fully connected, the input size is reduced but still maintains important properties.

The convolution filer number is certain with different width and slips across the entire matrix to get all the properties.

Chapter 4

Methods

In this chapter, we firstly present the methodology for our approaches, then describe our aspect-oriented model in detail, begin with an overview of the whole architecture, followed by the explanation of how each of the components work; lastly, we introduce the parameter system used in the model.

4.1 Theoretical Basis

4.1.1 K-nearest Neighbors

KNN is one of the most straightforward supervised algorithms, but effective in dealing some problems, in both classification and regression. KNN does not study anything from training data (therefore it's considered as a lazy learning method)

In classification problem, a new data point's class is derive from k nearest data point from the training set. In general, this label can be calculated by the label of nearest data point $k = 1$ $(weights = uniform)$ or average weight of the nearest points $(weights = distance)$, or a relationship between these weight.

KNN is simple as only two criteria needed: value of k and the function to calculate distance. KNN does not require any training period because it does not derived any function from training data and only make real-time prediction when evaluating, so this make the algorithm run much faster, especially considering small dataset. New data can be also add anytime without any affection to the algorithm's accuracy.

Despite lightning performance in small dataset, as the data size enlarges, KNN is not beneficial for both memory and time. The algorithm must remember all the training

data for label calculation, so it will become extensively slow as size of dataset grows.

4.1.2 Logistics Regression

LR, in its basic form, uses a logistic function (e.g., sigmoid, tanh) to model a binary dependent variable. The prediction score of LR is calculated by formula:

$$f(x) = \theta(\mathbf{w}^T \mathbf{x})$$

where

θ: logistics function (e.g., sigmoid, tanh, etc.)

The cost function for LR is defined as:

$$L = \sum_{D} -ylog(y') - (1-y)log(1-y'))$$

where

D: dataset, containing of labeled tuples (x, y)

y: the label in a assigned example, which is either be 0 or 1.

y' is the predicted value, which is between 0 and 1, given features in x.

LR is one of the easiest ML algorithms as it is easy to implement, interpret, and very efficient to train. Traning a model with LR doesn't need high computation effort. LR also less prone to overfitting in a low dimensional dataset, and in context of a higher dimensional dataset, regularization can be used to avoid overfitting. Moreover, new data can be updated using stochastic gradient descent (SGD).

But LR also has limitations as it only address linear separable data for non-linear problems transformation is required. Features used for training model should also be carefully extracted otherwise noise will make the probabilistic predictions may be incorrect. LR requires a large dataset and sufficient training examples for all the categories it needs to identify. Lastly, each training tuples must be isolated to all others, because relationship between any of them will make model give more importance to these relative examples.

4.1.3 Support Vector Machine

A SVM model takes data points and outputs the hyperplane (a line in context of two dimensioal dataset) that best separates the classes.

The best hyperplane is the one has largest distance to neares data point of each class. In other words, SVM maximizes the margins from both class.

With nonlinear data, additional dimensions will be required. SVM can classify vectors in multidimensional space.

4.1.4 Naive Bayes

NB classifier based on applying Bayes' theorem. Bayes' theorem finds out the probability of an event if the occurence another event is probably knows, which is defined as:

$$\Pr(A|B) = \frac{\Pr(B|A)\Pr(A)}{\Pr(B|A)\Pr(A) + \Pr(B|\neg A)\Pr(\neg A)} f(x)$$

where A and B are events.

NB classifiers assumed all the features are independent and equally contributed to final result. Probability of feature y given a set of X features as $x_1, x_2, ..., x_n$ is calculated by formula:

$$P(y|x_1, x_2, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

We find the probability of all possible values of class y and choose the maximum, which can be expressed as:

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y) \tag{4.1}$$

Although completely independent features are barely exist, in practical terms NB is widely used since it's highly scalable, time-saving and suitable for categorical input variables.

4.1.5 Decision Tree

A decision tree is used to represent decisions and decision-making process. The tree can be explained as the leaves, which is final outcome, and the decision node, which is the place data splits.

For classification purpose, the outcomes (or leaves) are corresponding to features, which is selected throughout the tree. Begin with the root, or the starting node, we find out the best decision (e.g., best attribute) by applying selection measures (Information Gain, Gini Index, etc.) in the dataset, then divide dataset into subsets and recursively perform the process until the nodes can not be further divided. The leaf nodes now is the final outcome, and the total process make out the model.

4.1.6 Random Forest

RF is the combination of multiple DT that opearate as an ensemble, which results in more reliable and stable prediction. Each DT finds out a class prediction, and the most chosen becomes the total model's prediction. As those DTs are relatively independent, RF is protected from individual DT's error (unless all the DTs are wrong in the same way).

RF works well with categorical variables. Missing or imbalanced values can be handled easily. Noise or new data point also not a problem as it can only affect some of DTs but not to entire forest.

4.1.7 Convolutional Neural Network

CNN is a popular Deep Learning model. A typical CNN consists of a set of basic layers including: convolution layer with kernels, pooling layer, fully connected layer, which are linked together in a certain order.

Convolution is the first class to extract the features from the input sentence. Convolution maintains relationship between words by exploring sentence feature by using small segments of input data. It is a mathematical operation that takes two inputs such as a matrix of words and filter of kernels.

The maxpooling layer will reduce the number of parameters when the word count is too large. Spatial aggregation is also known as sub-sampling, which reduces the dimensionals of each map but retains important information.

The last layer flattens its matrix into vector and brings it to a fully connected layer like a neural network, which combines the features together to create a model.

Finally, we have a activation function like softmax or sigmoid to to classify with probability value from 0 to 1.

4.2 Model overview

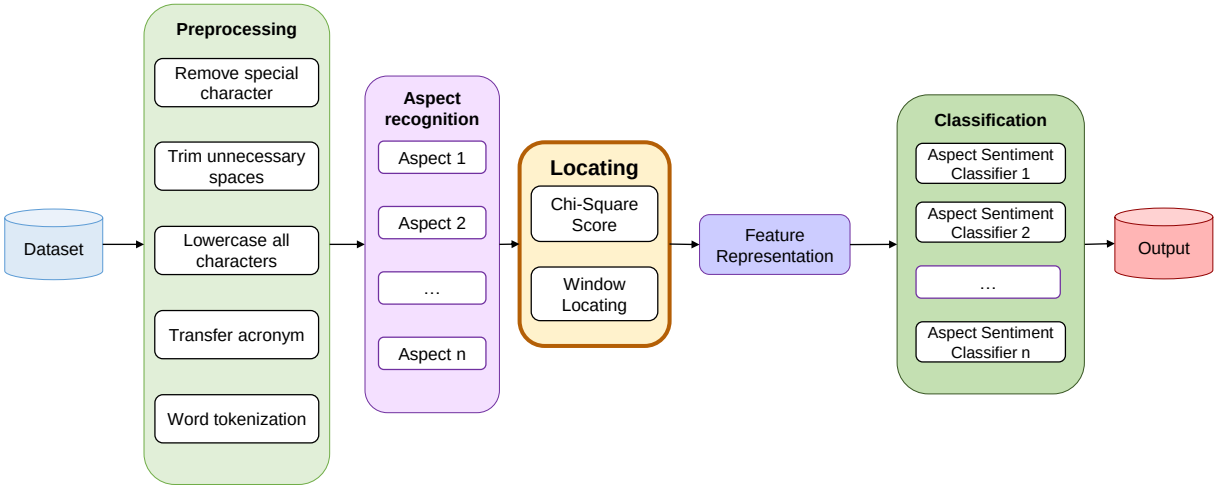

Figure 4.1: The aspect and sentiment analysis pipeline

The main objective of our model is to address the second sub-task of the ABSA problem as it must be able to perform the aspect polarity task. In this case, the system should identify which of two polarity labels - positive or negative could be assigned with the corresponding aspect. Figure 4.1 shows the position of our problem in the opinion mining overall architecture. In which aspect recognition is considered as the preceding problem of sentiment classification and is outside the scope of this study. In this report, we assume that aspect recognition is completed before.

Our aspect-oriented sentiment analysis model consists of four main phases, illustrated in Figure 4.2: reprocessing, locating, representation and classification. The first phase helps cleaning and preparing data for classification, the second one applies word-window method to find out the words which has a high ability to decide which sentimental status the aspects are. Following to that, we tested different ways to represent words with a view to extract meaningful features from data. Finally, the last phase is our application of multiple classifiers, including both classic ones like SVM, Random Forest, etc. and modern one as Deep Learning, which will be all described within this section.

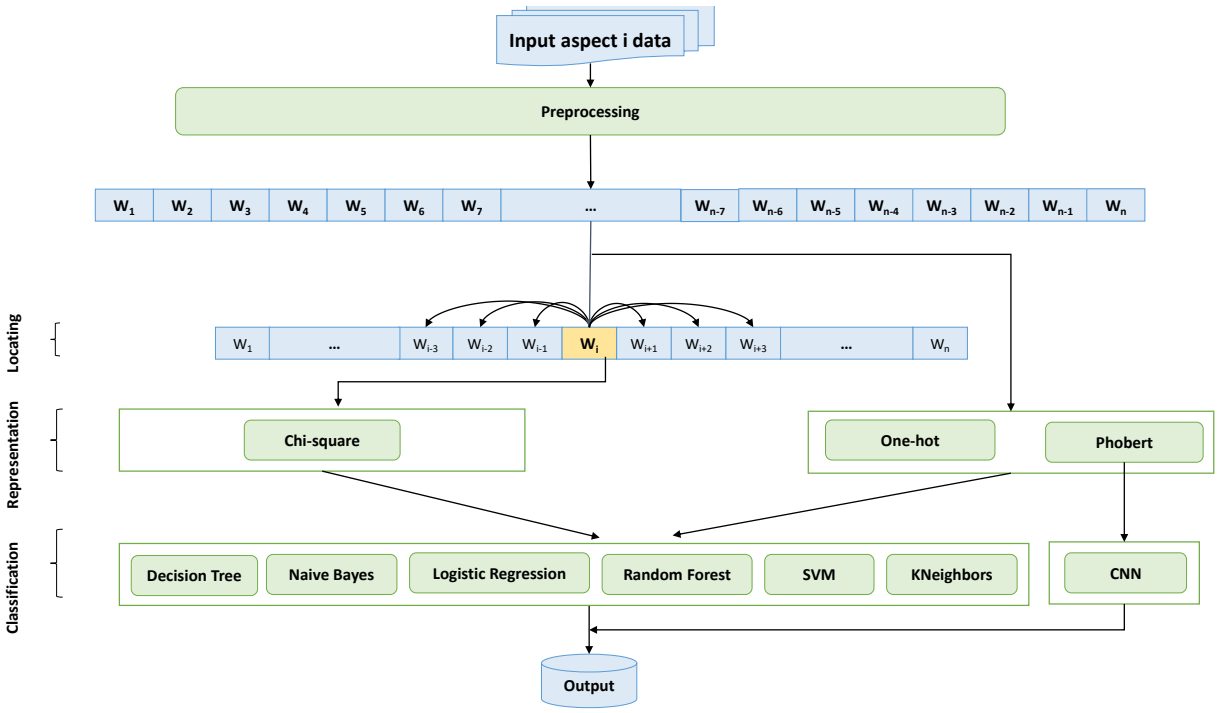

Figure 4.2: Implementation steps for a single aspect

4.3 Preprocessing

Preprocessing is one of the key steps in every natural language processing problem as it transforms data into usable one which machine can easily interprete. Since the characteristics of the input data are raw text collected from websites, the input text (from the dataset contains the specific aspect that we have gathered and processed in Chapter **??** above) can have noise which can harmful to machine learning performance such as special characters, spelling mistakes, spacing errors, etc. To standardize the data and reduce the amount of noise, preprocessing process is applied according to the following steps:

- **Step 1:** Special characters (including punctuations) were completely removed.
- **Step 2:** Trim unnecessary spaces made by spacing errors.
- **Step 3:** Lowercase all characters.
- **Step 4:** Transfer acronyms to the full form of them and translated from context.
- **Step 5:** Word tokenization. Tokenization is essentially splitting a phrase, sentence, paragraph, or an entire text document into smaller units called tokens, such as individual words or terms. By tokenization, the meaning of the text can be interpret

Input	**Bỉm mềm , chất lượng tốt , hút dc nhiều nhưng giao hàng khá chậm.**
Step 1	Bỉm mềm chất lượng tốt hút dc nhiều nhưng giao hàng khá chậm
Step 2	Bỉm mềm chất lượng tốt hút dc nhiều nhưng giao hàng khá chậm
Step 3	bỉm mềm chất lượng tốt hút dc nhiều nhưng giao hàng khá chậm
Step 4	bỉm mềm chất lượng tốt hút được nhiều nhưng giao hàng khá chậm
Step 5	**bỉm mềm chất_lượng tốt hút nhiều giao_hàng chậm**

Table 4.1: Step-by-step preprocessing illustration

easier by some analysing methods such as count the number of words appeared, the frequency of the word, and so on.

VnCoreNLP (Vu et al., 2018 [3]) is used for all pre-processing steps.

4.4 Locating

4.4.1 Chi-Squared Test for Feature Selection

In statistics, the Chi-squared test is used to determine whether two categorical variables independent or related. In feature selection, the two variables are the observations of the feature and the occurrence of the class. The outcome of the test is a test statistic that has a chi-squared distribution and can be clarified or fail to reject the assumption or null hypothesis H_0 that the observed and expected frequencies are equal.

Given a document D, we estimate the (χ^2) value and rank them by their score:

$$\chi^2 = \sum_{t=1}\sum_{c=1} \frac{(O_{t,c} - E_{t,c})^2}{E_{t,c}} = N \sum_{t,c} p_t p_c \left(\frac{(O_{t,c}/N) - p_t p_c}{p_t p_c} \right)^2$$

where

χ^2 = Pearson's cumulative test statistic, which asymptotically approaches a χ^2 distribution.

$O_{t,c}$ = the number of observations of type t in class c.

N = total number of observations.

$E_{t,c} = Np_{t,c}$ = the expected (theoretical) count of type t in class c, asserted by the null hypothesis that the fraction of type t in class c in the population is $p_{t,c}$

For each feature, a corresponding high χ^2 score indicates that the null hypothesis of independence (meaning the document class has no impact over the term's frequency) should be dismissed and the occurrence of the term and class are dependent, therefore we should select the feature for classification. In other words, using this method remove the feature that are most likely autonomous of class and consequently unessential for classification.

We use Scikit-learn (Pedregosa et al., 2011 [2]) as it provide multiple feature selection methods, including Chi-Squared Test. Scikit-learn gives a *SelectKBest* class that can be used with various statistical tests. It will rank the features with the statistical test that we've determined (Chi-Squared Test in detail) and select the top k performing ones (implying that these terms is viewed as more relevant to the task than the others). These top performing features will be used for locating and one of the representation method that we will described below.

4.4.2 Word-Window Locating

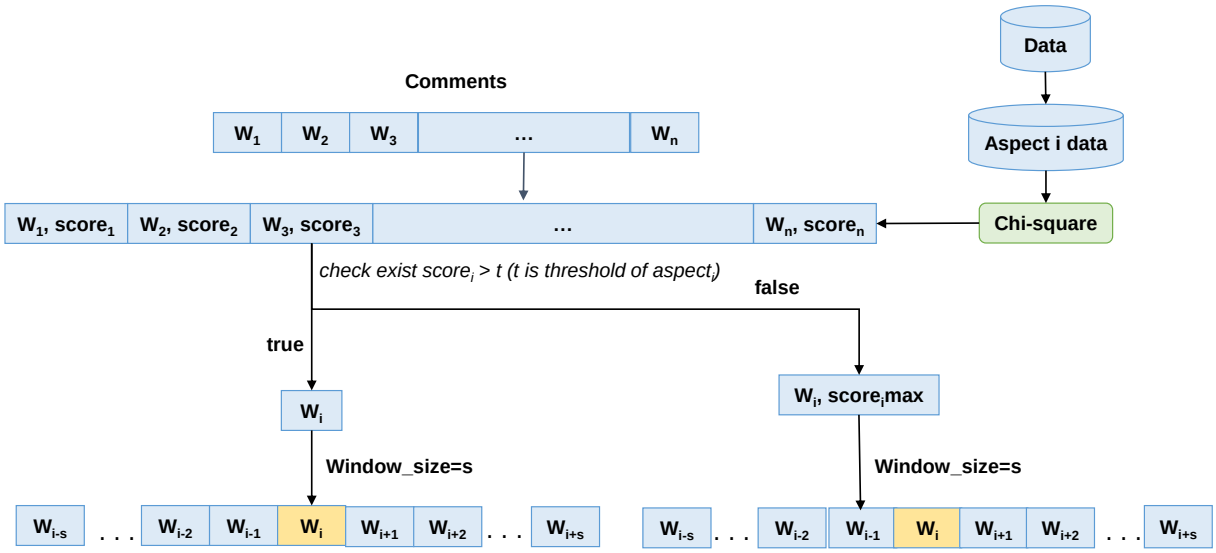

Figure 4.3: Word-Window Locating process

We use chi-squared rank as calculated above to weight the words in the comment, with a view to select out which word play the important role on determine aspect's sentiment.

Given a sentence as a set of word $W = \{w_0, w_1, w_2, ..., w_n\}$ and a set of class $C =$

$\{c_0, c_1, c_2, ..., c_m\}$, we simply define score $s_{i,j}$ as χ^2 score of the word w_i in class c_j. For each aspect c_j , we choose a threshold s_{c_j}, and ignore all the words which have lower score than that threshold. If the sentence has no word with score higher than threshold, the word with highest score will be selected. Based on calculation above, we take out 3 words before and after the high-score word w_i. This combination, or *window*, can be illustrated as: $\{w_{i-3}, w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}, w_{i+3}\}$

The results collected is helpful for classifier as WWL removed the non-related words of each aspect and its sentiment; therefore, only relevant words extracted to put in learning model.

4.5 Representation

4.5.1 One Hot encoding

Every word which are part of the text data are written in the form of vectors, constituting only of 1 and 0 (1 if word in vocabulary else 0). Although it does not highlight the importance of words in sentiment classification, but it has yielded quite surprising results. Further information will be provided in Section 5.2.

4.5.2 Chi-Squared

We use word-level χ^2 as calculated above to weight the words in the vocabulary, and use that vocabulary to represent data. Then we proceed to filter each aspect's vocabulary manually to reduce dimensions used for classfiers. The results using this data representation method shown in Section 5.2.

4.5.3 Word Embedding using PhoBERT

BERT BERT (Bidirectional Encoder Representations from Transformers) is understood as a pre-trained model, which learns vectors that represent two dimensions of words while models such as Word2vec, fastText find a vector that represents each word based on a large set of materials, so it does not represent the diversity of context. BERT has succeeded in improving recent work in finding representatives of words in digital spaces (spaces that computers can understand) through its context.

Transformer is an attention mechanism that learns the correlation between words (or part of a word) in a text. Transformer consists of two main parts: Encoder and

Decoder, encoder that reads input data and decoder makes predictions. In this sitution, BERT uses only Encoder.

BERT is used to support other problems in the field of natural language processing such as emotional classification, amantic analysis, spam filtering, news classification, etc. Our proposed work use BERT for the problem of sentment classification.

PhoBERT Pre-trained PhoBERT models are the state-of-the-art language models for Vietnamese (Dat and Anh [1]). PhoBERT uses VnCoreNLP to separate words for input data before passing encoder.

PhoBERT output consists of 2 layers: the first one is the last hidden layer created by token embedding, sentences embedding, transformer positional embedding into a 3D vector, while the second is the layer that has been pooling into a 2D vector.

In each string of words after using PhoBERT, we have two ways of expressing the word and sentence level:

- Represented by word level: each word will be represented as a 768-dimensional vector, so in total a $n*768$ matrix will represent a sentence.
- Represented by sentence level: the whole sentence is also represented by a 768-dimensional vector.

Typically, a sentence after being represented by the matrix bar will be spread through the convolution layer + nonlinear layer first (RELU), after which the calculated values will spread through the pooling layer, multiple kernels.

Specifically, the word vectors from w_i to w_{i+j-1} as $[w_i, w_{i+1}, w_{i+2}, ..., w_{i+j-1}]$ will be combine to represent one feature. Afterwards, a filter is applied to the specific k-word vector (in the proposed work we use k = 2, 3 and 4) to captured most useful features for sentiment detection.

Feature c_i is generated from a nonlinear activation function (hyperbolic tangent or ReLU)

$$c_i = f(F(w_{i,i+k} + b) \tag{4.2}$$

where b is a bias, a vector performs as a additional function to the input.

We will get 1 characteristic matrix for a specific feature: $c = [c_1, c_2, c_3, ..., c_h]$ Then, maxpooling (take out the maximum characteristic c_i among the matrix) is applied to

choose the most important features. The reason for choosing the highest c is because the input will inevitably map to a fixed output size so that it is fully connected, the input size is reduced but still maintains important properties.

The convolution filer number is certain with different width and slips across the entire matrix to get all the properties.

Chapter 5

Experiments and Results

5.1 Experiment Setup

5.1.1 Experimental Data

In preparation to evaluate the effectiveness of our model, we proposed experiments on the Baby Care dataset collected from Shopee and Tiki, which has been described previously.

5.1.2 Baseline Methods

We compare various combinations of classifiers and different representation methods to figure out which one is most effective for our situation. The evaluation process is:

(1) Classic models: Logistics Regression, Random Forest, Decision Tree, Naive Bayes, K-nearest Neighbors and Support Vector Machine combine with different representation methods: One-Hot, Chi-Squared, PhoBERT.

(2) Logistics Regression with different representation methods.

(3) General comparison with/without WWL to find out the best model.

5.1.3 Evaluation Metrics

Precision Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. High precision relates to the low false positive rate.

$$Precision = \frac{TP}{TP + FP} \tag{5.1}$$

where

TP = True Positive - correctly predicted positive values which means that the value of actual class is 1 and value of predicted class is also 1.

FP = False Positive - actual class is 0 and predicted class is 1.

Recall Recall is the ratio of correctly predicted positive observations to the all observations in actual class. High recall is synonymous with the low false negative rate.

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

where

FP = False Negative - actual class is 1 and predicted class is 0.

F1-sscore The F1 is a way of combining the precision and recall of the model. The higher the F1 score the better, with 0 being the worst possible, which means the precision or recall is zero; and 1 being the best, indicating perfect precision and recall. F1-score is defined as the harmonic mean of the model's precision and recall.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{5.3}$$

Micro-average and Macro-average Performance Micro Average and Macro Average Performance are used to evaluate multi-label classification model. A macro-average will compute the metric independently for each class and then take the average, whereas a micro-average will aggregate the contributions of all classes to compute the average metric. Micro- and macro-average for Precision, Recall and F1-score is calculated by the formulas as shown in the Equations from 5.4 to 5.7.

$$P_{macro} = \frac{\sum_i P_i}{N} \qquad R_{macro} = \frac{\sum_i R_i}{N} \qquad F1_{macro} = \frac{\sum_i F1_i}{N} \tag{5.4}$$

$$P_{micro} = \frac{\sum_j TP_j}{\sum_j TP_j + \sum_j FP_j} \tag{5.5}$$

$$R_{micro} = \frac{\sum_j TP_j}{\sum_j TP_j + \sum_j FN_j} \tag{5.6}$$

$$F1_{micro} = \frac{2 * P_{micro} * R_{micro}}{P_{micro} + R_{micro}} \tag{5.7}$$

5.2 Experiment Result and Analysis

In context of facing imbalance data (positive outnumbered negative) as analysed in Section **??**, we appreciate the results of class negative in particular. In this section, we will use Micro/Macro-average metrics calculated for negative class as the measure of evaluation.

5.2.1 Classic Models with One-Hot Representation Method.

With the OH data representation method, we use the Mother & Baby data collected from Tiki as the input data for traditional machine learning models. Applying this simple method surprisingly brings fairly good results for such unbalanced data (compared to the methods below) but the negative value of some aspects is not yet assigned.

Based on the results of applying models, we find that the model using the LR method has the most outstanding results. We will use a combination of this method and the data representation methods below to compare results.

Evaluation results are illustrated in table 5.1.

	SVM	***DT***	***KN***	***LR***	***NB***	***RF***
micro-p	0,587	0,622	0,541	0,810	0,205	0,818
micro-r	0,228	0,599	0,370	0,525	0,746	0,500
micro-f1	0,329	0,610	0,440	**0,637**	0,322	0,621
macro-p	0,330	0,422	0,389	0,541	0,183	0,548
macro-r	0,159	0,397	0,258	0,351	0,641	0,329
macro-f1	0,176	0,406	0,299	0,405	0,277	0,395

Table 5.1: Micro/Macro-average for class negative of OH + traditional models

5.2.2 Logistics Regression with Different Representation Methods

We use the model using the LR method for comparison since it is the model with the best results as calculated above. Mother & Baby data collected from Tiki are used to evaluated results among three different representation methods: PhoBERT, CS and OH. OH and PhoBERT showed relatively good results, but in general CS has the best performance. Table 5.2 illustrates micro- and macro-average performance for class negative of LR with multiple representation methods.

	PhoBERT	*CS*	*OH*
micro-p	0,764	0,704	0,810
micro-r	0,540	0,660	0,525
micro-f1	0,633	**0,682**	0,637
macro-p	0,506	0,560	0,541
macro-r	0,368	0,532	0,351
macro-f1	0,423	**0,545**	0,405

Table 5.2: Micro/Macro-average for class negative of LR with representation methods

5.2.3 Chi-Squared with/without Word-Window Locating compared with CNN + PhoBERT

	CS + WWL	*CS*	*CNN + PhoBERT*
micro-p	0,738	0,704	0,801
micro-r	0,728	0,660	0,747
micro-f1	0,733	0,682	**0,773**
macro-p	0,781	0,560	0,512
macro-r	0,632	0,532	0,518
macro-f1	**0,674**	0,545	0,515

Table 5.3: Micro/Macro-average of CS with/without WWL and CNN + PhoBERT

We continue to use the model using LR method to compare between CS, CS + WWL and CNN + PhoBERT.

The results indicate that CS + WWL outperformed CS alone in every figures. In comparison of CS + WWL and CNN + PhoBERT, it is indicated that micro-F1 of CNN + PhoBERT is slightly better with a percentage of 4% but CS + WWL surpassed CNN + PhoBERT with a significant percentage of 16%.

Evaluation proves effective use of the WWL method.

Table 5.3 shows micro- and macro-average performance of CS with/without WWL and CNN + PhoBERT, while table 5.4 shows CS + WWL vs. CNN + PhoBERT detailed statistical results in all aspects.

Method	Class	Price	Service	Safety	Quality	Delivery	Authenticity
CS + WWL	Positive	0.983	0.757	0.892	0.907	0.928	0.960
	Negative	0.571	0.804	0.774	0.718	0.676	0.500
CNN	Positive	0.978	0.875	0.759	0.940	0.943	0.943
	Negative	**0.000**	0.884	0.632	0.792	0.763	**0.000**

Table 5.4: WWL + CS vs. CNN + PhoBERT detailed statistics in all aspects

5.2.4 WWL + CS detailed statistics in all domains

Given the proportional difference between the positive and negative classes of different datasets in the domain and the data source in the item (item name), we find that the model uses WWL with the CS method of representing the level data showed quite stable results. The statistical results are described in table 5.5.

	mb_tiki	***mb_shopee***	***tech_tiki***	***tech_shopee***
micro-p	0,738	0,615	0,707	0,709
micro-r	0,728	0,635	0,621	0,629
micro-f1	0,733	0,625	0,661	0,667
macro-p	0,781	0,541	0,582	0,593
macro-r	0,632	0,462	0,510	0,572
macro-f1	0,674	0,482	0,540	0,545

Table 5.5: CS + WWL detailed statistics for class negative in all domains

5.2.5 Error Analysis

- Unbalanced data results affect model training leading to low results for the indicated aspect.
- Lack of summarized data.
- Errors occur in the data annotation phase due to manual process.
- The investigation of the data is not in-depth, yield threshold selected for locating not matching the data.

5.3 Demonstrations

5.3.1 Data Visualization

We use Kibana and Elasticsearch to analyze and visualize data based on their labels and polarity. Figure 5.1 and 5.2 shows how our dataset are visualized based on labels and keywords.

Figure 5.1: Dataset visualization based on labels and polarity

Figure 5.2: Dataset visualization based on keywords

5.3.2 Web Application Demonstration

With a view to bring the model into practical situation, we built a web interface for users to manipulate easily. The users will put in data in two ways: insert each comment by the 'Add' button (figure 5.3) or import a CSV file containing comment data along with the respective aspect (figure 5.4).

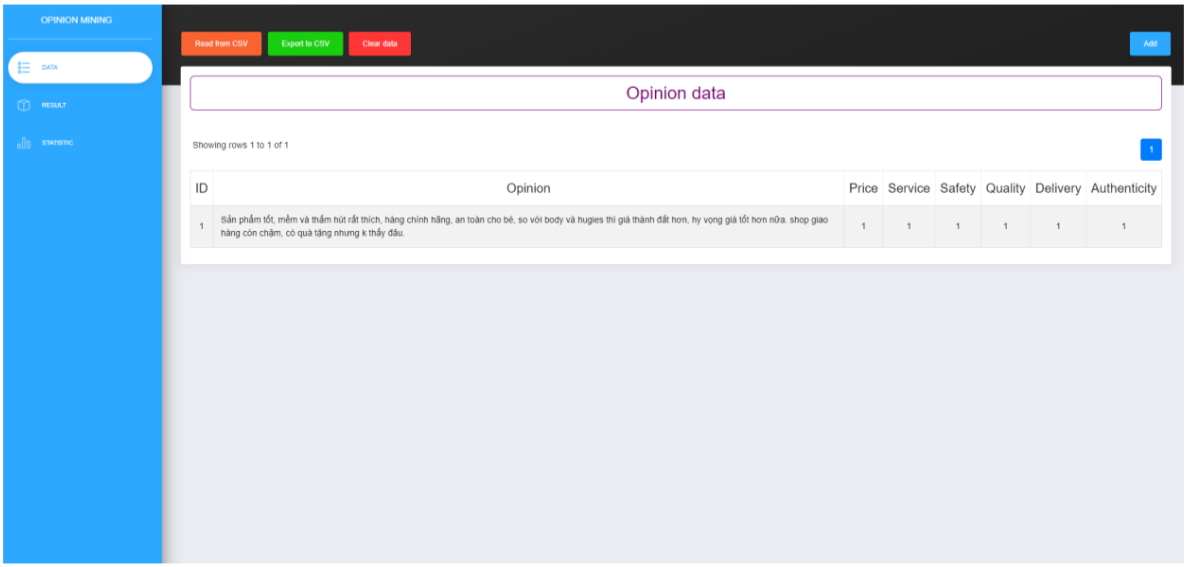

Figure 5.3: Insert comment by the 'Add' button

The model runs when the user selects 'Result' button. The results as illustrated in figure 5.5 and 5.6 show -1 and 1 on the aspects that appear in the comment corresponding to the negative and positive emotions related to that aspect.

In addition, we have built a few functions for the web application such as showing detailed statistical results of positive and negative sentiment on each aspects, exporting results in CSV, PNG, etc. formats for data statistics and visualization.

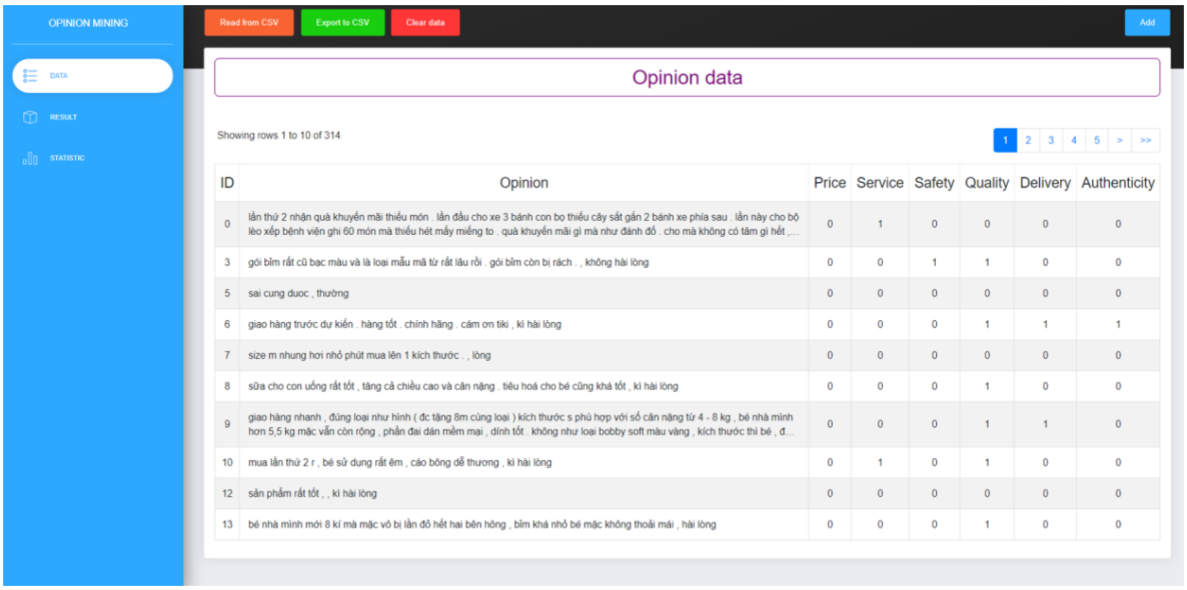

Figure 5.4: Insert comments by importing CSV file

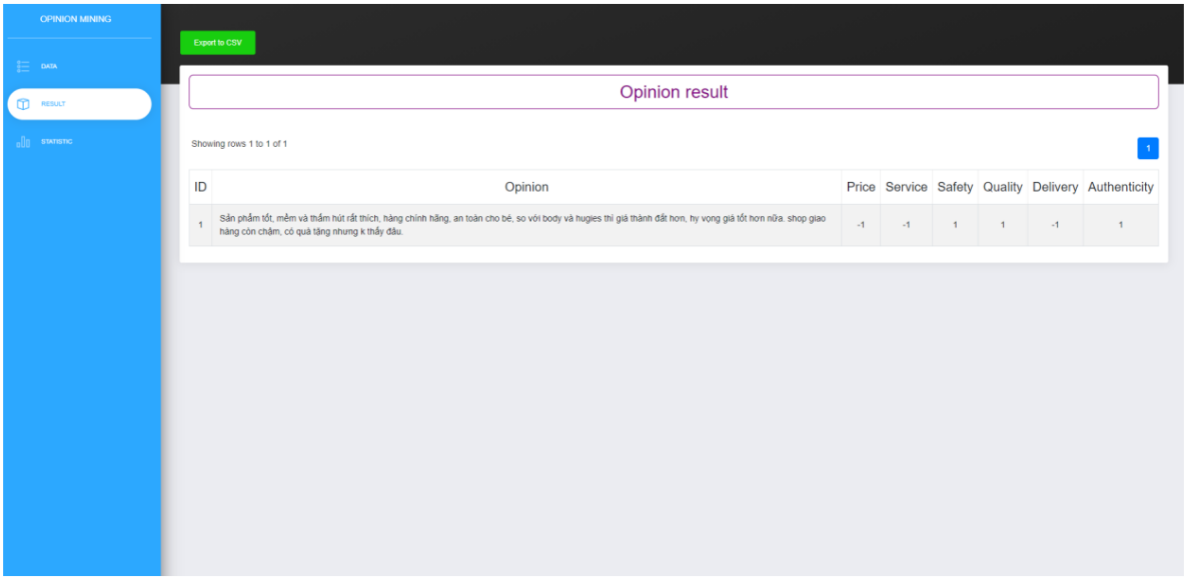

Figure 5.5: Model output when using ’Add" function

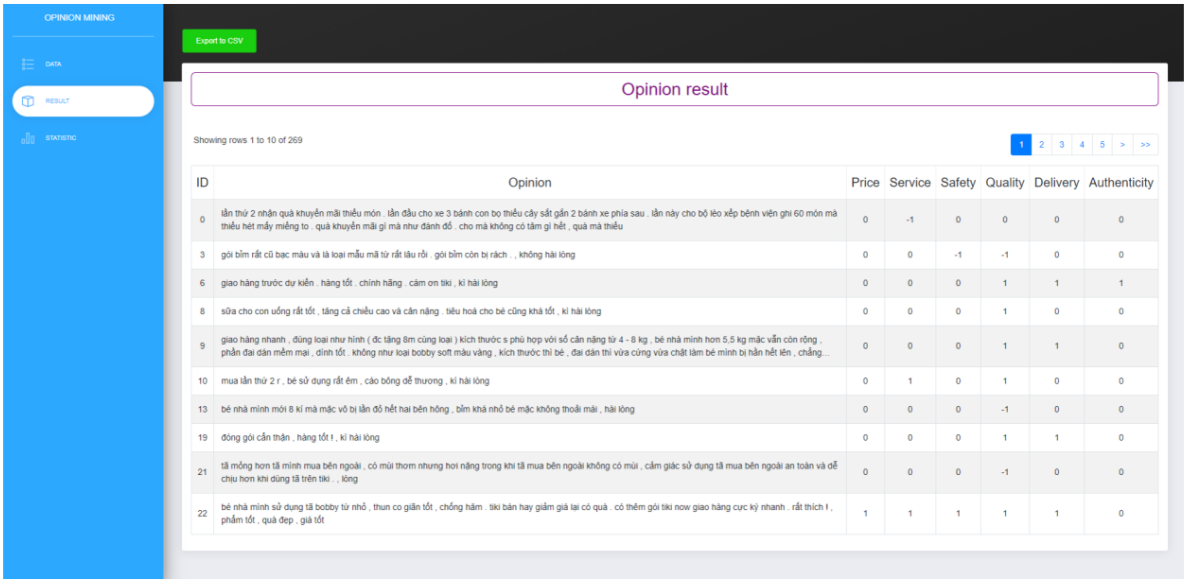

Figure 5.6: Model output when using ’Import" function

Figure 5.7: Detailed statistical results of positive and negative sentiment on each aspects

Chapter 6

Conclusions

Contributions

In this article, we proposed multiple approaches for the sentiment polarity detection - a sub-task of ABSA problem. Experiments indicated significant result as WWL combine with classic models has reach the maximum percentage of 95% in Micro- and Macro-Average Performance of F1-score.

Through experiments using the locating method and apply multiple data representation techniques, we can determine that using the WWL method with the Chi-Squared data representation brings very prominent results when combined with traditional machine learning methods. Because when classify the sentiment for multi-label data (i.e., multiple aspects in a data sample) using multiple models, we need to find the words associated with the label and the word representing its sentiment. WWL technique step solves this problem. In addition, in order for the data to focus on important words, we use Chi-Squared score to weight word level in the vocab and use it to represent data. This combination brings the most outstanding advantage to address the problem.

Limitations and Future Work

Selecting threshold for locating words inappropriately may be harmful to the result of the model. Window size (i.e., the number of words around from the center, which equals 3 in the proposed work) affects final result too. This requires manual data revision to be done carefully to choose which one is the best. This is the limitation of our method.

The weak point of proposed work indicates our next path to have in-depth research

in the future:

(i) More data should be collected to enlarge the dataset.

(ii) Handling the data imbalance problem.

(iii) Developing WWL in order to overcome current disadvantage.

References

[1] N. Q. Dat and N. T. Anh, "PhoBERT: Pre-trained language models for Vietnamese," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020, pp. 1037–1042.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] T. Vu, D. Q. Nguyen, D. Q. Nguyen, M. Dras, and M. Johnson, "VnCoreNLP: A Vietnamese natural language processing toolkit," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 56–60. [Online]. Available: https://www.aclweb.org/anthology/N18-5012