ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

BÁO CÁO ĐỒ ÁN TỔNG HỢP

"Nghiên cứu và so sánh phương pháp Weak Supervision và Supervised Learning trong phân tích cảm xúc lĩnh vực Gaming"

GVHD: Tiến sĩ Nguyễn An Khương

Sinh viên: Nguyễn Thành Công - 2310373
Lê Thúy Hiền - 2310990

THÀNH PHỐ HỒ CHÍ MINH, THÁNG 11 - 2025

Mục lục

Danh sách hình ảnh

Danh sách bảng

Chương 1

TỔNG QUAN

1.1. Đặt vấn đề

Trong kỷ nguyên số hóa hiện nay, ngành công nghiệp trò chơi điện tử (gaming) đang chứng kiến sự phát triển bùng nổ với quy mô thị trường toàn cầu đạt hàng trăm tỷ USD. Cùng với sự phát triển này là lượng nội dung đánh giá, thảo luận về game trên các nền tảng mạng xã hội tăng theo cấp số nhân. Các nền tảng như **Reddit** (với các subreddit gaming lớn như r/gaming, r/Games, r/pcgaming), **Steam**, **Metacritic**, và **Twitter** mỗi ngày tiếp nhận hàng triệu bài viết, bình luận từ cộng đồng người chơi.

Việc phân tích cảm xúc (sentiment analysis) từ khối lượng dữ liệu khổng lồ này mang lại giá trị to lớn cho nhiều đối tượng: các nhà phát triển game có thể nắm bắt phản hồi cộng đồng để cải thiện sản phẩm, nhà phân phối có thể đánh giá xu hướng thị trường, và người tiêu dùng có thể tham khảo đánh giá trước khi mua game. Tuy nhiên, việc áp dụng các phương pháp sentiment analysis truyền thống vào lĩnh vực gaming gặp phải ba thách thức đặc thù:

1.1.1. Thách thức 1: Đặc trưng ngôn ngữ gaming phức tạp

Ngôn ngữ được sử dụng trong cộng đồng game thủ mang những đặc điểm chuyên biệt, khác xa với ngôn ngữ phổ thông, tạo ra rào cản lớn cho các phương pháp phân tích cảm xúc truyền thống. Đầu tiên là sự xuất hiện dày đặc của các thuật ngữ kỹ thuật (technical jargon) như **"FPS drop"**, **"input lag"** hay **"hitbox"**; đòi hỏi mô hình phải hiểu được ngữ nghĩa chuyên ngành thay vì nghĩa đen thông thường. Bên cạnh đó, hệ thống tiếng lóng (slang) và từ viết tắt như **"P2W"** (pay-to-win) hay **"GOTY"** (Game of the Year) phát triển mạnh mẽ và thay đổi liên tục theo xu hướng. Khó khăn hơn cả là tính đa nghĩa theo ngữ cảnh (contextual polysemy) và sự phổ biến của lối nói châm biếm (sarcasm). Một từ ngữ mang sắc thái tích cực như **"masterpiece"** hoặc **"optimized"** hoàn toàn có thể được sử dụng để biểu đạt sự mỉa mai tiêu cực khi đi kèm với các bối cảnh lỗi game, khiến việc xác định đúng cảm xúc trở nên vô cùng phức tạp.

1.1.2. Thách thức 2: Chi phí gán nhãn thủ công cao

Việc xây dựng bộ dữ liệu huấn luyện chất lượng cao cho mô hình học có giám sát (Supervised Learning) đối mặt với bài toán nan giải về chi phí và khả năng mở rộng. Quá trình gán nhãn thủ công không chỉ đòi hỏi nguồn lực tài chính lớn để thuê chuyên gia có kiến thức nền tảng về game, mà còn tiêu tốn lượng thời gian đáng kể để xử lý hàng chục nghìn bài viết. Hơn nữa, tính nhất quán của dữ liệu thường không được đảm bảo do sự khác biệt trong quan điểm chủ quan của từng người gán nhãn (annotator), đặc biệt đối với các nội dung mang tính subjective. Đặc biệt, với tốc độ sinh dữ liệu khổng lồ hàng ngày trên các nền tảng mạng xã hội, phương pháp thủ công trở nên bất khả thi trong việc đáp ứng nhu cầu cập nhật dữ liệu liên tục ở quy mô lớn.

1.1.3. Thách thức 3: Mất cân bằng dữ liệu tự nhiên

Dữ liệu thực tế trong lĩnh vực gaming thường tồn tại sự mất cân bằng phân phối nghiêm trọng (class imbalance). Do tâm lý hành vi người dùng thường có xu hướng chia sẻ nhiều hơn khi hài lòng, dữ liệu gaming thường gặp hiện tượng "positive bias" với nhãn **Positive** chiếm đa số (thường dao động 40-50%), trong khi các đánh giá **Negative** (Tiêu cực)—nguồn thông tin quan trọng nhất cho nhà phát triển—lại chỉ chiếm tỷ lệ thiểu số (15-20%). Ngoài ra, lớp **Neutral** mang tính chất thảo luận hoặc hỏi đáp thường xuyên bị chồng lấn (overlapping) ranh giới với hai lớp còn lại. Sự mất cân bằng này khiến các mô hình học máy truyền thống dễ bị thiên kiến (bias) về phía lớp đa số để tối ưu hóa hàm mất mát, dẫn đến việc bỏ qua hoặc nhận diện kém các đặc trưng quan trọng của lớp thiểu số [1].

1.2. Mục tiêu nghiên cứu

Nghiên cứu này đặt ra ba mục tiêu chính nhằm giải quyết các thách thức đã nêu:

1.2.1. Mục tiêu 1: Xây dựng hệ thống Weak Supervision cho Gaming Domain

Mục tiêu đầu tiên là phát triển một hệ thống tự động gán nhãn cảm xúc cho nội dung gaming trên Reddit, tận dụng các tín hiệu cộng đồng (community signals) để thay thế quy trình gán nhãn thủ công tốn kém. Quá trình này bao gồm ba bước chính:

Đầu tiên, **thu thập dữ liệu** được thực hiện bằng cách crawl bài viết từ 6 subreddit gaming lớn bao gồm: r/gaming, r/Games, r/pcgaming, r/gamernews, r/gamedev, và r/indiegaming.

Tiếp theo, đề tài **thiết kế chiến lược 8-Signal**, kết hợp 8 loại tín hiệu với trọng số khác nhau để suy luận nhãn: **Awards** (trọng số 4.0 - tín hiệu mạnh nhất), **Comments Count** (3.0 - mức độ engagement), **Upvote Ratio** (2.5 - sự đồng thuận), **Post Score** (2.0 - độ phổ biến), **Gaming Text Features** (1.8-3.0 - từ vựng đặc thù), **Sarcasm Detection** (cơ chế đảo chiều sentiment), **Flair Analysis** (2.0-3.5 - phân loại từ nhãn bài viết), và **Top Comments** (2.0-2.5 - phân tích phản hồi cộng đồng).

Cuối cùng, hệ thống sử dụng **Weighted Voting Mechanism** để tổng hợp các tín hiệu trên với trọng số động (dynamic weighting từ 1.8 đến 3.5) kèm theo điểm tin cậy (confidence scoring) nhằm tạo ra các nhãn yếu (weak labels) chất lượng cao. Hiệu quả của phương pháp sẽ được

đánh giá thông qua việc so sánh Accuracy, F1-score và thời gian huấn luyện với phương pháp Supervised Learning truyền thống.

1.2.2. Mục tiêu 2: Giải quyết bài toán Class Imbalance

Để khắc phục hiện tượng mất cân bằng dữ liệu nghiêm trọng trong lĩnh vực gaming, đề tài nghiên cứu và so sánh hai nhóm phương pháp chính:

1.2.2.1. Phương pháp Algorithm-Level (Focal Loss và Class Weighting)

Đối với hướng tiếp cận ở mức thuật toán, đề tài tập trung vào Focal Loss, một kỹ thuật giúp mô hình tập trung vào các mẫu khó phân loại. Công thức toán học được định nghĩa như sau:

$$FL(p_t) = -\alpha_t(1-p_t)^\gamma \log(p_t) \tag{1.1}$$

Trong đó, p_t là xác suất dự đoán đúng của mẫu. Tham số α đóng vai trò trọng số cân bằng lớp, trong khi γ là tham số tập trung (focusing parameter) giúp giảm trọng số của các mẫu dễ (easy examples) thông qua hệ số điều chỉnh $(1-p_t)^\gamma$.

Bên cạnh đó, phương pháp Class Weighting cũng được áp dụng để tích hợp trọng số vào hàm CrossEntropyLoss, trong đó lớp thiểu số sẽ nhận trọng số cao hơn dựa trên công thức:

$$w_j = \frac{N}{K \cdot n_j} \tag{1.2}$$

Với N là tổng số mẫu, $K = 3$ là số lượng lớp, và n_j là số mẫu của lớp j. Ưu điểm của phương pháp này là đơn giản hơn Focal Loss do không cần tinh chỉnh các siêu tham số α, γ.

1.2.2.2. Phương pháp Data-Level (Undersampling)

Ở mức dữ liệu, phương pháp **Undersampling** được sử dụng làm cơ sở so sánh (baseline). Kỹ thuật này giảm số lượng mẫu của lớp đa số xuống mức cân bằng với lớp thiểu số (từ 21,821 xuống 12,093 mẫu). Mặc dù đảm bảo sự cân bằng hoàn toàn, phương pháp này phải đánh đổi bằng việc mất đi một lượng lớn dữ liệu huấn luyện (∼9,728 mẫu).

1.2.3. Mục tiêu 3: So sánh định lượng các phương pháp

Mục tiêu cuối cùng là xây dựng một pipeline gồm 4 giai đoạn (Stages) để thực hiện so sánh toàn diện giữa Weak Supervision và các chiến lược Supervised Learning:

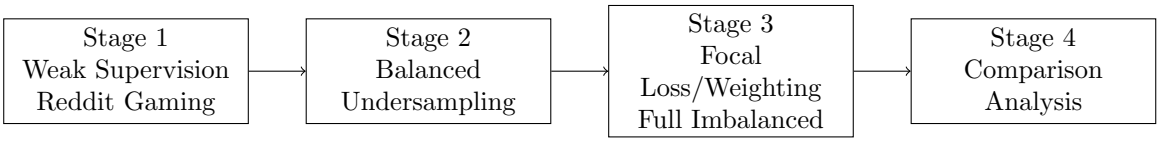

Figure 1.1: Pipeline 4 stages so sánh Weak Supervision và Supervised Learning

Các phương pháp sẽ được đánh giá đa chiều dựa trên 5 tiêu chí: (1) **Accuracy & F1-Score** (hiệu suất phân loại), (2) **Training Time** (tốc độ huấn luyện), (3) **Dataset Efficiency** (mức độ tận dụng dữ liệu), (4) **Labeling Cost** (chi phí gán nhãn), và (5) **Per-Class Performance** (hiệu quả trên từng lớp cụ thể: Negative, Neutral, Positive).

1.3. Đối tượng và phạm vi nghiên cứu

1.3.1. Nguồn dữ liệu nghiên cứu

1.3.1.1. Dữ liệu Weak Supervision (Stage 1)

Để xây dựng bộ dữ liệu huấn luyện cho mô hình Weak Supervision, nghiên cứu tập trung thu thập dữ liệu từ **Reddit** - nền tảng thảo luận lớn nhất thế giới về gaming. Cụ thể, dữ liệu được crawl từ 6 subreddit tiêu biểu bao gồm: **r/gaming** (∼35M thành viên), **r/Games** (∼3M thành viên - tập trung vào thảo luận chất lượng cao), **r/pcgaming** (∼2.5M thành viên), **r/gamernews**, **r/gamedev**, và **r/indiegaming**.

Các subreddit này được lựa chọn dựa trên các tiêu chí khắt khe nhằm đảm bảo chất lượng dữ liệu: số lượng thành viên đông đảo (>100K), tần suất đăng bài cao (>100 bài/ngày), tỷ lệ spam thấp (<5%) và mức độ tương tác tốt (tỷ lệ upvote trung bình >0.7). Nội dung thảo luận bao phủ đa dạng các chủ đề từ tin tức, review, phát triển game đến trải nghiệm người chơi.

Quá trình thu thập được thực hiện thông qua thư viện **PRAW** (Python Reddit API Wrapper), giới hạn trong các bài đăng thuộc tháng 11/2025. Các từ khóa tìm kiếm (keywords) được sử dụng bao gồm: "game", "gaming", "gameplay", "review", "experience", "bug", "issue", "performance", "story", và "fun".

1.3.1.2. Dữ liệu Supervised Learning (Stage 2, 3)

Đối với giai đoạn huấn luyện có giám sát, nghiên cứu sử dụng bộ dữ liệu chuẩn **"Reddit Gaming Comments with Sentiments"** từ Kaggle[1]. Bộ dữ liệu này bao gồm **21,821** bình luận game (game reviews) được tổng hợp từ Steam và Reddit, đã qua quá trình gán nhãn thủ công.

Một đặc điểm quan trọng của bộ dữ liệu này là sự mất cân bằng tự nhiên (Natural Imbalance) trong phân phối nhãn: lớp **Positive** chiếm đa số (∼44.8%), theo sau là lớp **Neutral** (∼35.7%), trong khi lớp **Negative** chỉ chiếm tỷ lệ thiểu số (∼18.5%), tạo ra tỷ lệ mất cân bằng khoảng 2.43:1 giữa lớp Positive và Negative. Dữ liệu bao gồm hai trường thông tin chính là nội dung văn bản (`comment`) và nhãn cảm xúc (`sentiment`).

[1] https://www.kaggle.com/datasets/sainitishmitta04/23k-reddit-gaming-comments-with-sentiments-dataset

Quy trình xử lý dữ liệu được chia làm hai hướng phục vụ cho các giai đoạn khác nhau: Tại **Stage 2**, kỹ thuật Undersampling được áp dụng để cân bằng số lượng mẫu, giảm kích thước dữ liệu xuống còn **12,093** mẫu. Ngược lại, ở **Stage 3**, toàn bộ **21,821** mẫu dữ liệu gốc được giữ nguyên để nghiên cứu hiệu quả của các phương pháp xử lý mất cân bằng dữ liệu.

1.3.2. Mô hình nghiên cứu

Nghiên cứu lựa chọn mô hình nền tảng (Base Model) là `cardiffnlp/twitter-roberta-base-sentiment-latest` [2]. Quyết định này dựa trên bốn ưu điểm vượt trội của kiến trúc RoBERTa-Twitter:

Thứ nhất, mô hình được **tối ưu hóa cho văn bản mạng xã hội** nhờ quá trình pre-training trên 124 triệu tweets. Điều này giúp mô hình xử lý hiệu quả các đặc trưng ngôn ngữ không chính thức như emoji, tiếng lóng (slang), và các ngữ cảnh ngắn (<512 tokens) thường gặp trong bình luận game.

Thứ hai, mô hình sở hữu **kiến trúc mạnh mẽ** với 12 lớp Transformer, 125 triệu tham số và bộ từ vựng (vocabulary) lên đến 50K tokens, bao quát tốt các thuật ngữ chuyên ngành gaming.

Thứ ba, RoBERTa-Twitter đã chứng minh **hiệu năng vượt trội** khi đạt kết quả SOTA (State-of-the-Art) trên bộ dữ liệu SemEval-2017 Sentiment Analysis và độ chính xác xấp xỉ 92% trên các tác vụ phân tích cảm xúc Twitter, cho thấy khả năng tổng quát hóa tốt sang miền dữ liệu gaming.

Cuối cùng, khả năng **Transfer Learning hiệu quả** cho phép tinh chỉnh (fine-tune) mô hình nhanh chóng mà không cần huấn luyện lại từ đầu, đồng thời giảm thiểu hiện tượng overfitting nhờ vào nền tảng pre-training vững chắc.

1.3.3. Phạm vi bài toán

1.3.3.1. Loại bài toán và Output Classes

Đề tài tập trung giải quyết bài toán **Multi-class Classification** với 3 lớp cảm xúc đầu ra:

- **Positive (Tích cực):** Bao gồm các đánh giá tốt, lời khen ngợi hoặc khuyến nghị chơi game. Ví dụ: "This game is a masterpiece!", "Best GOTY contender".
- **Neutral (Trung lập):** Bao gồm các thảo luận khách quan, câu hỏi hoặc chia sẻ thông tin thuần túy. Ví dụ: "What's your opinion on this game?", "Graphics are good but story is meh".
- **Negative (Tiêu cực):** Bao gồm các phản hồi tiêu cực, báo cáo lỗi (bug report) hoặc phàn nàn gay gắt (rant). Ví dụ: "Broken mess, avoid this!", "P2W garbage".

1.3.3.2. Metrics đánh giá và Giới hạn phạm vi

Để đánh giá toàn diện hiệu quả của mô hình, nghiên cứu sử dụng tập hợp các chỉ số đo lường bao gồm: **Accuracy** (tỷ lệ phân loại đúng tổng thể), **F1-Score Weighted** (trọng số theo kích thước lớp), **F1-Score Macro** (trung bình không trọng số), cùng với **Precision/Recall** cho từng lớp và **Confusion Matrix** để phân tích chi tiết các trường hợp phân loại sai. Bên cạnh đó, **Training Time** cũng được ghi nhận để so sánh chi phí tính toán.

Nghiên cứu giới hạn phạm vi trong phân tích cảm xúc dựa trên văn bản (Text-based sentiment) từ các nguồn reviews và comments. Các hướng tiếp cận đa phương thức (Multi-modal) kết hợp hình ảnh, phân tích cảm xúc theo khía cạnh (Aspect-based), hay phân tích cảm xúc chi tiết (Emotion analysis như vui, giận, sợ hãi...) nằm ngoài phạm vi của đề tài này.

1.4. Đóng góp của đề tài

Nghiên cứu mang lại những đóng góp quan trọng cho lĩnh vực Phân tích cảm xúc trong miền dữ liệu Gaming, cụ thể trên các phương diện sau:

1.4.1. Đóng góp 1: Chiến lược 8-Signal Weak Supervision cho Gaming

Tính mới: Đề tài đề xuất hệ thống gán nhãn yếu (weak labeling) đầu tiên được tối ưu hóa riêng cho miền dữ liệu gaming, kết hợp 8 loại tín hiệu cộng đồng với cơ chế trọng số động [3]. Hệ thống tín hiệu được phân thành ba nhóm chính:

Đầu tiên là nhóm **tín hiệu siêu dữ liệu (Metadata Signals)**, đóng vai trò chỉ báo định lượng mạnh mẽ. Trong đó, **Awards** (Gold, Platinum...) được xem là tín hiệu cao nhất với trọng số 4.0 do gắn liền với chi phí thực tế. Các chỉ số khác bao gồm **Comments Count** (trọng số 3.0) phản ánh mức độ tương tác, **Upvote Ratio** (trọng số 2.5) thể hiện sự đồng thuận, và **Post Score** (trọng số 2.0) đại diện cho độ phổ biến của bài viết.

Tiếp theo là nhóm **tín hiệu nội dung (Content Signals)**, tập trung khai thác đặc trưng văn bản. Hệ thống phân tích **Gaming Text Features** (trọng số 1.8-3.0) dựa trên các từ khóa chuyên ngành (ví dụ: "masterpiece" là tích cực, "broken mess" là tiêu cực). Đặc biệt, cơ chế **Sarcasm Detection** được tích hợp để xử lý các trường hợp châm biếm (như tag "/s" hoặc ngữ cảnh mâu thuẫn), cho phép đảo chiều nhãn cảm xúc chính xác.

Cuối cùng là nhóm **tín hiệu ngữ cảnh (Context Signals)**, bao gồm **Flair Analysis** (trọng số 2.0-3.5) để tận dụng nhãn phân loại có sẵn của bài viết (ví dụ: flair "Bug" mang nghĩa tiêu cực), và **Top Comments Sentiment** (trọng số 2.0-2.5) nhằm phân tích phản hồi từ cộng đồng thông qua 3 bình luận nổi bật nhất.

Các tín hiệu trên được tổng hợp thông qua cơ chế Weighted Voting Mechanism để tính toán độ tin cậy (Confidence):

$$\text{Confidence} = \frac{\sum_{i=1}^{8} w_i \cdot \mathbb{1}(\text{signal}_i = \text{label})}{\sum_{i=1}^{8} w_i} \tag{1.3}$$

Trong đó w_i là trọng số của tín hiệu thứ i và hàm chỉ thị $\mathbb{1}$ nhận giá trị 1 nếu tín hiệu đồng

thuận với nhãn dự đoán. Nhãn cuối cùng chỉ được chấp nhận nếu độ tin cậy vượt ngưỡng 0.6.

1.4.2. Đóng góp 2: Phân tích sâu Class Imbalance với Focal Loss và Class Weighting

Tính mới: Đề tài thực hiện nghiên cứu chuyên sâu và so sánh hai phương pháp algorithm-level để giải quyết vấn đề mất cân bằng dữ liệu trong phân tích cảm xúc gaming: Focal Loss và Class Weighting.
Về mặt kỹ thuật, Focal Loss được hiện thực hóa dựa trên công thức:

$$FL(p_t) = -\alpha_t(1-p_t)^\gamma \log(p_t) \tag{1.4}$$

Quy trình tính toán bắt đầu bằng việc chuyển đổi logits thành xác suất qua hàm softmax, sau đó xác định xác suất dự đoán đúng p_t cho nhãn thực tế. Trọng số focal (focal weight) được tính toán bằng $(1-p_t)^\gamma$ và áp dụng vào hàm loss Cross Entropy tiêu chuẩn. Các siêu tham số được tinh chỉnh cụ thể: **Alpha (α) = 0.25** giúp cân bằng trọng số giữa lớp Positive và Negative, trong khi **Gamma (γ) = 2.0** đóng vai trò tham số tập trung, giúp giảm trọng số của các mẫu dễ phân loại và buộc mô hình học kỹ hơn các mẫu khó.
Song song với Focal Loss, đề tài cũng triển khai phương pháp Class Weighting sử dụng trọng số nghịch đảo logarit:

$$w_j = \frac{1}{\ln(1+n_j)}, \quad \text{chuẩn hóa: } w'_j = \frac{w_j}{\sum_{k=1}^{K} w_k} \tag{1.5}$$

Trọng số này được tích hợp trực tiếp vào Weighted Cross-Entropy Loss, trong đó lớp thiểu số nhận trọng số cao hơn để cân bằng đóng góp vào gradient. Ưu điểm của Class Weighting là tính đơn giản trong triển khai (không cần tinh chỉnh α, γ) và ổn định trong quá trình huấn luyện nhờ logarit giảm chênh lệch trọng số quá lớn.

1.4.3. Đóng góp 3: Gaming-Specific Lexicon

Tính mới: Đề tài xây dựng và chuẩn hóa một bộ từ điển cảm xúc chuyên biệt cho lĩnh vực gaming, được phân cấp thành 3 tầng từ vựng (tiers) dựa trên mức độ tác động đến cảm xúc:

Table 1.1: Gaming-Specific Lexicon với 3 tiers

Tier	**Weight**	**Examples**
Strong Indicators	3.0	masterpiece, GOTY, unplayable, broken mess
N-grams	2.5	worth the price, waste of money, cash grab
Single Keywords	1.8	amazing, fun, buggy, p2w, grindy

1.4.4. Đóng góp 4: Reproducible Research Package

Tính mới: Nhằm đảm bảo tính minh bạch khoa học và khả năng tái lập kết quả, đề tài cung cấp trọn bộ tài nguyên nghiên cứu (artifacts) bao gồm: (1) **Code Package** với 5 Jupyter Notebooks tài liệu hóa đầy đủ quy trình từ stage 1 đến stage 4; (2) **Processed Datasets** chứa dữ liệu Reddit đã gán nhãn yếu và dữ liệu Kaggle sạch; (3) **Trained Models** là các checkpoint của mô hình RoBERTa đã được fine-tune; và (4) **Results JSON** lưu trữ kết quả thực nghiệm định dạng chuẩn.

Toàn bộ mã nguồn và dữ liệu được công khai tại GitHub:
https://github.com/ThanhCongNguyen-2310373/Sentiment_Analysis_Demo

1.5. Cấu trúc báo cáo

Báo cáo được tổ chức thành 6 chương với luồng logic từ lý thuyết đến thực nghiệm:

CHƯƠNG 1: TỔNG QUAN
Trình bày bối cảnh, ba thách thức chính và mục tiêu nghiên cứu, các đóng góp cốt lõi về chiến lược gán nhãn yếu tự động và các giải pháp xử lý mất cân bằng cho dữ liệu gaming.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT
Hệ thống hóa lý thuyết về phân tích cảm xúc trong game, mô hình RoBERTa và các phương pháp Weak Supervision. Trình bày cơ sở toán học của Focal Loss và Class Weighting làm nền tảng giải quyết bài toán mất cân bằng dữ liệu.

CHƯƠNG 3: PHƯƠNG PHÁP ĐỀ XUẤT
Mô tả Pipeline 4 giai đoạn, chiến lược sinh nhãn yếu "8-Signal" từ dữ liệu Reddit, kỹ thuật cân bằng dữ liệu Undersampling và thuật toán Focal Loss/Class Weighting để tối ưu hóa quá trình huấn luyện mô hình.

CHƯƠNG 4: THỰC NGHIỆM VÀ KẾT QUẢ
Thiết lập môi trường và kết quả thực nghiệm chi tiết từng giai đoạn. Tổng hợp so sánh định lượng đa chiều về độ chính xác, F1-Score và tốc độ huấn luyện giữa các chiến lược thông qua biểu đồ và bảng số liệu.

CHƯƠNG 5: ĐÁNH GIÁ VÀ THẢO LUẬN
Thảo luận về giới hạn Domain Shift của phương pháp Weak Supervision. Phân tích sự đánh đổi kỹ thuật giữa các chiến lược Supervised và lý giải nguyên nhân Focal Loss đạt hiệu quả tối ưu nhất về độ ổn định và chính xác.

CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN
Tổng kết hiệu quả của pipeline đề xuất, khẳng định vai trò của Focal Loss trong bài toán dữ liệu lệch. Đề xuất hướng mở rộng nghiên cứu sử dụng LLM Teacher hoặc Active Learning để nâng cao chất lượng hệ thống.

Chương 2

CƠ SỞ LÝ THUYẾT

2.1. Sentiment Analysis trong Gaming

2.1.1. Khái niệm Sentiment Analysis

Sentiment Analysis (Phân tích cảm xúc), hay còn gọi là Opinion Mining, là một nhánh của Natural Language Processing (NLP) nhằm xác định thái độ, quan điểm, hoặc cảm xúc của người viết đối với một chủ đề, sản phẩm, hoặc sự kiện cụ thể [4].

2.1.1.1. Định nghĩa hình thức

Cho một văn bản đầu vào $x = (w_1, w_2, ..., w_n)$ với w_i là các từ trong câu, bài toán sentiment analysis là tìm hàm ánh xạ:

$$f : \mathcal{X} \to \mathcal{Y} \tag{2.1}$$

Trong đó, $\mathcal{X}$ là không gian văn bản đầu vào, và $\mathcal{Y}$ là tập nhãn cảm xúc với $\mathcal{Y} = \{\text{Positive}, \text{Neutral}, \text{Negative}\}$ cho bài toán 3 classes.

Với deep learning, hàm f được parameterize bởi neural network với trọng số θ:

$$f_\theta(x) = \arg\max_{y \in \mathcal{Y}} P(y|x; \theta) \tag{2.2}$$

Mục tiêu huấn luyện là tìm θ^* minimize loss function:

$$\theta^* = \arg\min_\theta \mathcal{L}(\theta) = \arg\min_\theta \sum_{i=1}^{N} \ell(f_\theta(x_i), y_i) \tag{2.3}$$

với ℓ là loss function (Cross Entropy, Focal Loss, etc.), N là số mẫu training.

2.1.1.2. Ba cấp độ Sentiment Analysis

1. **Document-level:** Phân loại toàn bộ văn bản (review, article)
2. **Sentence-level:** Phân tích từng câu riêng lẻ
3. **Aspect-level:** Phân tích theo khía cạnh cụ thể

Đề tài này: Sentence-level sentiment classification trên gaming reviews/comments (3-class).

2.1.2. Đặc điểm ngôn ngữ Gaming

Ngôn ngữ gaming có những đặc trưng riêng biệt so với văn bản thông thường, đòi hỏi xử lý đặc biệt:

2.1.2.1. Thuật ngữ kỹ thuật chuyên ngành (Technical Jargon)

Gaming sử dụng rất nhiều thuật ngữ kỹ thuật chỉ có trong lĩnh vực này:

Table 2.1: Ví dụ gaming technical terms với sentiment context

Term	**Nghĩa**	**Sentiment Context**
FPS drop	Giảm khung hình	Negative (performance issue)
Input lag	Độ trễ nhập liệu	Negative (responsiveness)
Hitbox	Vùng va chạm	Neutral/Negative (nếu "bad hitbox")
Nerf	Giảm sức mạnh	Negative (nếu favorite character)
Buff	Tăng sức mạnh	Positive (nếu benefit player)
Meta	Strategy dominant	Neutral (descriptive)
Grinding	Chơi lặp lại kiếm item	Negative (tedious)
RNG	Random number generator	Negative (luck-based)

Thách thức: General-purpose sentiment models (trained trên movie/product reviews) không hiểu các thuật ngữ này. Ví dụ: "This game has terrible FPS drops" → Model cần hiểu "FPS drops" = negative indicator.

Giải pháp trong đề tài: Đề tài giải quyết vấn đề này bằng cách sử dụng RoBERTa-Twitter đã được pre-trained trên social media để xử lý tốt hơn slang và jargon, bổ sung gaming-specific lexicon trong Weak Supervision (Signal 5), và fine-tune trên gaming data để model học gaming vocabulary đặc thù.

2.1.2.2. Slang và từ viết tắt đặc thù (Gaming Slang)

Cộng đồng gaming phát triển ngôn ngữ slang riêng, thay đổi nhanh theo xu hướng:

Table 2.2: Gaming slang với sentiment implications

Slang	**Nghĩa đầy đủ**	**Sentiment**
P2W	Pay-to-win (trả tiền thắng)	Strongly Negative
GOTY	Game of the Year	Strongly Positive
GG EZ	Good game, easy	Positive/Toxic context
OP	Overpowered (quá mạnh)	Mixed (positive if fun)
Trash/dogshit	Extremely bad	Strongly Negative
Banger	Excellent game/music	Strongly Positive
Mid	Mediocre, average	Neutral/Slightly Negative
Bussin	Very good	Positive

Ví dụ phức tạp: Câu "This P2W garbage is a cash grab" thể hiện Strongly Negative do sự kết hợp của "P2W" (negative), "garbage" (negative) và "cash grab" (negative), yêu cầu model phải học được cụm từ "cash grab" như một negative gaming term. Ngược lại, "GOTY material, absolute banger" mang sentiment Strongly Positive với sự kết hợp của "GOTY" (positive), "banger" (positive) và "absolute" đóng vai trò như amplifier. Trong khi đó, "Game is mid, nothing special" thể hiện Neutral/Slightly Negative, với "mid" là Gen Z slang có nghĩa mediocre, không phải positive nhưng cũng không phải strongly negative.

2.1.2.3. Đa nghĩa theo ngữ cảnh (Polysemy)

Hiện tượng đa nghĩa là một thách thức đặc thù trong ngôn ngữ gaming, nơi cùng một từ vựng có thể mang sắc thái cảm xúc hoàn toàn trái ngược tùy thuộc vào ngữ cảnh sử dụng. Điển hình là thuật ngữ "broken"; trong khi ngữ cảnh kỹ thuật ("Game is broken"), từ này mang nghĩa tiêu cực chỉ sự hỏng hóc hay lỗi game, thì khi đề cập đến chiến thuật ("This combo is broken"), nó lại chuyển sang hàm ý tích cực, khen ngợi sức mạnh vượt trội (overpowered) của nhân vật. Tương tự, từ "sick" cũng biến đổi linh hoạt từ thái độ chán nản, khó chịu ("sick of this bug") sang sự thán phục, phấn khích cực độ trước một pha xử lý hay ("That play was sick!"). Ngoài ra, các thuật ngữ như "grinding" (cày cuốc) cũng mang tính lưỡng nghĩa, có thể bị coi là nhàm chán (tedious) hoặc thú vị tùy vào trải nghiệm người chơi. Chính sự phức tạp này đòi hỏi hệ thống phải sử dụng các mô hình kiến trúc Transformer như RoBERTa với cơ chế self-attention để nắm bắt chính xác ngữ cảnh toàn cục thay vì chỉ dựa vào ý nghĩa đơn lẻ của từ khóa.

2.1.2.4. Sarcasm và Irony (Châm biếm)

Gaming community thường sử dụng sarcasm để bày tỏ thất vọng hoặc châm chọc:

Marker rõ ràng: "/s" tag: Câu "Yeah, 60 FPS on a 4090, totally optimized /s" có literal sentiment là Positive (60 FPS, optimized), nhưng khi có /s tag thì trở thành Negative do sarcasm về optimization kém, với ý nghĩa thực sự là game chạy chỉ 60 FPS trên card đồ họa mạnh nhất chứng tỏ tối ưu tệ. Tương tự, "Oh great, another microtransaction /s" có literal sentiment Positive ("great") nhưng với /s tag biến thành Negative thể hiện sự phản đối microtransaction.

Implicit sarcasm (không có /s): Câu "Wow, crashing 10 times in an hour, what a masterpiece" chứa "masterpiece" (positive word) kết hợp với "crashing 10 times" (negative fact), sarcasm được implied và sentiment thực sự là Negative. Tương tự, "Thanks for the 20GB patch that fixed nothing" có "Thanks" (positive) nhưng đi kèm "fixed nothing" (negative), tạo thành ironic gratitude với sentiment Negative.

Xử lý trong đề tài: Đề tài sử dụng Weak Supervision (Signal 6) để detect "/s" tag và flip sentiment (positive ↔ negative), đồng thời detect các combinations như "great" + negative context để tính sarcasm probability. Tuy nhiên, một limitation là implicit sarcasm vẫn khó detect hoàn toàn do require human-like reasoning.

2.2. RoBERTa Model

2.2.1. Từ BERT đến RoBERTa

2.2.1.1. BERT (Bidirectional Encoder Representations from Transformers)

BERT [5] là mô hình Transformer-based được Google giới thiệu năm 2018, tạo ra bước ngoặt trong NLP:

Kiến trúc: BERT sử dụng encoder-only Transformer, chỉ dùng encoder stack mà không có decoder, với 12 layers cho phiên bản base và 24 layers cho large. Hidden size là 768 (base) hoặc 1024 (large), số attention heads là 12 (base) hoặc 16 (large), và tổng số parameters là 110M (base) hoặc 340M (large).

Pre-training Tasks:

1. **Masked Language Modeling (MLM):** Task này mask 15% tokens trong câu với [MASK], và mô hình dự đoán từ bị mask dựa vào context hai chiều. Ví dụ: "The game is [MASK]" → predict "amazing".
2. **Next Sentence Prediction (NSP):** Task này cho 2 câu A, B và predict liệu B có theo sau A không, với mục đích học sentence-level relationships.

Hạn chế của BERT: BERT có một số hạn chế như NSP task không thực sự hữu ích (RoBERTa sau này đã bỏ), static masking với mask pattern giống nhau mỗi epoch, batch size nhỏ (256) và train time ngắn, cùng với việc pre-trained chỉ trên BookCorpus và Wikipedia (formal text).

2.2.1.2. RoBERTa (Robustly Optimized BERT Pretraining Approach)

RoBERTa [2] (Facebook AI, 2019) là phiên bản tối ưu của BERT:

Cải tiến chính:

1. **Bỏ Next Sentence Prediction (NSP):** Research chứng minh NSP không giúp tăng performance, do đó RoBERTa chỉ giữ lại Masked Language Modeling (MLM).
2. **Dynamic Masking:** Khác với BERT sử dụng static mask (cùng pattern mỗi epoch), RoBERTa mask tokens khác nhau mỗi lần đưa vào model, giúp model thấy nhiều masking patterns hơn và generalize tốt hơn.
3. **Larger Batches & Longer Training:** BERT sử dụng batch size 256 và train 1M steps, trong khi RoBERTa tăng batch size lên 8K và train với nhiều data hơn, giúp đạt stable gradients và better convergence.
4. **Byte-Pair Encoding (BPE):** BERT sử dụng WordPiece với 30K vocab, trong khi RoBERTa sử dụng BPE với 50K vocab (character-level subwords), giúp handle rare words tốt hơn như gaming slang và typos.
5. **More Training Data:** BERT sử dụng 16GB text (BookCorpus + Wikipedia), trong khi RoBERTa tăng lên 160GB text bằng cách thêm CC-News, OpenWebText và Stories.

Kết quả: RoBERTa đạt SOTA trên GLUE, SQuAD, RACE benchmarks, vượt BERT 1-3% accuracy.

2.2.2. RoBERTa-Twitter Variant

Model sử dụng: `cardiffnlp/twitter-roberta-base-sentiment-latest`

2.2.2.1. Đặc điểm RoBERTa-Twitter

RoBERTa-Twitter base model có 125M parameters, được pre-trained trên **124 million tweets** (thay vì BookCorpus). Corpus này bao gồm informal language, slang, abbreviations, emoji, hashtags, @mentions, short-form text (<280 characters), và social media linguistic patterns.

Model này được fine-tuned cho Sentiment Analysis trên TweetEval benchmark với 3-class (Negative, Neutral, Positive), ∼60K labeled tweets, và F1-score: 0.92 (SOTA trên Twitter sentiment). Vocabulary gồm 50K BPE tokens bao gồm common emoji tokens, social media abbreviations (lol, omg, tbh, etc.), và gaming terms learned from tweets (gg, fps, noob, etc.).

2.2.2.2. Tại sao phù hợp với Gaming Sentiment?

Lợi ích:

Table 2.3: So sánh linguistic features: Twitter vs Gaming Comments

Feature	**Twitter**	**Gaming (Reddit/Steam)**
Average length	50-100 chars	100-300 chars
Informal language	✓	✓
Slang/abbreviations	✓	✓
Emoji usage	High	Medium
Sarcasm frequency	High	High
Technical jargon	Low	High (gaming-specific)
Sentiment expression	Direct	Direct + sarcasm

1. **Linguistic similarity:** Reddit/Steam comments tương tự tweets về style
2. **Informal language handling:** Trained on slang, không bị shock với "P2W", "GG EZ"
3. **Short-form optimization:** Gaming comments thường ngắn (<512 tokens)
4. **Sarcasm awareness:** Twitter có nhiều sarcasm, model đã học patterns
5. **Transfer learning efficient:** Pre-trained sentiment head, chỉ cần fine-tune

2.3. Weak Supervision

2.3.1. Định nghĩa và Motivation

Weak Supervision là paradigm học máy sử dụng các nguồn nhãn "yếu" (noisy, inexact, incomplete) thay vì nhãn thủ công chính xác để huấn luyện model.

2.3.1.1. Taxonomy of Supervision

Table 2.4: So sánh các loại supervision

Type	**Label Quality**	**Cost**	**Scale**	**Speed**
Supervised	Perfect	Very High	Small	Slow
Weak Supervision	Noisy	Low	Large	Fast
Semi-Supervised	Mixed	Medium	Medium	Medium
Unsupervised	None	Zero	Very Large	Fast

Trade-off chính: Label quality ↓, Cost ↓, Scale ↑

2.3.1.2. Nguồn gốc Weak Labels

1. **Crowdsourcing:** Nhãn từ non-expert workers (Amazon MTurk) với pros là cheap và scalable, nhưng cons là low quality, inconsistent, và need aggregation.
2. **Heuristic Rules:** Rules viết bởi domain experts, ví dụ gaming: "IF 'P2W' in text THEN Negative". Pros bao gồm high precision và interpretable, nhưng cons là low coverage và manual engineering.
3. **External Knowledge Bases:** Sentiment lexicons (VADER, SentiWordNet) có pros là general-purpose và pre-built, nhưng cons là domain mismatch (không có gaming terms).
4. **Community Signals:** Upvotes, awards, engagement (ĐỀ TÀI NÀY) với ví dụ high upvote ratio → Positive sentiment. Pros bao gồm automatic, free, scalable, gaming-specific, nhưng cons là noisy và indirect proxy for sentiment.

2.3.2. Snorkel Framework

Snorkel [3] là framework nổi tiếng cho weak supervision, phát triển bởi Stanford.

2.3.2.1. Kiến trúc Snorkel

Các bước chính:

1. **Define Labeling Functions (LFs):** LF là hàm $\lambda_i : \mathcal{X} \to \{-1, 0, 1, ..., k\}$ với -1 biểu thị Abstain (không vote) và $0, 1, ..., k$ là Class labels. Ví dụ: $\lambda_{\text{P2W}}(x) =$ Negative if "P2W" in x else Abstain.
2. **Apply LFs tạo Label Matrix:**
$$\Lambda \in \{-1, 0, 1, ..., k\}^{n \times m} \tag{2.4}$$
với n samples, m labeling functions
3. **Train Label Model (Probabilistic Graphical Model):** Bước này estimate accuracy của mỗi LF ($\alpha_i = P(\lambda_i \text{ correct})$), estimate correlations giữa LFs, và output probabilistic labels $\tilde{Y} = (\tilde{y}_1, ..., \tilde{y}_n)$.
4. **Train Discriminative Model:** Bước này sử dụng $\tilde{Y}$ (noisy labels) để train neural network, giúp model học generalize từ noisy data.

2.3.2.2. Label Model: Probabilistic Formulation

Giả sử có m labeling functions $\lambda_1, ..., \lambda_m$ và true label Y^*. Mô hình xác suất:

$$P(\Lambda, Y^*) = P(Y^*) \prod_{i=1}^{m} P(\lambda_i | Y^*) \tag{2.5}$$

Với assumptions bao gồm conditional independence $P(\lambda_i|Y^*, \lambda_j) = P(\lambda_i|Y^*)$ (given true label) và accuracy parameter $\alpha_i = P(\lambda_i = Y^*|Y^* \neq -1)$.

Label model estimate $P(Y^*|\Lambda)$ bằng maximum likelihood:

$$\theta^* = \arg\max_\theta \sum_{j=1}^{n} \log P(Y_j^*|\Lambda_j; \theta) \tag{2.6}$$

2.3.3. Weak Supervision trong Gaming (Đề tài này)

Đề tài này không sử dụng Snorkel trực tiếp, mà design custom weak supervision system tận dụng **Reddit community signals**.

2.3.3.1. Tại sao không dùng Snorkel?

Đề tài không dùng Snorkel vì Snorkel require viết nhiều LFs manually (heuristic rules), trong khi Reddit cung cấp sẵn rich metadata (upvotes, awards, comments) không cần rules phức tạp. Hơn nữa, custom voting mechanism phù hợp với gaming domain hơn general-purpose PGM.

2.3.3.2. 8-Signal Strategy (Weighted Voting)

Thay vì Snorkel's label model, đề tài dùng **weighted voting** với 8 signals:

$$\text{Label}(x) = \arg\max_{c \in \{\text{Pos, Neu, Neg}\}} \sum_{i=1}^{8} w_i \cdot \mathbb{1}(\text{Signal}_i(x) = c) \tag{2.7}$$

với constraints:

$$\text{Confidence}(x) = \frac{\max_c \sum_i w_i \cdot \mathbb{1}(\text{Signal}_i = c)}{\sum_i w_i} \geq 0.6 \tag{2.8}$$

Ưu điểm so với Snorkel: Phương pháp này simpler (không cần train PGM), faster (direct voting thay vì iterative optimization), interpretable (trọng số rõ ràng, có thể tune manually), và gaming-specific (signals thiết kế rieng cho Reddit gaming).

Nhược điểm: Phương pháp này không model correlations giữa signals, không tự động estimate signal accuracy, và trọng số cần tune manually (không learned).

2.4. Class Imbalance Problem

2.4.1. Định nghĩa và Tác động

2.4.1.1. Class Imbalance là gì?

Dataset có **class imbalance** khi phân phối các class không đều:

$$\text{Imbalance Ratio (IR)} = \frac{\max_i |C_i|}{\min_j |C_j|} \tag{2.9}$$

với $|C_i|$ là số mẫu của class i.

Ví dụ trong đề tài (Kaggle Dataset): Dataset bao gồm Positive: 9,777 (44.8%), Neutral: 8,013 (35.7%), Negative: 4,031 (18.5%), với IR $= 9,777/4,031 = 2.43$.

2.4.1.2. Tác động đến Machine Learning

1. **Bias toward Majority Class:** Model học theo đường tắt (shortcut), luôn predict Positive để đạt 44.8% accuracy, và bỏ qua Negative class (quan trọng nhất cho dev).
2. **Poor Minority Class Performance:** Recall Negative thấp (nhiều false negatives) vì model không học đủ patterns từ ít mẫu.
3. **Misleading Accuracy:** Accuracy cao không có nghĩa model tốt, ví dụ predict all Positive đạt 44.8% accuracy nhưng useless.

Ví dụ minh họa:

Table 2.5: Model performance trên imbalanced data

Class	Precision	Recall	F1-Score
Positive (majority)	0.88	0.92	0.90
Neutral (medium)	0.78	0.82	0.80
Negative (minority)	**0.65**	**0.58**	**0.61**

→ Negative class (quan trọng nhất) có performance thấp nhất!

2.4.2. Focal Loss

2.4.2.1. Motivation

Cross Entropy Loss chuẩn:

$$CE(p_t) = -\log(p_t) \tag{2.10}$$

với p_t là xác suất predicted cho true class.

Vấn đề: Tất cả mẫu contribute equally, kể cả "easy examples" (high confidence, $p_t \approx 1$).

Focal Loss idea: Down-weight easy examples, focus on hard examples.

2.4.2.2. Công thức Focal Loss

Lin et al. [1] đề xuất Focal Loss cho object detection, sau đó được adapt cho classification:

$$FL(p_t) = -\alpha_t(1-p_t)^\gamma \log(p_t) \tag{2.11}$$

Thành phần:

1. $(1-p_t)^\gamma$**: Modulating factor** với easy example ($p_t \to 1$) thì $(1-p_t)^\gamma \to 0$ dẫn đến loss ≈ 0, trong khi hard example ($p_t \to 0$) thì $(1-p_t)^\gamma \to 1$ dẫn đến loss $\approx CE$. Tham số γ controls focusing degree.
2. α_t**: Balancing factor** với α for positive class và $1-\alpha$ for negative class, compensate for class imbalance (similar to class weighting).

2.4.2.3. Tác động của Hyperparameters

Gamma (γ):

Table 2.6: Tác động của γ (với $p_t = 0.9$ - easy example)

γ	$(1-p_t)^\gamma$	**Loss reduction**	**Behavior**
0	1.0	0%	Cross Entropy
1	0.1	90%	Moderate focusing
2	0.01	99%	Strong focusing
5	0.00001	99.999%	Extreme focusing

Khuyến nghị: $\gamma = 2$ (paper recommendation), balance giữa easy/hard examples.

Alpha (α):

Giá trị $\alpha = 0.25$ tương ứng với minority class (negative) weight $= 1 - 0.25 = 0.75$, $\alpha = 0.5$ là balanced (không bias), và $\alpha = 0.75$ cho majority class weight $= 0.75$.

Trong đề tài: $\alpha = 0.25$, $\gamma = 2$ (optimal từ experiments).

Gradient Analysis:

Gradient của Focal Loss:

$$\frac{\partial FL}{\partial x} = \alpha_t y(1-p_t)^\gamma \left[\gamma p_t \log(p_t) + p_t - 1\right] \tag{2.12}$$

Với easy example ($p_t \to 1$), gradient $\to 0$ (ít update), trong khi hard example ($p_t \to 0$) có gradient lớn (nhiều update).

2.4.3. Class Weighting

2.4.3.1. Động lực và Công thức

Class Weighting là phương pháp đơn giản nhưng hiệu quả để xử lý mất cân bằng dữ liệu bằng cách gán trọng số khác nhau cho từng lớp trong hàm mất mát. Không giống như Focal Loss tập trung vào hard examples, Class Weighting điều chỉnh contribution của từng lớp dựa trên tần suất xuất hiện.

Inverse Frequency Weighting:

$$w_j = \frac{N}{K \cdot n_j} \tag{2.13}$$

với N là tổng số mẫu trong tập huấn luyện (21,821), K là số classes (3), và n_j là số mẫu của class j.

Ví dụ tính toán với tập dữ liệu r/gaming:

$$w_{\text{Positive}} = \frac{21,821}{3 \times 9,777} = 0.744 \tag{2.14}$$

$$w_{\text{Neutral}} = \frac{21,821}{3 \times 8,013} = 0.908 \tag{2.15}$$

$$w_{\text{Negative}} = \frac{21,821}{3 \times 4,031} = 1.804 \tag{2.16}$$

→ Class Negative (thiểu số) nhận trọng số gấp **2.43 lần** so với class Positive (đa số), giúp mô hình chú ý nhiều hơn đến các mẫu Negative trong quá trình huấn luyện.

2.4.3.2. Weighted Cross Entropy Loss

Trọng số được tích hợp trực tiếp vào Cross Entropy Loss:

$$WCE = -\sum_{i=1}^{N} w_{y_i} \log(p_{y_i}) \tag{2.17}$$

với N là số mẫu trong batch, y_i là nhãn thực tế của mẫu thứ i, p_{y_i} là xác suất dự đoán cho lớp đúng, và w_{y_i} là trọng số tương ứng với lớp y_i.

Cơ chế hoạt động: Khi một mẫu thuộc lớp thiểu số bị misclassify, loss value sẽ được nhân với trọng số lớn hơn (ví dụ: $w_{\text{Negative}} = 1.804$), tạo gradient mạnh hơn và buộc mô hình phải học tốt hơn trên lớp đó. Ngược lại, lớp đa số có trọng số nhỏ hơn ($w_{\text{Positive}} = 0.744$), giảm ảnh hưởng của chúng trong tổng loss, tránh mô hình bị bias về lớp đa số.

2.4.3.3. So sánh với Focal Loss

Table 2.7: So sánh Class Weighting vs Focal Loss

Tiêu chí	**Class Weighting**	**Focal Loss**
Cơ chế	Điều chỉnh loss theo class frequency	Điều chỉnh loss theo prediction confidence
Focus	Thiểu số vs đa số	Hard vs easy examples
Hyperparameters	Không cần tuning (auto-computed)	Cần tune γ và α
Ưu điểm	Đơn giản, ổn định, dễ implement	Mạnh hơn với hard examples
Nhược điểm	Không xử lý được hard examples	Phức tạp hơn, cần tuning cẩn thận

2.4.4. Data-Level Methods

Ngoài algorithm-level (Focal Loss, Weighting), còn có data-level methods:

1. **Undersampling:** Phương pháp giảm majority class xuống balance với minority, với pros là simple và balanced training, nhưng cons là mất data (21,821 → 12,093) và information loss.
2. **Oversampling:** Phương pháp duplicate minority class samples, với pros là không mất data, nhưng cons là overfitting risk và longer training.
3. **SMOTE (Synthetic Minority Oversampling):** Phương pháp generate synthetic samples by interpolation, với pros là tăng diversity và không duplicate, nhưng cons là không phù hợp với text (discrete tokens).

Đề tài chọn Algorithm-level (Focal Loss, Weighting): Không mất data, sử dụng full 21,821 samples.

Chương 3

PHƯƠNG PHÁP ĐỀ XUẤT

3.1. Tổng quan Pipeline 4-Stage

Đề tài xây dựng pipeline 4 stages để so sánh systematic giữa Weak Supervision và Supervised Learning trên gaming sentiment analysis.

3.1.1. Kiến trúc Pipeline

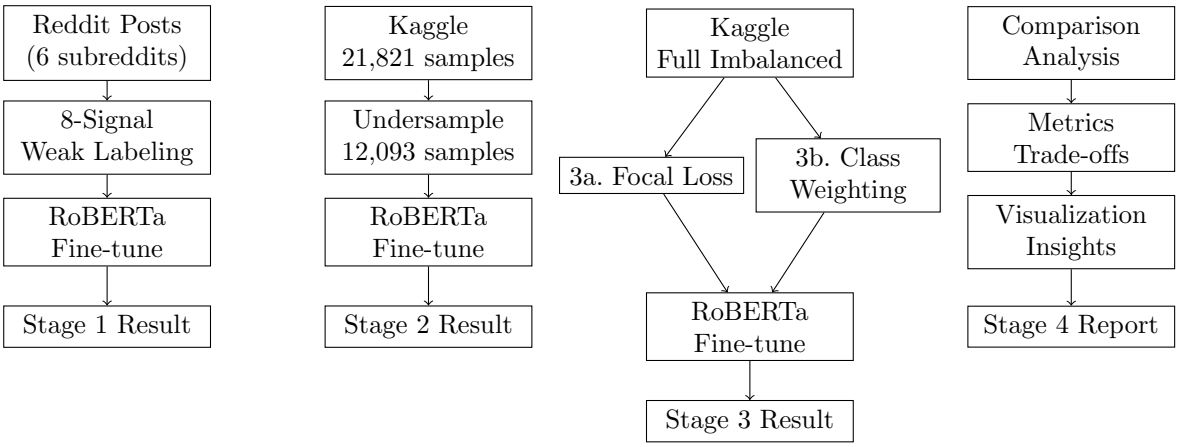

Figure 3.1: Pipeline 4-Stage Overview

3.1.2. Data Flow

Dữ liệu đầu vào (input) cho pipeline là raw gaming text bao gồm Reddit posts và Kaggle reviews. Dữ liệu này sẽ đi qua 4 stages khác nhau, mỗi stage có phương pháp xử lý riêng biệt như mô tả dưới đây.

3.1.2.1. Stage 1: Weak Supervision Flow

Luồng dữ liệu: Reddit Posts $\xrightarrow{\text{8 Signals}}$ Weak Labels $\xrightarrow{\text{RoBERTa}}$ Model_1

Mô tả: Stage 1 sử dụng dữ liệu Reddit posts từ 6 subreddits gaming khác nhau. Dữ liệu này chưa có nhãn sẵn, do đó đề tài áp dụng **8-Signal Weak Labeling Strategy** để tự động gắn nhãn cho dữ liệu. 8 signals bao gồm Awards, Comments, Upvote Ratio, Score, Gaming Text Features, Sarcasm Detection, Flair Analysis, và Top Comments Sentiment. Mỗi signal sẽ vote cho một sentiment label với mức trọng số khác nhau, sau đó weighted voting mechanism kết hợp tất cả để tạo weak labels với confidence score. Chỉ những samples có confidence ≥ 0.6 mới được giữ lại để fine-tune RoBERTa model, tạo ra Model_1.

3.1.2.2. Stage 2: Balanced Supervised Learning Flow

Luồng dữ liệu: Kaggle Reviews $\xrightarrow{\text{Undersample}}$ Balanced Data $\xrightarrow{\text{RoBERTa + CE}}$ Model_2

Mô tả: Stage 2 sử dụng Kaggle dataset với 21,821 samples đã có ground truth labels. Tuy nhiên, dataset này có class imbalance với Positive chiếm 44.8%, Neutral 35.7%, và Negative chỉ 18.5%. Để tạo baseline có dữ liệu cân bằng, đề tài áp dụng **undersampling** giảm xuống còn 12,093 samples với phân bố đều giữa 3 classes. Model RoBERTa sau đó được fine-tune trên balanced data này với Cross Entropy Loss chuẩn, tạo ra Model_2 làm baseline để so sánh.

3.1.2.3. Stage 3: Advanced Loss Functions Flow

Luồng dữ liệu: Kaggle Full $\xrightarrow{\text{3a: Focal Loss}\vee\text{3b: Class Weight}}$ RoBERTa $\xrightarrow{\text{Fine-tune}}$ Model_3

Mô tả: Stage 3 sử dụng **full imbalanced Kaggle dataset** (21,821 samples) không undersample, nhằm giữ lại tất cả thông tin. Để giải quyết class imbalance, đề tài thử nghiệm 2 chiến lược algorithm-level: **Stage 3a sử dứng Focal Loss** với $\alpha = 0.25, \gamma = 2.0$ để down-weight easy examples và focus vào hard examples (thường là minority class); **Stage 3b sử dụng Class Weighting** với inverse frequency weighting để tăng trọng số cho minority classes trong loss function. Cả hai phương pháp đều fine-tune RoBERTa trên full data, tạo ra Model_{3a} và Model_{3b} tương ứng.

3.1.2.4. Stage 4: Comparative Analysis Flow

Luồng dữ liệu: (Model_1, Model_2, Model_3) $\xrightarrow{\text{Evaluation}}$ Trade-off Analysis

Mô tả: Stage 4 không train model mới, mà thực hiện **comprehensive evaluation và comparison** giữa tất cả các models từ 3 stages trước. Evaluation bao gồm các metrics như Accuracy, Precision, Recall, F1-Score (macro và weighted), Confusion Matrix. Ngoài ra, stage này còn phân tích trade-offs giữa các phương pháp: Weak Supervision (chất lượng nhãn noisy nhưng không cần labeling cost), Balanced data (mất dữ liệu nhưng đơn giản), Focal Loss (phức tạp hơn nhưng hiệu quả với imbalance), và Class Weighting (đơn giản nhưng hiệu quả). Kết quả cuối cùng là báo cáo chi tiết về insights, recommendations, và visualizations để giúp hiểu rõ ưu nhược điểm của từng approach.

3.2. Stage 1: Weak Supervision với 8-Signal Strategy

3.2.1. Kiến trúc OptimizedWeakLabelGenerator

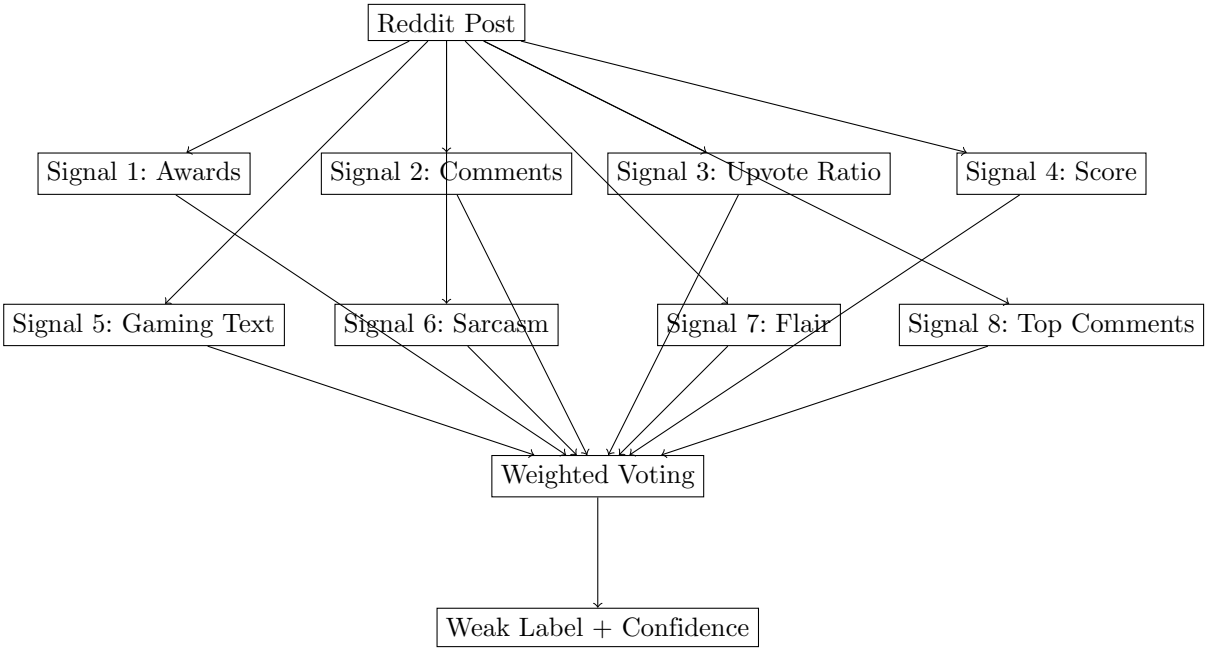

Figure 3.2: OptimizedWeakLabelGenerator Architecture

3.2.2. Chi tiết 8 Signals với Trọng số

3.2.2.1. Signal 1: Awards Analysis (Weight: 4.0)

Reddit awards (Gold, Silver, Platinum) là tín hiệu premium nhất vì require chi phí thực từ users. Khác với upvotes (free), việc trao awards thể hiện sự **highly engaged và genuine appreciation** từ cộng đồng. Một post nhận nhiều awards thường mang nội dung chất lượng cao, hữu ích, hoặc rất positive (ví dụ: game review xuất sắc, gameplay highlights, helpful guides). Do vậy, signal này được gán **trọng số cao nhất (4.0)** trong 8 signals vì độ tin cậy cao và noise thấp.

Logic và Implementation:

```
def analyze_awards(post):
    """
    Phân tích số lượng awards của post để đánh giá sentiment.

    Args:
        post: Reddit post object chứa thông tin total_awards_received

```

```
    Returns:
        tuple: (sentiment_label, weight)
            - 'positive' với weight 4.0 nếu total_awards >= 5 (strongly positive)
            - 'positive' với weight 3.5 nếu total_awards >= 2 (moderately positive)
            - (None, 0) nếu không đủ điều kiện (abstain)
    """
    total_awards = post.total_awards_received

    if total_awards >= 5:
        return ('positive', 4.0)  # Strongly positive - exceptional content
    elif total_awards >= 2:
        return ('positive', 3.5)  # Moderately positive - good content
    else:
        return (None, 0)  # Abstain - not enough signal
```

Giải thích chi tiết: Nếu post có ≥ 5 awards, đây là exceptional content với strongly positive sentiment (weight 4.0 tối đa). Nếu ≥ 2 awards, đây là good content với moderately positive sentiment (weight 3.5). Nếu < 2 awards, signal này không đủ mạnh để kết luận, do đó abstain (không vote). Lý do trọng số cao nhất: awards cần chi phí thực (tiền) nên là high-quality signal với ít noise, khác với upvotes có thể bị manipulate dễ dàng.

3.2.2.2. Signal 2: Comments Count (Weight: 3.0)

Số lượng comments phản ánh mức độ **engagement và discussion intensity** của cộng đồng. Posts với nhiều comments thường là nội dung gây chú ý, thú vị, hoặc controversial. Trong gaming context, high engagement thường liên quan đến positive content (exciting news, impressive gameplay) hoặc heated discussions. Moderate comments (50-100) thường là discussions trung lập, trong khi very low comments (<3) với positive score biểu thị low engagement neutral content.

Logic với dynamic weighting:

```
def analyze_comments(post):
    """
    Phân tích số lượng comments để đánh giá engagement và sentiment.

    Args:
        post: Reddit post object chứa num_comments và score

    Returns:
        tuple: (sentiment_label, weight) dựa trên số comments và score
    """
    num_comments = post.num_comments
    score = post.score

    if num_comments > 200:
        # Very high engagement - usually positive viral content
        return ('positive', 3.0)
    elif num_comments > 100:
        # High engagement - likely positive or interesting
        return ('positive', 2.5)
```

```
    elif 50 <= num_comments <= 100:
        # Moderate discussion - neutral content
        return ('neutral', 2.0)
    elif num_comments < 3 and score > 0:
        # Low engagement but positive score - likely neutral
        return ('neutral', 1.0)
    else:
        # Not enough signal to determine
        return (None, 0)
```

Giải thích chi tiết: Signal này sử dụng **dynamic weighting** dựa trên mức độ engagement. Posts có > 200 comments thường là viral positive content (exciting game releases, amazing updates) với weight 3.0. Posts có > 100 comments vẫn là highly engaging content với weight 2.5. Khoảng 50-100 comments biểu thị moderate discussion (neutral) với weight 2.0. Very low engagement (< 3 comments nhưng positive score) thường là neutral content ít tranh cãi với weight 1.0. Trọng số cao (3.0 max) vì engagement là tín hiệu mạnh về community interest.

3.2.2.3. Signal 3: Upvote Ratio (Weight: 2.5)

Upvote ratio phản ánh **community consensus** về một post. Nó được tính bằng tỷ lệ giữa upvotes và tổng số votes, cho biết

Công thức:

$$\text{Upvote Ratio} = \frac{\text{upvotes}}{\text{upvotes} + \text{downvotes}} \tag{3.1}$$

Threshold Analysis và Giải thích: Upvote ratio ≥ 0.95 biểu thị overwhelming positive consensus (95% users đồng ý) với weight 2.5 - thường là universally praised content. Ratio ≥ 0.85 là strong positive (85% support) với weight 2.2 - content chất lượng cao. Khoảng $0.55 - 0.70$ biểu thị controversial/divisive content với neutral label và weight 2.0 - opinions rất mixed, không rõ ràng positive hay negative. Cuối cùng, < 0.45 biểu thị negative consensus (majority disagrees) với weight 2.5 - thường là complaints, rants, hoặc unpopular opinions. Trọng số 2.5 cao vì upvote ratio là direct measure của community sentiment, không bị bias bởi số lượng votes (khác với score).

3.2.2.4. Signal 4: Post Score (Weight: 2.0)

Post score được tính bằng upvotes - downvotes, là **net popularity indicator** của post. Khác với upvote ratio (tỷ lệ consensus), score phản ánh **số lượng tuyệt đối** users upvote. Score cao (> 500) biểu thị viral positive content với wide reach, score trung bình (> 100) là popular positive content, score gần 0 ($0 - 10$) là low engagement neutral, và negative score (< -10) biểu thị strongly disliked content. Signal này có weight trung bình (2.0) vì có thể bị skew bởi timing và subreddit size.

Logic và Implementation:

```
def analyze_score(post):
```

```
    """
    Phân tích post score (net upvotes) để đánh giá popularity và sentiment.

    Args:
        post: Reddit post object chứa score (upvotes - downvotes)

    Returns:
        tuple: (sentiment_label, weight) dựa trên score thresholds
    """
    score = post.score

    if score > 500:
        # Viral content - very high positive engagement
        return ('positive', 2.0)
    elif score > 100:
        # Popular content - good positive reception
        return ('positive', 1.8)
    elif 0 <= score <= 10:
        # Low engagement - likely neutral
        return ('neutral', 1.8)
    elif score < -10:
        # Heavily downvoted - strongly negative
        return ('negative', 2.0)
    else:
        # Ambiguous range (score between -10 and 0)
        return (None, 0)
```

Giải thích chi tiết: Score > 500 biểu thị viral positive content với weight 2.0 - đây là posts được cộng đồng rất yêu thích. Score > 100 là popular positive content với weight 1.8 - tốt nhưng chưa viral. Score trong khoảng $0 - 10$ biểu thị low engagement neutral với weight 1.8 - không gây tranh cãi. Score < -10 là strongly negative với weight 2.0 - bị community reject mạnh. Trọng số 2.0 trung bình vì score có thể bị ảnh hưởng bởi factors khác (posting time, subreddit size, visibility), không chỉ sentiment.

3.2.2.5. Signal 5: Gaming Text Features (Weight: 1.8-3.0)

Signal này phân tích **gaming-specific vocabulary** trong text để detect sentiment. Gaming community có ngôn ngữ riêng biệt với slang, jargon, và các terms đặc thù không có trong general sentiment lexicons. Đề tài design **3-tier weighting system** dựa trên strength của sentiment indicators: Strong indicators (weight 3.0) là các terms cực kỳ mạnh (GOTY, masterpiece, unplayable), N-grams (weight 2.5) là các phrases phức tạp (worth the price, waste of money), và Single keywords (weight 1.8-2.2) là các từ đơn (amazing, buggy). Trọng số cao (tới 3.0) vì gaming vocabulary là **domain-specific strong signal** rất reliable trong gaming context.

Gaming Lexicon:

Table 3.1: Gaming-Specific Vocabulary (50+ terms)

Tier	Positive Terms	Negative Terms
Strong (3.0)	masterpiece, GOTY, flawless	unplayable, broken mess, cash grab
N-grams (2.5)	worth the price, must play	waste of money, avoid this
Keywords (1.8)	amazing, addictive, polished	buggy, p2w, grindy, laggy

Implementation với 3-Tier System:

```python
def extract_gaming_text_features(text):
    """
    Phân tích gaming-specific vocabulary trong text để detect sentiment.
    Sử dụng 3-tier weighting: Strong (3.0), N-grams (2.5), Keywords (1.8-2.2).

    Args:
        text: String chứa post title và content

    Returns:
        tuple: (sentiment_label, weight) dựa trên strongest gaming term found
    """
    text_lower = text.lower()

    # Tier 1: Strong indicators (weight 3.0) - cực kỳ mạnh
    strong_pos = ['masterpiece', 'goty', 'game of the year', 'flawless',
                  'perfect game', 'instant classic']
    strong_neg = ['unplayable', 'broken mess', 'cash grab', 'scam',
                  'worst game', 'complete failure']

    for term in strong_pos:
        if term in text_lower:
            return ('positive', 3.0)  # Strongest positive signal
    for term in strong_neg:
        if term in text_lower:
            return ('negative', 3.0)  # Strongest negative signal

    # Tier 2: N-grams/Phrases (weight 2.5) - phrases phức tạp
    ngram_pos = ['worth the price', 'must play', 'highly recommend',
                 'best game', 'absolutely love']
    ngram_neg = ['waste of money', 'avoid this', "don't buy",
                 'not worth it', 'save your money']

    for term in ngram_pos:
        if term in text_lower:
            return ('positive', 2.5)  # Strong positive phrase
    for term in ngram_neg:
        if term in text_lower:
            return ('negative', 2.5)  # Strong negative phrase

    # Tier 3: Single keywords (weight 1.8-2.2) - đếm số lượng keywords
    pos_count = count_positive_keywords(text_lower)
```

```
    neg_count = count_negative_keywords(text_lower)

    if pos_count >= 2:
        # Multiple positive keywords - quite positive
        return ('positive', 2.2)
    elif neg_count >= 2:
        # Multiple negative keywords - quite negative
        return ('negative', 2.2)
    elif pos_count == 1:
        # Single positive keyword - moderately positive
        return ('positive', 1.8)
    elif neg_count == 1:
        # Single negative keyword - moderately negative
        return ('negative', 1.8)

    # No gaming keywords found
    return (None, 0)
```

Giải thích chi tiết: Hàm kiểm tra text theo thứ tự **từ mạnh đến yếu**. Đầu tiên kiểm tra Strong indicators (tier 1) với weight 3.0 - các terms cực kỳ mạnh như "GOTY" (Game of the Year), "masterpiece", "unplayable". Nếu không tìm thấy, kiểm tra N-grams (tier 2) với weight 2.5 - các phrases phức tạp như "worth the price", "waste of money". Cuối cùng đếm Single keywords (tier 3) với weight 1.8-2.2 tùy số lượng. System này đảm bảo strongest signal được ưu tiên, và weight cao hơn cho multiple occurrences.

3.2.2.6. Signal 6: Sarcasm Detection (Flip Sentiment)

Sarcasm là một **thách thức lớn** trong sentiment analysis, đặc biệt trong gaming community nơi users thường dùng sarcasm để biểu thị thất vọng. Ví dụ: "Yeah, 60 FPS on a 4090, totally optimized /s" - literal sentiment là positive ("totally optimized") nhưng thực tế là negative sarcasm. Signal này không gán weight trực tiếp mà **flip sentiment** (positive ↔ negative) khi detect sarcasm markers. Đây là **modifier signal** quan trọng để correct false positives/negatives.

Patterns: Có 2 loại patterns: **explicit sarcasm** với "/s" tag rõ ràng (Reddit convention), và **implicit sarcasm** với các phrases như "yeah right", "totally", "sure", "perfectly" kết hợp với negative context.

Logic và Implementation:

```
def detect_sarcasm(text):
    """
    Detect sarcasm markers trong text.

    Args:
        text: String chứa post title và content

    Returns:
        bool: True nếu detect sarcasm, False nếu không
    """
    # List of sarcasm markers (both explicit and implicit)
```

```
    sarcasm_markers = [
        '/s',                    # Explicit Reddit sarcasm tag
        'yeah right',            # Implicit sarcasm phrase
        'totally balanced',      # Gaming sarcasm (unbalanced mechanics)
        'perfectly optimized',   # Gaming sarcasm (poor optimization)
        'great job /s',          # Explicit negative sarcasm
        'well done /s',          # Explicit negative sarcasm
        'amazing work /s'        # Explicit negative sarcasm
    ]

    text_lower = text.lower()
    for marker in sarcasm_markers:
        if marker in text_lower:
            return True  # Sarcasm detected
    return False  # No sarcasm

# Usage in main labeling function:
if detect_sarcasm(text):
    # Flip sentiment: positive <-> negative (neutral unchanged)
    if predicted_label == 'positive':
        predicted_label = 'negative'
    elif predicted_label == 'negative':
        predicted_label = 'positive'
    # Note: neutral remains neutral (no flip needed)
```

Giải thích chi tiết: Hàm kiểm tra danh sách sarcasm markers trong text. Nếu detect được bất kỳ marker nào, hàm trả về True. Trong main labeling logic, nếu detect sarcasm, sentiment label sẽ **bị flip**: positive → negative và ngược lại (neutral không flip). Ví dụ: nếu text "This game is great /s" ban đầu được label positive (do từ "great"), sau khi detect "/s" sẽ flip thành negative (thể hiện thực tế là game tệ). Signal này rất quan trọng để handle gaming community's sarcastic tone.

3.2.2.7. Signal 7: Flair Analysis (Weight: 2.0-3.5)

Reddit flairs là **user-assigned categories** cho posts, thường rất reliable vì users tự chọn flair phù hợp với nội dung. Flairs như "Rant", "Bug", "Complaint" chỉ rõ negative sentiment, trong khi "Praise", "Recommendation" chỉ positive sentiment. "Discussion", "Question" thường neutral. Signal này có **weight rất cao (2.0-3.5)** vì flair là explicit intent signal từ chính user, không cần inference. Ví dụ: post với flair "Rant" chắc chắn là negative (weight 3.5), "Praise" chắc chắn positive (weight 3.5).

Flair Mapping và Implementation:

```
def analyze_flair(post):
    """
    Phân tích Reddit post flair để determine sentiment.

    Args:
        post: Reddit post object chứa link_flair_text

```

```
    Returns:
        tuple: (sentiment_label, weight) dựa trên flair mapping
    """
    # Flair-to-sentiment mapping với weights
    flair_mapping = {
        # Negative flairs (high weight - strong signal)
        'Rant': ('negative', 3.5),         # Chắc chắn negative
        'Bug': ('negative', 3.0),          # Technical issues
        'Complaint': ('negative', 3.0),    # User complaints
        'Issue': ('negative', 2.5),        # Problems/issues

        # Positive flairs (high weight - strong signal)
        'Praise': ('positive', 3.5),       # Chắc chắn positive
        'Recommendation': ('positive', 3.0), # Recommend game
        'Review': ('positive', 2.5),       # Usually positive reviews

        # Neutral flairs (medium weight)
        'Discussion': ('neutral', 2.0),    # General discussion
        'Question': ('neutral', 2.0),      # Asking questions
        'Help': ('neutral', 2.0)           # Seeking help
    }

    flair = post.link_flair_text
    if flair and flair in flair_mapping:
        return flair_mapping[flair]  # Return (label, weight)
    else:
        return (None, 0)  # No flair or unknown flair
```

Giải thích chi tiết: Hàm kiểm tra post flair và map với sentiment label tương ứng. **Negative flairs** (Rant, Bug, Complaint) có weight cao nhất 3.0-3.5 vì rõ ràng biểu thị user dissatisfaction. **Positive flairs** (Praise, Recommendation) cũng có weight 2.5-3.5 biểu thị satisfaction. **Neutral flairs** (Discussion, Question, Help) có weight trung bình 2.0. Nếu post không có flair hoặc flair unknown, signal abstain. Flair là một trong những signals reliable nhất vì đến từ user's explicit categorization.

3.2.2.8. Signal 8: Top Comments Sentiment (Weight: 2.0-2.5)

Top comments phản ánh **community response** và discussion tone xung quanh post. Nếu top comments positive (agree, praise, support), post thường positive. Nếu top comments negative (criticize, disagree, complaints), post thường negative. Signal này analyze **top 3 comments** (highest score) vì đây là community consensus - top comments được upvoted nhiều nhất biểu thị majority opinion. Weight 2.0-2.5 vì comments có thể không relate trực tiếp đến post sentiment, nhưng vẫn là useful signal.

Logic và Implementation:

```
def analyze_top_comments(post):
    """
    Phân tích top 3 comments để capture community response sentiment.

```

```
    Args:
        post: Reddit post object để lấy comments

    Returns:
        tuple: (sentiment_label, weight) dựa trên comments sentiment
    """
    # Lấy top 3 comments (sorted by score)
    top_comments = get_top_comments(post, limit=3)

    pos_count = 0  # Count positive comments
    neg_count = 0  # Count negative comments

    for comment in top_comments:
        # Check sentiment of each comment
        if contains_positive_words(comment.body):
            # Weighted by comment score (higher score = stronger signal)
            pos_count += (1 if comment.score < 50 else 1.2)
        elif contains_negative_words(comment.body):
            neg_count += (1 if comment.score < 50 else 1.2)

    # Determine sentiment based on counts
    if pos_count >= 2:
        # Strong positive response (2+ positive comments)
        return ('positive', 2.5)
    elif neg_count >= 2:
        # Strong negative response (2+ negative comments)
        return ('negative', 2.5)
    elif pos_count > neg_count:
        # Moderate positive response
        return ('positive', 2.0)
    elif neg_count > pos_count:
        # Moderate negative response
        return ('negative', 2.0)
    else:
        # Mixed or neutral response
        return ('neutral', 2.0)
```

Giải thích chi tiết: Hàm lấy top 3 comments (highest scores) và kiểm tra sentiment của mỗi comment. Mỗi comment được **weighted by score**: comments với score ≥ 50 (highly upvoted) nhận multiplier 1.2, comments score < 50 nhận 1.0. Nếu có ≥ 2 positive comments, trả về positive với weight 2.5 (strong positive response). Tương tự cho negative. Nếu `pos_count` > `neg_count` (nhưng chưa đạt 2), trả về positive với weight 2.0 (moderate). Nếu bằng nhau hoặc không rõ ràng, trả về neutral với weight 2.0. Signal này capture **community consensus** thông qua top-voted comments.

3.2.3. Weighted Voting Mechanism

3.2.3.1. Công thức Voting

Aggregate 8 signals với trọng số để tạo final label:

$$\text{Score}(c) = \sum_{i=1}^{8} w_i \cdot \mathbb{1}(\text{Signal}_i = c) \tag{3.2}$$

$$\text{Label}_{\text{pred}} = \arg\max_{c \in \{P, N, Ne\}} \text{Score}(c) \tag{3.3}$$

Confidence Scoring:

$$\text{Confidence} = \frac{\text{Score}(\text{Label}_{\text{pred}})}{\sum_{i=1}^{8} w_i} \tag{3.4}$$

với threshold: Confidence ≥ 0.6

3.2.3.2. Implementation

```python
def generate_optimized_weak_label(post):
    """
    Tạo weak label cho Reddit post bằng weighted voting từ 8 signals.

    Args:
        post: Reddit post object

    Returns:
        tuple: (predicted_label, confidence) hoặc (None, confidence) nếu low confidence
    """
    signals = []

    # Collect all 8 signals
    signals.append(analyze_awards(post))              # Signal 1: Awards (weight 4.0)
    signals.append(analyze_comments(post))            # Signal 2: Comments (weight 3.0)
    signals.append(analyze_upvote_ratio(post))        # Signal 3: Upvote Ratio (weight 2.5)
    signals.append(analyze_score(post))               # Signal 4: Score (weight 2.0)
    signals.append(extract_gaming_text_features(post))# Signal 5: Gaming Text (weight 1.8-3.0)
    signals.append(analyze_flair(post))               # Signal 7: Flair (weight 2.0-3.5)
    signals.append(analyze_top_comments(post))        # Signal 8: Top Comments (weight 2.0-2.5)

    # Weighted voting
    votes = {'positive': 0, 'negative': 0, 'neutral': 0}
    total_weight = 0

    for label, weight in signals:
        if label is not None:
            votes[label] += weight
            total_weight += weight

```

```
    # Get winner (label with highest weighted votes)
    predicted_label = max(votes, key=votes.get)
    confidence = votes[predicted_label] / total_weight if total_weight > 0 else 0

    # Signal 6: Sarcasm detection (flip sentiment if sarcasm detected)
    text = post.title + ' ' + post.selftext
    if detect_sarcasm(text):
        if predicted_label == 'positive':
            predicted_label = 'negative'
        elif predicted_label == 'negative':
            predicted_label = 'positive'
        # Note: neutral remains unchanged

    # Check confidence threshold (only keep high-confidence labels)
    if confidence >= 0.6:
        return predicted_label, confidence  # High confidence - accept label
    else:
        return None, confidence  # Low confidence - reject sample
```

3.3. Stage 2 – Supervised Learning (Balanced)

Giai đoạn 2 tập trung xây dựng một mô hình baseline nhằm đánh giá năng lực của RoBERTa trên dữ liệu đã được cân bằng lớp. Mục tiêu là loại bỏ *data bias* từ sự mất cân bằng dữ liệu và cung cấp điểm chuẩn để so sánh với các chiến lược nâng cao (Class Weighting, Focal Loss) trong Giai đoạn 3.

3.3.1. Chiến lược Cân bằng Dữ liệu: Undersampling

Dữ liệu gốc của bộ review game mất cân bằng rõ rệt, với lớp *Positive* chiếm đa số trong khi *Neutral* và *Negative* ít mẫu hơn. Điều này làm tăng nguy cơ mô hình thiên vị lớp đa số, dẫn đến giảm Recall và F1 của các lớp thiểu số, đặc biệt đối với các bình luận tiêu cực. Để khắc phục, chiến lược **Undersampling** được áp dụng, nhằm cân bằng số lượng mẫu giữa các lớp trước khi huấn luyện mà không cần thay đổi hàm mất mát.

Đầu tiên xác định số lượng mẫu tối thiểu $n_{\min}$ trong tập huấn luyện:

$$n_{\min} = \min_{c \in C}(N_c)$$

trong đó N_c là số mẫu của lớp c. Sau đó, mỗi lớp được randomly sampled về $n_{\min}$ mẫu:

$$\text{balanced\_train\_val} = \bigcup_{c \in C} \text{RandomSample}(D_c, n_{\min})$$

Chiến lược này giúp mô hình học đồng đều các đặc trưng của lớp thiểu số, giữ gradient ổn định trong quá trình huấn luyện và không làm mất thông tin quan trọng từ lớp đa số.

```
class_counts = train_val_df[label_col].value_counts()
```

```
min_count = class_counts.min()

balanced_train_val_df = train_val_df.groupby(label_col).apply(
    lambda x: x.sample(min_count, random_state=42)
).reset_index(drop=True)
```

Sau khi cân bằng, dữ liệu được chia Train/Validation theo tỉ lệ 80/20 bằng *stratified split* để duy trì cân bằng lớp, trong khi Test Set vẫn giữ nguyên phân phối gốc, giúp đánh giá mô hình một cách công bằng:

```
train_df, val_df = train_test_split(
    balanced_train_val_df,
    test_size=0.2,
    stratify=balanced_train_val_df[label_col],
    random_state=42
)
```

Chiến lược Undersampling là một *data-level approach* đơn giản nhưng hiệu quả, giúp mô hình học đồng đều đặc trưng của lớp thiểu số mà không làm mất thông tin lớp đa số, đồng thời cung cấp baseline rõ ràng để so sánh với các chiến lược nâng cao như Class Weighting hoặc Focal Loss trong Stage 3.

3.3.2. Tiền xử lý và Cấu hình Huấn luyện

Tokenization và Ánh xạ Nhãn:

- **Tokenizer**: AutoTokenizer từ `cardiffnlp/twitter-roberta-base-sentiment-latest`.
- **Encoding**: `truncation=True`, `padding="max_length"`, `max_length=512` để giữ toàn bộ thông tin bình luận dài.
- **Ánh xạ nhãn**: `label2id`, dữ liệu được chuyển sang HF Dataset để tối ưu tải GPU.

```
tokenizer = AutoTokenizer.from_pretrained(MODEL)

def tokenize(example):
    return tokenizer(example["text"], truncation=True, padding="max_length",
    ↪  max_length=512)
```

⇒ Tokenization chuẩn hóa văn bản, giúp mô hình học chính xác các đặc trưng ngôn ngữ, kể cả slang và emoji phổ biến trong bình luận game.

Hàm compute_metrics:

Hàm này tính các chỉ số *accuracy*, *F1-weighted* và *F1-macro* nhằm đánh giá hiệu năng phân loại một cách toàn diện. F1-weighted phản ánh hiệu năng tổng thể, cân nhắc số lượng mẫu từng

lớp, trong khi F1-macro đánh giá đồng đều giữa tất cả các lớp, đặc biệt quan trọng khi dữ liệu mất cân bằng và dùng để so sánh với các chiến lược nâng cao. Hàm được truyền vào Trainer qua tham số `compute_metrics`, đảm bảo mỗi lần đánh giá trên tập validation hoặc test trả về đầy đủ bộ metric, hỗ trợ tối ưu hóa mô hình.

```python
def compute_metrics(p):
    predictions = np.argmax(p.predictions, axis=1)
    labels = p.label_ids
    acc = accuracy_score(labels, predictions)
    f1_weighted = f1_score(labels, predictions, average='weighted')
    f1_macro = f1_score(labels, predictions, average='macro')
    return {
        'accuracy': acc,
        'f1_weighted': f1_weighted,
        'f1_macro': f1_macro
    }
```

Tham số Huấn luyện và Loss Function

Mô hình RoBERTa được fine-tune trên dữ liệu cân bằng sử dụng `CrossEntropyLoss`, do Undersampling đã cân bằng số lượng mẫu, nên không cần weighted loss. Các tham số huấn luyện được cấu hình qua `TrainingArguments` như sau:

```python
training_args = TrainingArguments(
    output_dir="./stage2_roberta_balanced_final",
    num_train_epochs=3,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=32,
    learning_rate=2e-5,
    weight_decay=0.05,
    warmup_steps=500,
    eval_strategy="steps",
    eval_steps=200,
    load_best_model_at_end=True,
    metric_for_best_model="f1",
    max_grad_norm=1.0
)
```

Phân tích:

- **Learning rate = 2e-5**: giữ nguyên kiến thức tiền huấn luyện, tránh làm mất embedding đã học.
- **Weight decay = 0.05**: thực hiện regularization, giúp hạn chế overfitting trên tập nhỏ sau undersampling.
- **Batch size = 16 (Train) / 32 (Eval)**: cân bằng giữa tốc độ huấn luyện và ổn định gradient.
- **Max gradient norm = 1.0**: áp dụng gradient clipping, tránh hiện tượng exploding gradient.

- **Metric F1-weighted**: đánh giá hiệu quả mô hình đồng đều trên các lớp, đặc biệt quan trọng với lớp thiểu số.
- **Warmup steps = 500**: giúp ổn định quá trình huấn luyện ban đầu, tránh sốc gradient.

Trainer

Khởi tạo Trainer kết hợp mô hình, dataset, tokenizer, data collator, callback và hàm metrics.

```
trainer = Trainer(
    model=model,  # RoBERTa model
    args=training_args,  # training arguments
    train_dataset=train_data,  # training dataset
    eval_dataset=val_data,  # validation dataset
    tokenizer=tokenizer,  # tokenizer cho padding/encoding
    data_collator=DataCollatorWithPadding(tokenizer),  # dynamic padding
    callbacks=[EarlyStoppingCallback(early_stopping_patience=2)],  # early stopping
    compute_metrics=compute_metrics  # hàm tính accuracy, F1-weighted, F1-macro
)
```

Trainer tích hợp toàn bộ pipeline huấn luyện của Hugging Face, trong đó `data_collator` đảm bảo mỗi batch được padding động để tối ưu hiệu năng GPU. Đồng thời, `EarlyStoppingCallback` sẽ dừng quá trình huấn luyện nếu kết quả trên tập validation không cải thiện liên tiếp 2 lần, giúp tránh overfitting và tiết kiệm thời gian. Hàm `compute_metrics` được sử dụng để trả về các chỉ số *accuracy*, *F1-weighted* (phản ánh hiệu năng tổng thể) và *F1-macro* (đánh giá đồng đều tất cả các lớp, đặc biệt quan trọng với dữ liệu mất cân bằng).

3.4. Stage 3a: Focal Loss

3.4.1. Chiến lược: Focal Loss

Để giải quyết vấn đề mất cân bằng dữ liệu nghiêm trọng trong tập dữ liệu Kaggle, đề tài áp dụng chiến lược Focal Loss thay vì Cross Entropy truyền thống.

3.4.1.1. Công thức toán học

Focal Loss được định nghĩa như sau:

$$FL(p_t) = -\alpha_t(1-p_t)^\gamma \log(p_t) \tag{3.5}$$

Trong đó, p_t là xác suất dự đoán đúng của mô hình đối với nhãn thực tế (ground truth); α_t (alpha) là tham số cân bằng (balancing factor) giúp điều chỉnh trọng số giữa các lớp positive/negative; γ (gamma) là tham số tập trung (focusing parameter) giúp điều chỉnh mức độ giảm trọng số của các mẫu dễ; và $(1-p_t)^\gamma$ là hệ số điều chỉnh (modulating factor).

3.4.1.2. Cơ chế hoạt động và Ý nghĩa

Focal Loss hoạt động dựa trên 3 cơ chế chính. **Thứ nhất, giảm trọng số mẫu dễ (Down-weighting easy examples):** Khi một mẫu được phân loại đúng với độ tự tin cao ($p_t \to 1$), hệ số $(1-p_t)^\gamma$ sẽ tiến về 0, làm cho đóng góp của mẫu đó vào tổng loss trở nên rất nhỏ, giúp mô hình không bị chi phối bởi số lượng lớn các mẫu dễ (thường thuộc lớp đa số như Positive). **Thứ hai, tập trung vào mẫu khó (Focusing on hard examples):** Khi một mẫu bị phân loại sai hoặc độ tự tin thấp ($p_t \to 0$), hệ số $(1-p_t)^\gamma$ sẽ gần bằng 1, lúc này Focal Loss xấp xỉ Cross Entropy Loss và giữ nguyên mức độ phạt lớn đối với mô hình, nhờ đó quá trình huấn luyện sẽ tập trung cập nhật trọng số để cải thiện khả năng nhận diện các mẫu khó (thường thuộc lớp thiểu số như Negative). **Thứ ba, cân bằng lớp (Class Balancing):** Tham số α_t được thiết lập khác nhau cho từng lớp để bù đắp sự chênh lệch số lượng mẫu, tương tự như phương pháp Class Weighting nhưng được tích hợp trực tiếp vào hàm loss.

3.4.2. FocalLoss Class Design

Class FocalLoss kế thừa từ `torch.nn.Module` và implement Focal Loss formula. Class này nhận 3 parameters: `alpha` (balancing factor, default 0.25), `gamma` (focusing parameter, default 2.0), và `num_classes` (số classes, default 3 cho sentiment). Method `forward()` thực hiện 7 steps: compute probabilities qua softmax, one-hot encode labels, lấy p_t (probability for true class), tính focal weight $(1-p_t)^\gamma$, tính cross entropy loss, kết hợp với alpha và focal weight, và return mean loss.

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class FocalLoss(nn.Module):
    """
    Focal Loss for addressing class imbalance.

    FL(p_t) = -alpha_t * (1 - p_t)^gamma * log(p_t)

    Args:
        alpha (float): Balancing factor for class weighting
        gamma (float): Focusing parameter for hard examples
        num_classes (int): Number of output classes
    """

    def __init__(self, alpha=0.25, gamma=2.0, num_classes=3):
        super(FocalLoss, self).__init__()
        self.alpha = alpha
        self.gamma = gamma
        self.num_classes = num_classes

    def forward(self, logits, labels):
        """
        Args:
            logits: (batch_size, num_classes) - Raw model outputs
            labels: (batch_size,) - Ground truth class indices
```

```
28
29          Returns:
30              loss: Scalar focal loss value
31          """
32          # Step 1: Compute probabilities via softmax
33          probs = F.softmax(logits, dim=-1)  # (batch, num_classes)
34
35          # Step 2: One-hot encode labels
36          labels_one_hot = F.one_hot(labels, self.num_classes).float()
37
38          # Step 3: Get p_t (probability for true class)
39          p_t = (probs * labels_one_hot).sum(dim=1)  # (batch,)
40
41          # Step 4: Compute focal weight: (1 - p_t)^gamma
42          focal_weight = (1 - p_t) ** self.gamma
43
44          # Step 5: Compute cross entropy loss
45          ce_loss = F.cross_entropy(logits, labels, reduction='none')
46
47          # Step 6: Combine with alpha and focal weight
48          focal_loss = self.alpha * focal_weight * ce_loss
49
50          # Step 7: Return mean loss
51          return focal_loss.mean()
```

3.4.3. FocalLossTrainer

`FocalLossTrainer` là custom Trainer kế thừa từ Hugging Face `Trainer` class, override method `compute_loss()` để sử dụng Focal Loss thay vì Cross Entropy Loss mặc định. Trong `__init__()` method, FocalLoss object được khởi tạo với `alpha` và `gamma` parameters. Method `compute_loss()` thực hiện forward pass qua model, lấy logits, rồi tính Focal Loss với labels. Đây là cách tiêu chuẩn để integrate custom loss function với Hugging Face training pipeline.

```
1   from transformers import Trainer
2
3   class FocalLossTrainer(Trainer):
4       """
5       Custom Trainer using Focal Loss instead of CrossEntropyLoss.
6       """
7
8       def __init__(self, *args, alpha=0.25, gamma=2.0, **kwargs):
9           super().__init__(*args, **kwargs)
10          self.focal_loss = FocalLoss(
11              alpha=alpha,
12              gamma=gamma,
13              num_classes=3
14          )
15
16      def compute_loss(self, model, inputs, return_outputs=False):
17          """
18          Override compute_loss to use Focal Loss.
```

```
        """
        labels = inputs.pop("labels")

        # Forward pass
        outputs = model(**inputs)
        logits = outputs.logits

        # Compute focal loss
        loss = self.focal_loss(logits, labels)

        return (loss, outputs) if return_outputs else loss
```

3.4.4. Training Configuration

Training configuration được thiết lập qua `TrainingArguments` của Hugging Face. Các hyperparameters quan trọng bao gồm: `num_train_epochs=5`, `per_device_train_batch_size=16` và `eval_batch_size=32`, `learning_rate=2e-5` với `weight_decay=0.01` cho optimizer, `warmup_steps=500` để tăng dần learning rate, `evaluation_strategy='epoch'` để evaluate mỗi epoch, `load_best_model_at_end=True` để giữ model tốt nhất, `metric_for_best_model='eval_accuracy'`, và `fp16=True` để enable mixed precision training (giảm memory, tăng tốc độ).

```
from transformers import TrainingArguments

training_args = TrainingArguments(
    output_dir='./results_stage3_focal',

    # Training parameters
    num_train_epochs=5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=32,

    # Optimizer
    learning_rate=2e-5,
    weight_decay=0.01,
    warmup_steps=500,

    # Evaluation
    evaluation_strategy='epoch',
    save_strategy='epoch',
    load_best_model_at_end=True,
    metric_for_best_model='eval_accuracy',

    # Logging
    logging_steps=100,
    logging_dir='./logs_stage3',

    # Hardware
    fp16=True,  # Mixed precision training
)
```

3.4.5. Training Process

Quá trình training bao gồm 3 bước chính. **Bước 1:** Initialize RoBERTa model từ pre-trained checkpoint `cardiffnlp/twitter-roberta-base-sentiment-latest` với `num_labels=3` cho 3-class sentiment. **Bước 2:** Khởi tạo `FocalLossTrainer` với model, training arguments, datasets, và Focal Loss parameters (`alpha=0.25, gamma=2.0`). **Bước 3:** Gọi `trainer.train()` để bắt đầu training, sau đó `trainer.evaluate()` để đánh giá performance trên validation set. Kết quả bao gồm accuracy và F1-score.

```
# Initialize model
from transformers import AutoModelForSequenceClassification

model = AutoModelForSequenceClassification.from_pretrained(
    'cardiffnlp/twitter-roberta-base-sentiment-latest',
    num_labels=3,
    ignore_mismatched_sizes=True
)

# Initialize trainer with Focal Loss
trainer = FocalLossTrainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    alpha=0.25,  # Focal Loss alpha
    gamma=2.0,   # Focal Loss gamma
    compute_metrics=compute_metrics
)

# Train
trainer.train()

# Evaluate
results = trainer.evaluate()
print(f"Accuracy: {results['eval_accuracy']:.4f}")
print(f"F1-Score: {results['eval_f1']:.4f}")
```

3.5. Stage 3b: Class Weighting

3.5.1. Chiến lược Class Weighting

Dữ liệu cảm xúc về game thường mất cân bằng, với lớp Positive chiếm đa số, trong khi Negative và Neutral ít mẫu hơn. Điều này khiến mô hình dễ thiên về lớp đa số, giảm hiệu quả nhận diện các lớp ít xuất hiện – đặc biệt là những bình luận tiêu cực quan trọng trong đánh giá game.

Chiến lược **Class Weighting** giải quyết vấn đề này bằng cách tác động trực tiếp vào hàm mất mát (Loss Function): các lớp thiểu số được tăng trọng số, giúp mô hình học đồng đều trên tất cả các lớp mà không cần thay đổi dữ liệu. Kết hợp với tiêu chí F1-macro, chiến lược này cân bằng giữa độ chính xác tổng thể và khả năng nhận diện lớp thiểu số, phù hợp với dataset gaming reviews có nhiều slang và ngôn ngữ đặc thù.

3.5.1.1. Cơ sở Toán học và Tính toán Trọng số

Chúng em sử dụng **Tần suất Nghịch đảo có điều chỉnh Logarit** (Log-scaled Inverse Frequency Weighting) để giảm trọng số quá lớn cho lớp cực thiểu và giữ quá trình huấn luyện ổn định.

Tính Trọng số Nghịch đảo Logarit

$$w'_j = \frac{1}{\ln(1 + n_j)}$$

- n_j: số lượng mẫu thuộc lớp j trong tập huấn luyện.
- Logarit giúp giảm chênh lệch quá lớn giữa lớp đa số (Positive) và lớp thiểu số (Negative).

Chuẩn hóa Trọng số

$$w_j = \frac{w'_j}{\sum_{k=1}^{K} w'_k}, \quad K = 3 \text{ (Negative, Neutral, Positive)}$$

Chuẩn hóa đảm bảo tổng trọng số bằng 1, giữ tỷ lệ ảnh hưởng hợp lý giữa các lớp.

Code tính trọng số lớp

```
# 1. Lấy số lượng mẫu huấn luyện của từng lớp
train_counts = train_df["label_id"].value_counts().sort_index()

# 2. Tính trọng số nghịch đảo logarit
weights = 1 / np.log1p(train_counts)

# 3. Chuẩn hóa trọng số
weights = weights / weights.sum()

# 4. Chuyển sang PyTorch Tensor và đưa lên GPU
class_weights = torch.tensor(weights.values, dtype=torch.float32).to(device)
```

Phân tích: Trọng số lớn hơn dành cho lớp ít mẫu, nhỏ hơn cho lớp đa số nhưng không quá cực đoan nhờ logarit. Điều này giữ gradient ổn định khi backpropagation, tránh mô hình bị "quá nhạy" với các mẫu hiếm. Đây là một cách tiếp cận thuật toán-level, không thay đổi dữ liệu, giữ nguyên toàn bộ thông tin từ tập huấn luyện.

3.5.1.2. Tích hợp vào Hàm Mất mát (Weighted Cross-Entropy Loss)

$$\text{Loss} = -\frac{1}{N} \sum_{i=1}^{N} w_{y_i} \log(p_{i,y_i})$$

- w_{y_i}: trọng số lớp của nhãn thực tế y_i.
- p_{i,y_i}: xác suất dự đoán cho lớp y_i của mẫu thứ i.

Cơ chế hoạt động

- Khi một mẫu thuộc lớp thiểu số bị phân loại sai, gradient được nhân với trọng số lớn hơn → tăng cường điều chỉnh các tham số của mô hình cho lớp này.
- Mẫu lớp đa số bị sai vẫn có gradient, nhưng trọng số nhỏ hơn → giảm tác động thiên lệch về lớp đa số.
- Kết quả: RoBERTa học đồng đều trên tất cả lớp, cải thiện recall/F1-score cho Negative và Neutral mà không ảnh hưởng nhiều đến Positive.

Code triển khai WeightedCELoss

```
class WeightedCELoss(nn.Module):
    def __init__(self, weights):
        super().__init__()
        self.weights = weights  # Tensor [w_0, w_1, w_2]

    def forward(self, logits, labels):
        return F.cross_entropy(logits, labels, weight=self.weights)

ce_loss_fn = WeightedCELoss(class_weights)
```

Phân tích chi tiết

Cross-Entropy Loss chuẩn: mỗi lỗi mẫu đều ảnh hưởng bằng 1. WeightedCE: nhấn mạnh lỗi của lớp ít mẫu, giảm lỗi lớp nhiều mẫu.

Trong backpropagation:

$$\frac{\partial \text{Loss}}{\partial z_j} = w_{y_i}(p_j - y_j)$$

⇒ Gradient được scale theo trọng số lớp, giúp optimizer điều chỉnh weights của mô hình tập trung hơn vào lớp thiểu số. Chiến lược này đơn giản, dễ triển khai, phù hợp với các dataset gaming reviews mất cân bằng mà vẫn giữ thông tin lớp đa số.So với Cross-Entropy chuẩn, WeightedCE nhấn mạnh lớp ít mẫu, giữ gradient ổn định, cải thiện recall và F1 cho các lớp ít mẫu (Negative/Neutral) mà không làm giảm hiệu năng lớp đa số (Positive).

3.5.2. Tiền xử lí và Cấu hình Huấn luyện

3.5.2.1. Tiền xử lí

Phân chia Dữ liệu

Dữ liệu được chia thành Train, Validation và Test với tỉ lệ 70/15/15, sử dụng *stratified split* để đảm bảo cân bằng tỉ lệ lớp trong từng split. Việc này giúp giữ nguyên phân phối lớp và minh họa trực quan sự cân bằng dữ liệu.

```
# Chia Train/Validation/Test
train_df, temp_df = train_test_split(
    df, test_size=0.3, stratify=df['label_id'], random_state=42
)

val_df, test_df = train_test_split(
    temp_df, test_size=0.5, stratify=temp_df['label_id'], random_state=42
)
```

Tokenization và ánh xạ nhãn

Tokenization được thực hiện bằng `AutoTokenizer` từ mô hình, với truncation và padding `"max_length"`, giữ toàn bộ thông tin từ các bình luận dài. Nhãn được ánh xạ sang dạng số (`label2id`) để phù hợp với mô hình và HF Dataset.

```
tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)

def tokenize_fn(examples):
    return tokenizer(examples[text_col], truncation=True, padding=True, max_length=512)

# Chuyển DataFrame → HF Dataset
train_ds = Dataset.from_pandas(train_df[[text_col,
↪  'label_id']].rename(columns={'label_id': 'label'}))
val_ds   = Dataset.from_pandas(val_df[[text_col,
↪  'label_id']].rename(columns={'label_id': 'label'}))
test_ds  = Dataset.from_pandas(test_df[[text_col,
↪  'label_id']].rename(columns={'label_id': 'label'}))
```

3.5.2.2. Cấu hình Huấn luyện

Weighted Loss và Custom Trainer

Để sử dụng hàm mất mát `WeightedCELoss` trong framework Hugging Face Trainer, chúng em tạo lớp `WeightedTrainer` kế thừa từ `Trainer` và ghi đè phương thức `compute_loss()`. Cơ chế này cho phép mô hình RoBERTa sử dụng trực tiếp trọng số lớp khi tính loss, từ đó học đồng đều trên cả lớp đa số và lớp thiểu số.

```
class WeightedTrainer(Trainer):
    def __init__(self, loss_fn, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.loss_fn = loss_fn

    def compute_loss(self, model, inputs, return_outputs=False, **kwargs):
        labels = inputs.pop("labels")
        outputs = model(**inputs)
```

```
        loss = self.loss_fn(outputs.logits, labels)
        return (loss, outputs) if return_outputs else loss
```

Cấu hình TrainingArguments:

```
training_args = TrainingArguments(
    output_dir="./stage3_roberta_class_weight",  # checkpoint folder
    num_train_epochs=5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=32,
    warmup_steps=500,                            # LR warmup
    learning_rate=1e-5,                          # base LR
    weight_decay=0.01,
    eval_strategy="steps",
    eval_steps=200,
    load_best_model_at_end=True,                 # restore best checkpoint
    metric_for_best_model="f1_macro",
    max_grad_norm=1.0                            # gradient clipping
)
)
```

Các tham số huấn luyện được lựa chọn nhằm đảm bảo quá trình fine-tuning mô hình RoBERTa ổn định và hiệu quả khi áp dụng weighted loss. Learning rate thấp (1e-5) giúp tránh hiện tượng gradient overshoot, đặc biệt khi lớp thiểu số được scale trọng số lên, từ đó giảm rủi ro dao động lớn của loss. Warmup steps 500 cho phép tăng dần learning rate từ 0 đến giá trị gốc trong giai đoạn đầu huấn luyện, giúp tránh sốc gradient và cải thiện sự ổn định, nhất là với tập dữ liệu nhỏ hoặc mất cân bằng. Weight decay 0.01 thực hiện L2 regularization, hạn chế overfitting và tránh mô hình trở nên quá nhạy với các lớp ít mẫu. Kích thước batch được cân đối giữa train và eval nhằm tối ưu tốc độ huấn luyện đồng thời giữ gradient ổn định, giúp optimizer cập nhật weights hiệu quả và tận dụng GPU tốt hơn. Việc đánh giá trên tập validation theo mỗi 200 bước kết hợp với `load_best_model_at_end` đảm bảo lưu lại mô hình tốt nhất dựa trên metric `f1_macro`, từ đó tránh mất hiệu năng do dao động của loss trong huấn luyện. Metric `f1_macro` được ưu tiên nhằm tối ưu đồng đều trên tất cả các lớp, cải thiện recall và F1 cho các lớp thiểu số mà vẫn giữ hiệu năng lớp đa số. Cuối cùng, `max_grad_norm = 1.0` thực hiện gradient clipping, giúp ổn định quá trình backpropagation và tránh exploding gradient, đặc biệt khi gradient của các lớp ít mẫu bị scale lên nhờ weighted loss.

WeightedTrainer

Khởi tạo Trainer kết hợp mô hình, dataset, tokenizer, data collator, callback và hàm loss:

```
trainer = WeightedTrainer(
    loss_fn=ce_loss_fn,                                    # weighted loss
    model=model,                                           # RoBERTa model
    args=training_args,                                    # training arguments
    train_dataset=train_ds,                                # training dataset
    eval_dataset=val_ds,                                   # validation dataset
    tokenizer=tokenizer,                                   # tokenizer for
    ↪ padding/encoding
    data_collator=DataCollatorWithPadding(tokenizer),      # dynamic padding
```

```
    callbacks=[EarlyStoppingCallback(early_stopping_patience=3)], # early stopping
    compute_metrics=compute_metrics                               # metric function
)
```

WeightedTrainer tích hợp mô hình RoBERTa với hàm loss đã cân lớp (weighted loss), giúp gradient của các lớp ít mẫu được tăng cường, từ đó cải thiện khả năng nhận diện lớp thiểu số mà vẫn giữ thông tin lớp đa số. DataCollatorWithPadding đảm bảo mỗi batch được padding động, tối ưu bộ nhớ GPU và tốc độ huấn luyện. EarlyStoppingCallback với `patience=3` dừng huấn luyện khi `f1_macro` trên validation không cải thiện sau 3 lần đánh giá liên tiếp, tránh overfitting. Hàm `compute_metrics` theo dõi `f1_macro`, `f1_weighted` và `accuracy`, cung cấp đánh giá đồng đều giữa các lớp và giám sát hiệu năng mô hình xuyên suốt quá trình huấn luyện.

Như vậy, với chiến lược này, mô hình RoBERTa học đồng đều trên các lớp, cải thiện recall/F1 cho Negative và Neutral, đồng thời giữ hiệu năng lớp Positive. Weighted Cross-Entropy nhấn mạnh lớp ít mẫu nhưng vẫn đảm bảo gradient ổn định, giúp huấn luyện hiệu quả, đơn giản và dễ triển khai.

3.6. Evaluation Metrics

3.6.1. Metrics cho Multi-class Classification

3.6.1.1. Accuracy

Tỷ lệ phân loại đúng tổng thể:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total Samples}} = \frac{\sum_{i=1}^{N} \mathbb{1}(y_i = \hat{y}_i)}{N} \tag{3.6}$$

Ưu điểm: Đơn giản, dễ hiểu

Nhược điểm: Misleading với imbalanced data (high accuracy nhưng poor minority class)

3.6.1.2. Precision, Recall, F1-Score

Precision (Độ chính xác): Tỷ lệ dự đoán đúng trong các predictions positive

$$\text{Precision}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c} \tag{3.7}$$

Recall (Độ phủ): Tỷ lệ phát hiện đúng trong tất cả mẫu positive thực tế

$$\text{Recall}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FN}_c} \tag{3.8}$$

F1-Score: Harmonic mean của Precision và Recall

$$\text{F1}_c = 2 \cdot \frac{\text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c} \tag{3.9}$$

3.6.1.3. Macro vs Weighted F1

F1-Macro: Trung bình không trọng số (tất cả classes đều quan trọng)

$$\text{F1-Macro} = \frac{1}{K} \sum_{c=1}^{K} \text{F1}_c \tag{3.10}$$

F1-Weighted: Trung bình có trọng số theo kích thước class

$$\text{F1-Weighted} = \sum_{c=1}^{K} \frac{n_c}{N} \cdot \text{F1}_c \tag{3.11}$$

với n_c là số mẫu class c, N là tổng số mẫu.

Khi nào dùng:

- F1-Macro: Quan tâm đều tất cả classes (including minority)
- F1-Weighted: Phản ánh hiệu suất tổng thể (majority class matter more)

3.6.2. Confusion Matrix

Định nghĩa: Ma trận $C \in \mathbb{R}^{K \times K}$ với C_{ij} = số mẫu true class i được predict là class j.

$$C = \begin{bmatrix} \text{TP}_{\text{Neg}} & \text{FP}_{\text{Neu (from Neg)}} & \text{FP}_{\text{Pos (from Neg)}} \\ \text{FP}_{\text{Neg (from Neu)}} & \text{TP}_{\text{Neu}} & \text{FP}_{\text{Pos (from Neu)}} \\ \text{FP}_{\text{Neg (from Pos)}} & \text{FP}_{\text{Neu (from Pos)}} & \text{TP}_{\text{Pos}} \end{bmatrix} \tag{3.12}$$

Ví dụ Stage 3 Confusion Matrix (normalized):

Table 3.2: Confusion Matrix (Stage 3 Focal Loss)

True / Pred	Negative	Neutral	Positive
Negative	0.85	0.10	0.05
Neutral	0.08	0.88	0.04
Positive	0.03	0.07	0.90

Phân tích:

- Diagonal cao (0.85, 0.88, 0.90) = good classification
- Main confusion: Negative ↔ Neutral (0.10, 0.08)
- Ít confusion Negative ↔ Positive (0.05, 0.03) = the model distinguishes the extremes well

Chương 4

Thực nghiệm và Kết quả

4.1. Thiết lập môi trường

Quá trình thực nghiệm được thiết kế nhằm đảm bảo tính tái lập (reproducibility) và khai thác tối đa khả năng xử lý của GPU trong quá trình fine-tuning các mô hình ngôn ngữ lớn (LLM).

Môi trường phần cứng và phần mềm

Tiêu chí	**Chi tiết**	**Ghi chú**
Nền tảng (Platform)	Google Colab	Môi trường đám mây tiêu chuẩn, thuận tiện cho chia sẻ và tái lập thí nghiệm
Thiết bị tính toán (Compute)	Tesla T4 GPU	Đảm bảo tốc độ fine-tuning cao, hỗ trợ Mixed Precision
Framework ML	PyTorch	Framework lõi cho huấn luyện mô hình
Thư viện chính	Hugging Face Transformers, Datasets, Accelerate	Cung cấp API mô hình, tiện ích Trainer và quản lý dataset

Table 4.1: Môi trường phần cứng và phần mềm cho các thí nghiệm

Cấu hình mô hình và tiền xử lý

Tiêu chí	Chi tiết	Ghi chú
Mô hình cơ sở	RoBERTa-Twitter (cardiffnlp/twitter-roberta-base-sentiment-latest)	Tối ưu cho văn bản mạng xã hội
Max Token Length	512 tokens	Giới hạn tối đa tokenization, giữ đầy đủ thông tin cho bình luận dài

Table 4.2: Cấu hình mô hình và tiền xử lý

Cấu hình đặc thù Stage 1 – Weak Supervision

Stage 1 được thiết kế nhằm xây dựng nhãn yếu và khởi động mô hình (bootstrap):

- **Dữ liệu:** Thu thập từ 6 subreddit lớn về gaming trên Reddit.
- **Chiến lược nhãn:** Áp dụng "8-Signal Strategy" (Metadata, Content, Context) kết hợp cơ chế Weighted Voting để tạo 3,003 mẫu nhãn yếu với độ tin cậy ≥ 0.6.
- **Huấn luyện:** Mô hình fine-tune trên tập Reddit đã được cân bằng từ nhãn yếu, gồm 2,102 mẫu.

Cấu hình Supervised Learning (Stage 2 & 3)

Tham số	Giá trị sử dụng	Giai đoạn áp dụng
Batch Size (Train)	16 (Effective: 32)	per_device_train_batch_size = 16 + gradient_accumulation_steps = 2 (Stage 3a)
Learning Rate (LR)	$1 \times 10^{-5} - 2 \times 10^{-5}$	Phạm vi LR tối ưu cho fine-tuning LLM
Kỹ thuật mất cân bằng	Focal Loss ($\alpha = 0.25, \gamma = 2.0$)	Stage 3a: Giảm trọng số các mẫu khó và tập trung vào lớp thiểu số
Kỹ thuật mất cân bằng	Class Weighting (Log-scaled)	Stage 3b: Tăng trọng số cho lớp thiểu số dựa trên tần suất nghịch đảo

Table 4.3: Cấu hình Supervised Learning Stage 2 & 3

4.2. Phân tích từng giai đoạn thực nghiệm

Stage 1: Weak Supervision (Reddit Gaming)

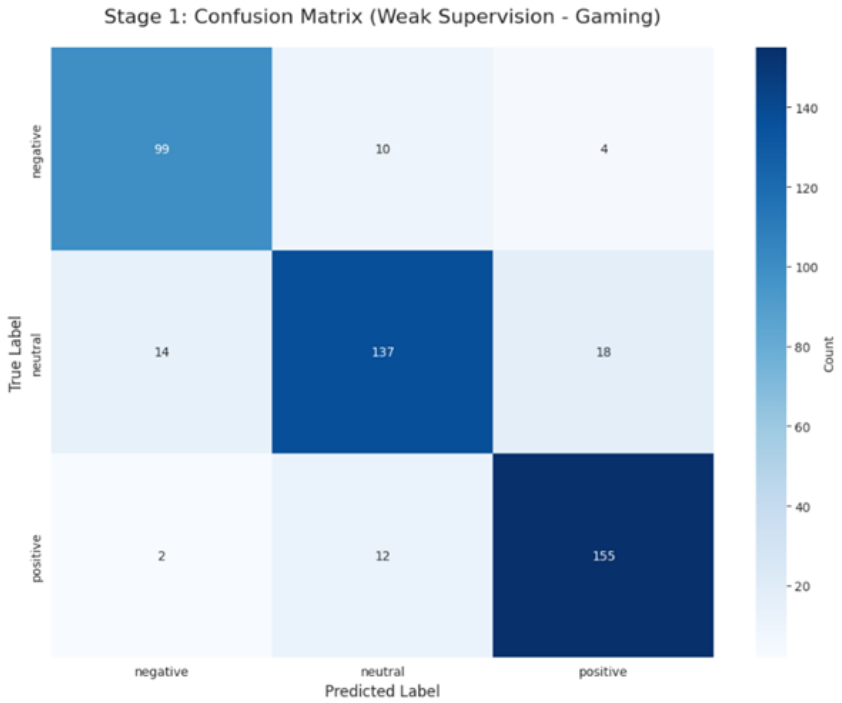

Figure 4.1: Confusion Matrix Stage 1 - Weak Supervision (Reddit Gaming)

Tiêu chí	**Kết quả trên tập Reddit (Weak Test)**	**Kết quả trên tập Kaggle (Cross-Eval)**
Accuracy	86.70%	47.88%
F1-Weighted	86.64%	44.01%
Negative Recall	87.61%	3.82%
Thời gian Train	19.16 phút	0.00 phút (Model có sẵn)

Table 4.4: So sánh Hiệu năng Stage 1 trên Tập Nội bộ và Cross-Evaluation

Phân tích:

- **Hiệu quả nội bộ:** Trên tập Reddit, Negative Recall 87.61%, Positive Recall 91.72% và Neutral Recall 81.07%. Điều này chứng tỏ 8-Signal Weak Supervision tạo nhãn nhiễu chất lượng cao, giúp mô hình học tốt trên miền dữ liệu tương tự.
- **Khoảng cách tổng quát hóa:** Khi đánh giá trên tập Kaggle chuẩn, Accuracy giảm còn 47.88% và Negative Recall chỉ 3.82%, phản ánh Generalization Gap, cho thấy dữ liệu nhãn yếu từ Reddit không tổng quát hóa tốt sang dữ liệu chuẩn.

Stage 2: Supervised Learning (Balanced – Undersampling)

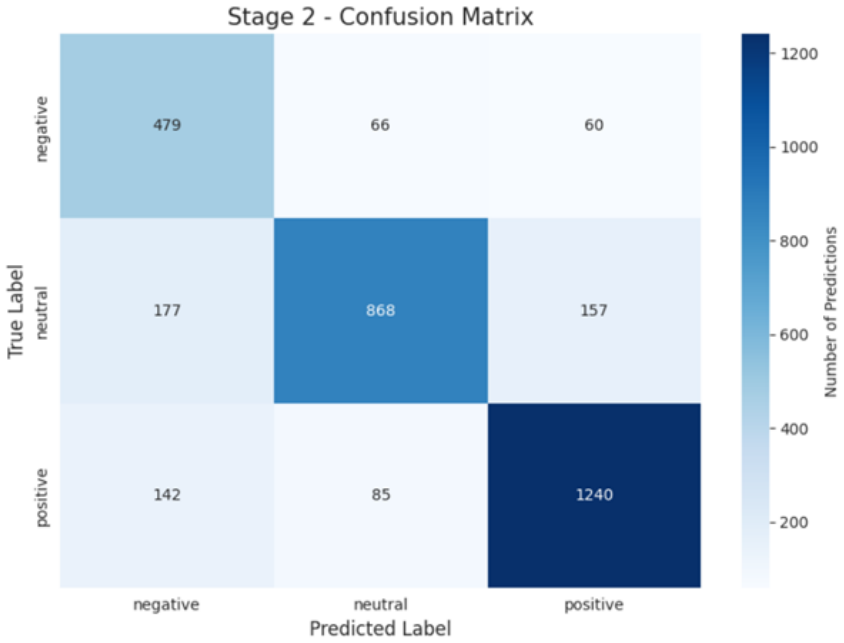

Figure 4.2: Confusion Matrix Stage 2 - Supervised Learning (Balanced – Undersampling)

Tiêu chí	**Kaggle Test Set**	**Chi tiết Loss Function**
Accuracy	79.02%	Standard CrossEntropy
F1-Weighted	79.32%	Final Loss: 0.5560
Negative Recall	79%	Training Time: 52.34 phút

Table 4.5: Kết quả Stage 2 trên tập Kaggle

Phân tích:

- Tác động của Undersampling: Việc loại bỏ khoảng 11.000 mẫu lớp đa số giúp mô hình tập trung vào lớp thiểu số, đạt Negative Recall cao nhất (79%).
- Đánh đổi: Mặc dù tăng khả năng nhận diện lớp thiểu số, hiệu năng tổng thể giảm so với Stage 3 (Accuracy 79.02% so với >82%), và Neutral Recall cũng giảm xuống 72%. Final Loss 0.5560 phản ánh mô hình học ổn định, thấp hơn hẳn mức ngẫu nhiên (~ 1.098).

Stage 3: Supervised Learning (Imbalanced – Focal Loss / Class Weighting)

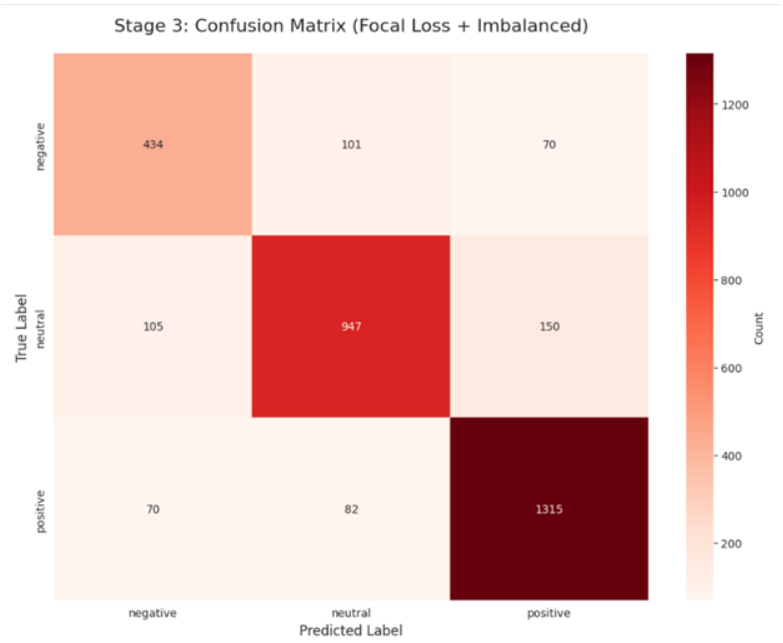

Figure 4.3: Confusion Matrix Stage 3: Supervised Learning (Imbalanced – Focal Loss)

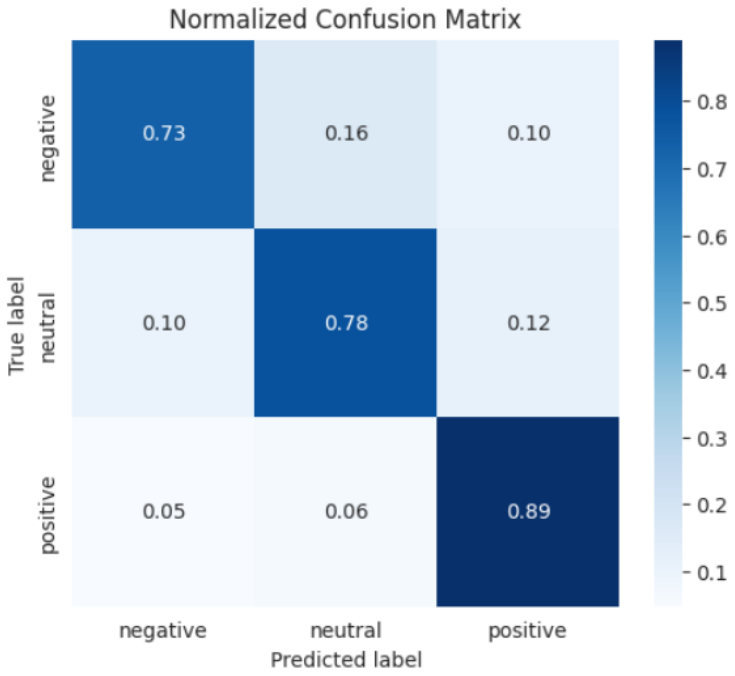

Figure 4.4: Confusion Matrix Stage 3: Supervised Learning (Imbalanced – Class Weighting)

Tiêu chí	Stage 3a (Focal Loss)	Stage 3b (Class Weighting)
Accuracy	82.35%	82.22%
F1-Weighted	82.29%	82.22%
Negative Recall	71.74%	73%
Final Loss	0.0416	0.4895
Thời gian Train	35.86 phút	86.17 phút

Table 4.6: Kết quả Stage 3: Focal Loss và Class Weighting

Phân tích:

- Hiệu quả tổng thể: Cả hai phương pháp Stage 3 đều tận dụng toàn bộ 15.274 mẫu dữ liệu, giúp đạt Accuracy >82% và F1-weighted tương đương, cao hơn Stage 2 khoảng 3%.
- Đặc trưng Focal Loss (Stage 3a): Loss giảm xuống 0.0416, cho thấy mô hình tập trung hiệu quả vào các mẫu khó (hard examples). Thời gian huấn luyện ngắn hơn gần 2.5 lần so với Class Weighting, phản ánh ưu điểm về hiệu suất.
- Đặc trưng Class Weighting (Stage 3b): Tăng Negative Recall lên 73%, nhưng mất nhiều thời gian huấn luyện hơn (86.17 phút), cho thấy trade-off giữa tốc độ và cân bằng lớp.

4.3. Tổng hợp và So sánh Hiệu năng Đa chiều

4.3.1. Kết quả định lượng tổng hợp

Kết quả từ bốn giai đoạn thử nghiệm được tổng hợp và sắp xếp theo độ chính xác giảm dần.

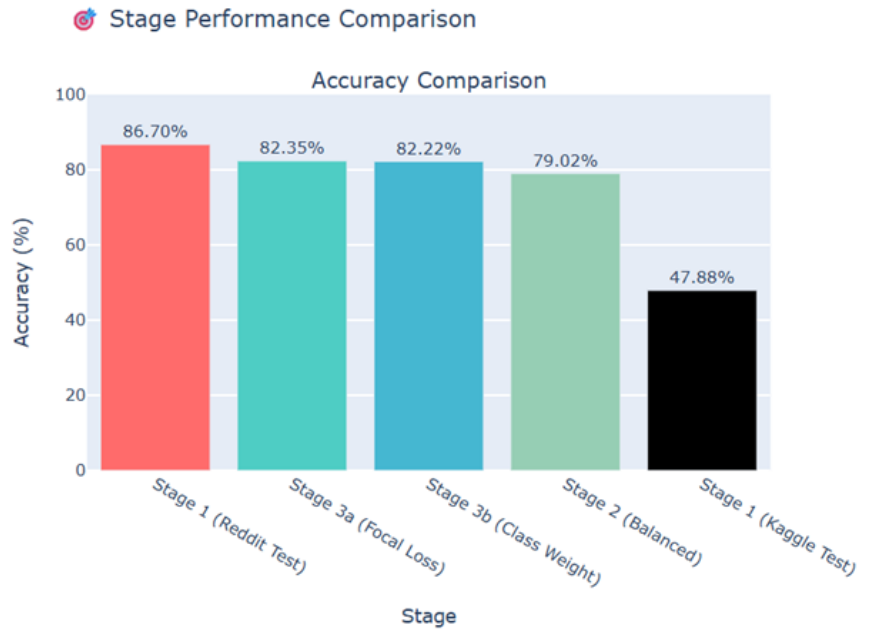

Figure 4.5: Accuracy Comparison

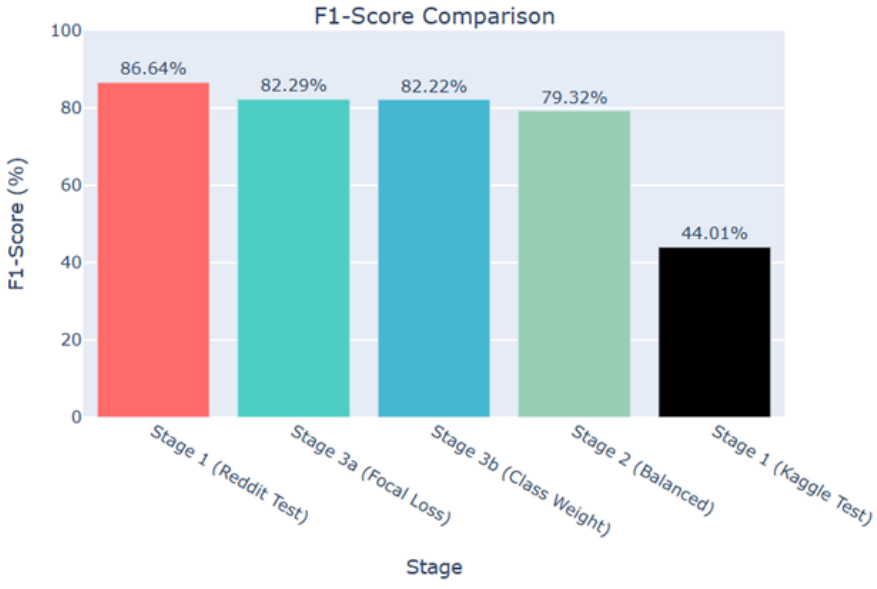

Figure 4.6: F1-Score Comparison

Phương pháp	Accuracy (%)	F1-Score (Weighted %)	Thời gian Train (phút)	Test Set	Manual Labels
Stage 1 (Weak)	86.70	86.64	19.16	Reddit (Weak)	No
Stage 3a (Focal Loss)	82.35	82.29	35.86	Kaggle (Human)	Yes
Stage 3b (Class Weighting)	82.22	82.22	86.17	Kaggle (Human)	Yes
Stage 2 (Balanced)	79.02	79.32	52.34	Kaggle (Human)	Yes
Stage 1 (Cross-Eval)	47.88	44.01	0.00	Kaggle (Human)	No (Weak)

Table 4.7: So sánh Hiệu năng tổng hợp các chiến lược huấn luyện

Tổng hợp cho thấy hiệu năng các chiến lược phân tách thành hai nhóm rõ rệt. Nhóm Weak Supervision (Stage 1) đạt hiệu năng nội bộ cao nhất (86.70% Accuracy) và có tốc độ nhanh nhất (19.16 phút). Tuy nhiên, rủi ro tổng quát hóa là rất lớn, được chứng minh bằng sự sụt giảm $\approx 38.82\%$ Accuracy khi kiểm tra trên dữ liệu chuẩn Kaggle (chỉ còn 47.88%). Ngược lại, Nhóm Supervised (Stage 2 & 3) có hiệu năng tổng thể ổn định hơn. Các phương pháp Algorithm-Level (Stage 3a/3b) tận dụng toàn bộ dữ liệu gốc (15,274 mẫu) đã đạt Accuracy cao nhất (trên 82%), vượt trội so với Undersampling (Stage 2) (79.02% Accuracy) do việc giảm mẫu làm giảm khả năng học của mô hình. Trong nhóm Supervised, Focal Loss (Stage 3a) nổi bật về hiệu suất khi đạt Accuracy 82.35% với thời gian huấn luyện nhanh nhất (35.86 phút), làm nổi bật ưu thế về hiệu suất tính toán khi xử lý dữ liệu mất cân bằng.

4.3.2. Phân tích hiệu suất lớp

Phân tích chi tiết hiệu quả phân loại trên lớp thiểu số (Negative) được thể hiện trong bảng 4.8.

Phương pháp	Negative Recall (%)	Neutral Recall (%)	Positive Recall (%)
Stage 1 (Weak)	87.61	81.07	91.72
Stage 3a (Focal Loss)	71.74	78.79	89.64
Stage 3b (Class Weighting)	73	78	89
Stage 2 (Balanced)	79	72	85

Table 4.8: Tỷ lệ phân loại lớp thiểu số (Negative Recall)

Nhận xét:

- Stage 2 (Balanced) đạt Recall cao nhất trên lớp Negative (79%), phản ánh đặc trưng của phương pháp Undersampling: loại bỏ một phần mẫu lớp đa số khiến mô hình phải học sâu hơn từ các mẫu thiểu số, giảm bỏ sót Negative.
- Stage 2 cải thiện Recall lớp Negative (79%), nhưng đi kèm với giảm Accuracy tổng thể (79.02% vs. 82.35% của Stage 3) và Precision. Điều này cho thấy cần cân nhắc giữa khả năng nhận diện lớp thiểu số và hiệu năng tổng thể.

4.3.3. Phân tích đa chiều

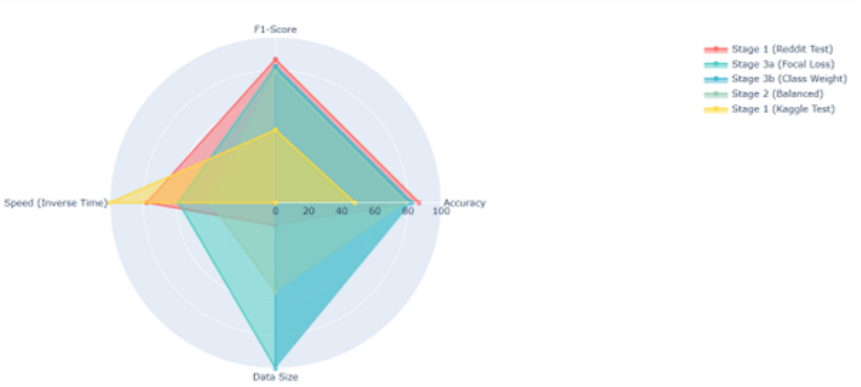

Figure 4.7: Biểu đồ Radar Chart

Biểu đồ Radar Chart được sử dụng để trực quan hóa sự đánh đổi giữa các yếu tố quan trọng trong quá trình huấn luyện và đánh giá mô hình. Các tiêu chí chính bao gồm:

- **Accuracy và F1-Score:** Các phương pháp Supervised Stage 3a (Focal Loss) và Stage 3b (Class Weighting) thể hiện hiệu năng tốt nhất trên tập dữ liệu chuẩn, khẳng định chất lượng phân loại cao.
- **Tận dụng dữ liệu (Data Size):** Stage 3a và 3b khai thác toàn bộ 15,274 mẫu, trong khi Stage 2 chỉ sử dụng 8,222 mẫu, làm hạn chế khả năng học của mô hình và giảm hiệu năng tổng thể.
- **Tốc độ huấn luyện (Speed – Inverse Time):** Stage 1 trên Reddit Test Set có thời gian nhanh nhất (19.16 phút), trong khi Stage 3a là phương pháp Supervised mất cân bằng nhanh nhất (35.86 phút), còn Stage 3b mất nhiều thời gian nhất (86.17 phút) do cơ chế Class Weighting tốn kém hơn.

⇒ Stage 3a (Focal Loss) được đánh giá là chiến lược tối ưu cho triển khai thực tế (Production), nhờ khả năng đạt Accuracy cao nhất (82.35%) trong các phương pháp sử dụng nhãn chuẩn, đồng thời tiết kiệm thời gian huấn luyện hơn so với Class Weighting. Biểu đồ Radar Chart minh họa rõ ràng lợi ích đồng thời của việc sử dụng toàn bộ dữ liệu và tốc độ huấn luyện hợp lý.

Chương 5

Đánh giá và Thảo luận

Chương này không chỉ trình bày kết quả mà còn phân tích cơ chế kỹ thuật tạo ra các hiện tượng quan sát được, từ đó làm rõ ưu – nhược điểm – trade-off của từng chiến lược huấn luyện trong pipeline. Mục tiêu là giải thích hành vi của mô hình dựa trên dữ liệu và phương pháp học, thay vì chỉ báo cáo số liệu.

5.1. Giới hạn và Rủi ro của Weak Supervision (Stage 1)

Stage 1 chứng minh rằng mô hình chịu ảnh hưởng mạnh bởi **Domain Shift** và **Heuristic Overfitting** khi học từ tập dữ liệu Reddit gán nhãn tự động.

Kết quả định lượng quan trọng:

Chỉ số	**Stage 1**
Accuracy	47.88% (giảm $\approx$ 38.82% từ 86.70%)
Negative Recall	3.82% (giảm từ 87.61%)
Training Loss trung bình	0.5826
Thời gian huấn luyện	1149.50s ($\sim$19.16 phút)

Table 5.1: Hiệu năng định lượng của Stage 1 (Weak Supervision)

Giải thích Training Loss: Giá trị Training Loss ≈ 0.58 phản ánh mô hình hội tụ ổn định trên dữ liệu Reddit, mặc dù hiệu năng trên tập dữ liệu chuẩn giảm mạnh. Loss chưa quá cao, cho thấy mô hình vẫn học được các mối tương quan trong dữ liệu nguồn, nhưng các mối quan hệ này không tổng quát sang tập dữ liệu khác, dẫn tới sụt giảm Negative Recall và Accuracy — minh họa điển hình cho **Domain Shift**.

Nguyên nhân sụt giảm hiệu năng: Mô hình học các mẫu không liên quan đến ngữ nghĩa thực, bao gồm:

- Cấu trúc comment đặc thù Reddit,

- Token và slang của cộng đồng Gaming/Meme,
- Giọng điệu meme / Internet culture,
- Phân phối từ vựng lệch hẳn so với dữ liệu chuẩn.

Như vậy, Stage 1 không thất bại do thuật toán, mà là do môi trường học không đồng nhất — một minh chứng rõ ràng cho **Domain Adaptation Failure**.

Trade-off của Weak Supervision:

Lợi ích	**Rủi ro**
Tiết kiệm chi phí: huấn luyện nhanh (∼19.16 phút)	Sai lệch ngữ nghĩa nghiêm trọng
Tạo pseudo-label nhanh ở quy mô lớn	Học nhầm các dấu hiệu "meme-biased"
Phù hợp làm pretraining cho các Stage tiếp theo	Không thể triển khai trực tiếp cho môi trường production

Table 5.2: Trade-off của Weak Supervision (Stage 1)

Nhận xét: Weak Supervision vẫn hữu ích khi dùng làm pretraining, giúp mô hình nắm được các quy luật tổng quát trong dữ liệu nguồn. Tuy nhiên, tuyệt đối không triển khai trực tiếp, mà cần Denoising và Supervised Fine-tuning để điều chỉnh nhãn yếu và cải thiện khả năng tổng quát hóa trên dữ liệu thật.

5.2. Hiệu quả Huấn luyện và Phân tích Chiến lược của Các Stage Supervised

Các Stage supervised (Stage 2, Stage 3a, Stage 3b) áp dụng những chiến lược khác nhau để xử lý dữ liệu mất cân bằng, từ hy sinh dữ liệu đến tối ưu Loss Function. Những lựa chọn này dẫn đến khác biệt rõ rệt trong hành vi huấn luyện, tốc độ hội tụ, hiệu suất phân loại và khả năng tổng quát của mô hình.

Thứ nhất, hạn chế cốt lõi của Data-Level Balancing (Stage 2):

Stage 2 sử dụng kỹ thuật undersampling, loại bỏ khoảng 11K mẫu lớp đa số để cân bằng dữ liệu. Những đặc điểm quan trọng:

- **Lợi ích cục bộ:** Mô hình tập trung vào lớp thiểu số, đạt Negative Recall cao nhất trong nhóm supervised (79%).
- **Chi phí về tính toàn vẹn dữ liệu:** Việc giảm mẫu phá vỡ phân phối tự nhiên, làm mất thông tin quan trọng.
- **Suy giảm tổng thể:** Accuracy toàn Stage 2 thấp nhất (79.02%).
- **Mờ decision boundary của lớp Neutral:** Neutral Recall giảm xuống 72%, vì các mẫu near-boundary với lớp Positive/Negative bị loại bỏ, làm mô hình khó phân biệt chính xác.

Kết luận: Stage 2 cải thiện recall lớp thiểu số nhưng đánh đổi nghiêm trọng về khả năng học tổng quát và chất lượng phân loại các lớp khác.

Thứ hai, ưu thế của Algorithm-Level Balancing (Stage 3)

Stage 3a (Focal Loss) và Stage 3b (Class Weighting) áp dụng triết lý *Data Preservation*, giữ nguyên toàn bộ dữ liệu (∼21,8K mẫu), đồng thời điều chỉnh trọng số hoặc cơ chế loss để xử lý mất cân bằng.

- **Tận dụng dữ liệu đầy đủ:** Cả Stage 3a và 3b đạt Accuracy cao nhất (82.35% và 82.22%).
- **Khác biệt về tốc độ hội tụ:** Stage 3a hội tụ nhanh nhất (35.86 phút) nhờ cơ chế Focal Loss triệt tiêu gradient từ các mẫu dễ, tập trung cập nhật mẫu khó; Stage 3b chậm nhất (86.17 phút) do gradient dao động mạnh khi phóng đại trọng số lớp thiểu số.
- **Hành vi gradient và loss:** Focal Loss giảm nhiễu gradient và làm trơn landscape của hàm loss, cho phép mô hình tối ưu ổn định, giữ nguyên cấu trúc dữ liệu và học hiệu quả các mẫu khó mà không overfit.

Thứ ba, phân tích Loss Function và Tác động

Tiêu chí	Stage 2 (CE)	Stage 3b (Class Weighting)	Stage 3a (Focal Loss)
Final Loss	0.5560	0.4895	0.0416
Training Time	52.34 phút	86.17 phút	35.86 phút

Table 5.3: So sánh Loss và thời gian huấn luyện giữa các Stage

- **Cross-Entropy Loss (Stage 2 & 3b):** Giá trị ∼ 0.5–0.49 phản ánh mô hình đã hội tụ ổn định, vượt xa mức ngẫu nhiên (∼ 1.098), đồng thời duy trì khả năng học các đặc trưng phân biệt giữa các lớp.
- **Focal Loss (Stage 3a):** Loss cực thấp (0.0416) nhờ cơ chế *focusing factor* $(1-p_t)^\gamma$, triệt tiêu gradient từ các mẫu dễ, tập trung vào các mẫu khó. Giá trị thấp này không phải overfitting mà là dấu hiệu của quá trình tối ưu hiệu quả.
- **Tốc độ hội tụ liên quan tới Loss:** Gradient dao động mạnh trong Class Weighting làm hội tụ chậm; Focal Loss với gradient tinh gọn dẫn đến tốc độ hội tụ nhanh nhất; CE Loss trung bình cho tốc độ hội tụ trung bình.

Thứ tư, trade-off và khả năng tối ưu

- **Stage 2 (Undersampling):** Trade-off rõ ràng giữa recall lớp thiểu số và accuracy tổng thể; neutral recall giảm, mất thông tin near-boundary, giảm tính tổng quát.
- **Stage 3b (Class Weighting):** Giữ dữ liệu đầy đủ, cải thiện recall nhưng gradient dao động mạnh, hội tụ chậm; thích hợp khi cần ưu tiên lớp critical nhưng không quá quan tâm tốc độ.

- **Stage 3a (Focal Loss):** Cân bằng tối ưu giữa recall, accuracy, tốc độ hội tụ và tổng quát. Tập trung học các mẫu khó mà vẫn giữ cấu trúc phân phối dữ liệu gốc, giảm bias vào lớp đa số, đạt hiệu suất tốt nhất toàn diện.

Thứ năm, tác động trên từng lớp dữ liệu

- **Lớp Negative (thiểu số, critical):** Stage 2 tối ưu recall (79%) nhưng trade-off neutral recall; Stage 3b duy trì recall tốt (73%) nhưng gradient dao động gây cập nhật không ổn định; Stage 3a cân bằng recall và accuracy, học hiệu quả các mẫu khó mà không phá vỡ phân phối gốc.
- **Lớp Neutral (boundary class):** Stage 2 suy giảm (72% recall) do mất các mẫu near-boundary; Stage 3a và 3b duy trì cấu trúc dữ liệu, neutral recall cao hơn, decision boundary rõ ràng.
- **Lớp Positive (đa số):** Stage 2 giảm accuracy vì mất nhiều mẫu; Stage 3a và 3b giữ toàn bộ dữ liệu, duy trì khả năng phân loại Positive chính xác, cải thiện tổng thể accuracy.

Thứ sáu, phân tích lỗi và nguyên nhân

- **Lỗi ngữ cảnh (Pragmatic Failures):** Xuất hiện ở sarcasm/irony (False Negative) hoặc hài hước/trích dẫn (False Positive); nguyên nhân do RoBERTa-base chưa có khả năng inference ngữ dụng tinh vi.
- **Lỗi do thay đổi cấu trúc dữ liệu (Stage 2):** Loại bỏ dữ liệu near-boundary phá vỡ distribution, mờ decision boundary, đặc biệt giữa Neutral và Positive.
- **Lỗi gradient dao động (Stage 3b):** Class Weighting phóng đại gradient lớp thiểu số, gây noisy updates, khó hội tụ ổn định, đôi khi overshoot boundary quan trọng.

Kết luận tổng thể

- **Stage 2**: Ưu tiên recall lớp thiểu số, đánh đổi accuracy tổng thể và khả năng học tổng quát.
- **Stage 3b**: Giữ dữ liệu, cải thiện recall nhưng hội tụ chậm do gradient dao động.
- **Stage 3a (Focal Loss)**: Cân bằng tối ưu giữa recall, accuracy, tốc độ hội tụ và khả năng tổng quát; giữ nguyên cấu trúc dữ liệu, tập trung học mẫu khó.

Focal Loss là lựa chọn ưu tiên khi huấn luyện trên dữ liệu mất cân bằng, đặc biệt khi cần duy trì cả hiệu suất phân loại và khả năng tổng quát của mô hình.

5.3. Kết luận

Tổng hợp từ các phân tích trên, có thể rút ra những nhận định chính:

- **Weak Supervision**: Chỉ phù hợp cho giai đoạn pretraining; sử dụng trực tiếp cho supervised learning gặp vấn đề do Domain Shift lớn, dẫn đến hiệu quả hạn chế.

- **Cross-Entropy (CE)**: Cung cấp baseline ổn định, nhưng khi dữ liệu mất cân bằng, CE không đủ khả năng tập trung học các mẫu khó, hạn chế recall cho lớp thiểu số.
- **Class Weighting (Stage 3b)**: Cải thiện một phần hiệu quả trên lớp thiểu số, nhưng tạo ra gradient dao động mạnh, làm quá trình hội tụ chậm và thiếu ổn định.
- **Focal Loss (Stage 3a)**: Thể hiện hiệu quả tối ưu trong mọi tiêu chí:
  - Accuracy cao nhất trong nhóm supervised.
  - Thời gian huấn luyện nhanh nhất nhờ gradient triệt tiêu từ các mẫu dễ.
  - Giữ nguyên phân phối dữ liệu gốc, tránh bias và overfitting.

Tuy vậy, mô hình vẫn gặp khó khăn với những dạng ngữ cảnh phức tạp. Cụ thể, những câu mang sắc thái sarcasm hoặc irony thường bị dự đoán sai, các đoạn đối thoại đa câu đòi hỏi inference liên câu vẫn vượt quá khả năng hiện tại của mô hình, và những câu quote hoặc trích dẫn chứa ý nghĩa ẩn cần khả năng *pragmatic inference* mà RoBERTa-base chưa thể thực hiện hiệu quả.

Như vậy, Stage 3a (Focal Loss) là lựa chọn tối ưu cho supervised learning trên dữ liệu mất cân bằng, nhưng việc cải thiện khả năng xử lý ngữ cảnh tinh vi vẫn cần các phương pháp bổ trợ, như *active learning* hoặc *teacher models* nâng cao.

Chương 6

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1. Kết luận

Đề tài đã xây dựng một pipeline huấn luyện gồm bốn Stage, nhằm phân tích toàn diện tác động của các chiến lược xử lý mất cân bằng dữ liệu trong bài toán phân loại cảm xúc trên tập 21.8k bình luận về game – vốn có phân bố cảm xúc lệch mạnh theo tự nhiên. Việc tổ chức pipeline theo từng Stage độc lập cho phép so sánh có hệ thống giữa các hướng tiếp cận: từ nhãn yếu dựa trên tín hiệu heuristic, các phương pháp cân bằng ở mức dữ liệu, cho đến những kỹ thuật cân bằng ở mức hàm mất mát. Nhờ đó, đề tài không chỉ đánh giá hiệu năng mà còn đưa ra nhận định về tính ổn định, khả năng học ngữ nghĩa và mức độ phù hợp của từng chiến lược trong bối cảnh dữ liệu ngôn ngữ.

Weak Supervision (Stage 1)

Stage 1 cho thấy ưu điểm rõ rệt của Weak Supervision nằm ở khả năng tạo nhãn nhanh và chi phí thấp nhờ cơ chế tự động hóa với 8-Signal Strategy áp dụng trên dữ liệu Reddit. Cách tiếp cận này giúp mở rộng quy mô dữ liệu huấn luyện mà không cần gán nhãn thủ công, đồng thời cho phép mô hình học được các tín hiệu cảm xúc phổ biến trong cộng đồng người dùng.

Tuy vậy, khi đánh giá trên tập Kaggle chuẩn, mô hình giảm gần 39% Accuracy, cho thấy Domain Shift mạnh giữa Reddit và miền mục tiêu. Các nhãn được sinh từ heuristic Reddit mang theo nhiều bias ngữ cảnh—bao gồm slang, lối nói ẩn ý, và văn hoá cộng đồng đặc thù—khiến mô hình học sai quy luật và không thể tổng quát hoá sang dữ liệu có phong cách trung tính hơn.

Bên cạnh đó, các tín hiệu heuristic chỉ mô phỏng một phần cấu trúc cảm xúc thực và không đủ tinh vi để mô tả các trường hợp khó như *sarcasm* hoặc câu đa nghĩa. Kết quả Stage 1 vì vậy chủ yếu phản ánh chất lượng và giới hạn của nhãn yếu, thay vì đóng vai trò tối ưu hiệu năng cuối. Weak Supervision vẫn hữu ích để mở rộng dữ liệu và khảo sát xu hướng, nhưng không thể thay thế nhãn chất lượng cao trong các giai đoạn huấn luyện quyết định.

Supervised Learning (Stage 2 & 3)

Trái với Stage 1, các Stage Supervised sử dụng nhãn chuẩn từ Kaggle, cho phép đánh giá mô

hình một cách đáng tin cậy hơn và phản ánh chính xác tác động của từng kỹ thuật cân bằng. Nhờ dữ liệu được gán nhãn thủ công với chất lượng cao, mô hình có thể học được phân bố ngữ nghĩa tự nhiên của bài toán, từ đó cho phép quan sát rõ ràng hơn hiệu quả và giới hạn của từng phương pháp.

- Stage 2 – Undersampling

  Undersampling giúp tăng Recall của lớp thiểu số, đặc biệt với các câu chứa cảm xúc Positive hoặc Negative hiếm gặp. Tuy nhiên, lợi ích này đi kèm với chi phí lớn: việc loại bỏ hàng nghìn mẫu của lớp đa số khiến mô hình mất nhiều thông tin quan trọng nằm gần biên quyết định (decision boundary).

  Trong bài toán xử lý ngôn ngữ tự nhiên, các mẫu near-boundary thường chứa cấu trúc ngữ nghĩa phức tạp, mơ hồ hoặc đa nghĩa. Đây là những tín hiệu then chốt giúp mô hình phân biệt cảm xúc tinh tế giữa Neutral và Positive/Negative. Khi mất các mẫu này, không gian biểu diễn bị méo, mô hình học lệch và dễ overfit vào phần dữ liệu còn lại.

  Kết quả cho thấy Accuracy tổng thể giảm, tốc độ hội tụ kém và hiệu năng dao động mạnh qua các epoch. Stage 2 vì vậy minh họa rõ rằng các kỹ thuật cân bằng ở mức dữ liệu – mặc dù đơn giản và dễ áp dụng – lại không phù hợp cho các mô hình ngôn ngữ tiền huấn luyện, vốn phụ thuộc nhiều vào việc giữ nguyên phân bố ngữ nghĩa đầu vào.

- Stage 3a – Focal Loss

  Focal Loss thể hiện hiệu năng toàn diện và ổn định nhất trong toàn bộ pipeline, đạt 82.35% Accuracy, đồng thời hội tụ nhanh và giảm mạnh dao động so với Class Weighting.

  Điểm mạnh then chốt của Focal Loss nằm ở cơ chế điều chỉnh trọng số động theo độ khó của mẫu. Các mẫu dễ – thường là lớp đa số – được giảm trọng số, còn các mẫu khó hoặc ít xuất hiện được ưu tiên trong quá trình cập nhật tham số. Điều này làm giảm sự lấn át của lớp đa số mà không phải loại bỏ dữ liệu, giúp mô hình bảo toàn đầy đủ ngữ nghĩa và học ranh giới phân lớp mượt hơn.

  Nhờ kết hợp giữa "tập trung vào mẫu khó" và "giữ nguyên toàn bộ dữ liệu đầu vào", Focal Loss vượt trội hơn cả Undersampling và Class Weighting trong điều kiện mất cân bằng nặng. Việc mô hình giữ được độ ổn định qua nhiều lần chạy chứng minh rằng phương pháp này vừa hiệu quả vừa đáng tin cậy.

- Stage 3b – Class Weighting

  Class Weighting là một kỹ thuật cân bằng ở mức thuật toán, tăng trọng số cho lớp thiểu số nhằm giảm xu hướng dự đoán lệch về lớp đa số.

  Tuy nhiên, Class Weighting thể hiện mức ổn định kém hơn Focal Loss, đặc biệt khi phân bố cảm xúc lệch mạnh (Positive chỉ khoảng 30%). Mô hình có xu hướng dao động giữa Precision và Recall, cho thấy ranh giới phân lớp dễ bị dịch chuyển theo hàm loss. Ngoài ra, tốc độ hội tụ chậm hơn khiến việc huấn luyện cần nhiều epoch hơn để đạt hiệu năng tối ưu.

  Mặc dù tốt hơn Undersampling và vẫn phù hợp với các bài toán mất cân bằng vừa phải, Class Weighting bộc lộ giới hạn rõ rệt khi dữ liệu chứa nhiều câu phức tạp hoặc sắc thái ngữ nghĩa mơ hồ – những tình huống vốn đòi hỏi mô hình xử lý *fine-grained context* tốt hơn.

Giới hạn chung của mô hình

Dù đã áp dụng nhiều kỹ thuật cân bằng, mô hình vẫn gặp hạn chế rõ rệt với các dạng ngôn ngữ phức tạp. Đầu tiên, *sarcasm* và *irony* vẫn là điểm yếu cố hữu vì chúng đòi hỏi mô hình phải nhận diện sự lệch giữa nghĩa đen và nghĩa ngầm — một dạng suy luận nằm ngoài phạm vi token-level của RoBERTa-base.

Bên cạnh đó, mô hình khó xử lý *pragmatic inference* và các bình luận mang tính văn hoá cộng đồng (meme, ẩn ý, shorthand), vốn yêu cầu kiến thức nền và khả năng suy luận ngữ dụng mà mô hình encoder-base không có.

Đối với các câu chứa trích dẫn đa tầng nghĩa hoặc hội thoại dạng "quote + reply", mô hình thường xác định sai đối tượng bị nhận xét do không có cơ chế mô hình hóa discourse hoặc theo dõi ngữ cảnh liên câu.

Cuối cùng, trong hội thoại đa câu, đặc biệt khi người dùng thay đổi thái độ qua từng câu, RoBERTa-base khó nắm bắt quan hệ liên câu (contrast, concession, shift in stance), khiến phân loại cảm xúc thiếu ổn định.

Những giới hạn này xuất phát chủ yếu từ kiến trúc RoBERTa-base — tối ưu cho phân loại trực tiếp, nhưng không đủ mạnh cho các nhiệm vụ đòi hỏi reasoning, ngữ dụng sâu, hoặc phân tích ngữ cảnh mở rộng.

Kết luận tổng thể

Từ toàn bộ kết quả thực nghiệm, Focal Loss (Stage 3a) nổi bật là phương pháp mang lại hiệu năng cao nhất, độ ổn định tốt, và đặc biệt phù hợp với bài toán phân loại cảm xúc có phân bố lệch nặng trong miền Gaming. Cơ chế tập trung vào mẫu khó giúp mô hình vừa duy trì độ chính xác tổng thể, vừa cải thiện khả năng nhận diện lớp thiểu số mà không cần can thiệp trực tiếp vào dữ liệu.

Ngược lại, Weak Supervision và Undersampling tuy không thể sử dụng như chiến lược huấn luyện chính, nhưng vẫn đóng vai trò quan trọng trong việc mở rộng góc nhìn về pipeline: Weak Supervision cho thấy giới hạn của nhãn yếu và tác động của Domain Shift; còn Undersampling minh họa rõ ràng rủi ro mất mát thông tin ngữ nghĩa khi can thiệp vào phân bố dữ liệu.

Tổng thể, kết quả củng cố nhận định rằng các phương pháp cân bằng ở mức thuật toán (loss-level) — vốn bảo toàn toàn bộ phân bố ngữ nghĩa tự nhiên — mang lại sự ổn định và khả năng tổng quát tốt hơn đáng kể so với các chiến lược thao tác dữ liệu. Điều này đặc biệt quan trọng trong các bài toán ngôn ngữ, nơi mỗi mẫu đều chứa nhiều tín hiệu ngữ dụng mà mô hình cần giữ lại để đạt tối ưu.

6.2. Hướng phát triển

Dựa trên các hạn chế đã được nhận diện, đề tài đề xuất một số định hướng mở rộng nhằm cải thiện chất lượng nhãn, tăng khả năng tổng quát hóa và nâng cao hiệu quả của mô hình trong các bối cảnh phức tạp. Mặc dù cả Stage 1 và tập Kaggle đều lấy dữ liệu từ Reddit, sự khác biệt nằm ở mức độ nhiễu và chất lượng gán nhãn: tập Kaggle được tuyển chọn và kiểm duyệt thủ công, trong khi Stage 1 sử dụng Reddit raw cùng heuristic nên chứa nhiều noise và bias.

Ứng dụng Large Language Models (LLM) làm Teacher Model

Thay vì phụ thuộc hoàn toàn vào hệ thống heuristic từ Reddit, các mô hình ngôn ngữ lớn như GPT-4 hoặc Llama-3 có thể được sử dụng như một "Teacher Model" để:

- Tinh lọc và giảm nhiễu cho nhãn yếu, hạn chế các dạng bias đặc thù cộng đồng.
- Sinh pseudo-label chất lượng cao cho các mẫu khó, bao gồm sarcasm, câu đa nghĩa hoặc cấu trúc ngữ dụng phức tạp.

Cách tiếp cận LLM-as-Teacher cho phép kết hợp lợi thế của Weak Supervision (tốc độ, chi phí thấp) với độ chính xác của Supervised Learning, đặc biệt hiệu quả khi kích thước dữ liệu lớn nhưng nguồn lực gán nhãn hạn chế.

Tận dụng Active Learning để tối ưu chi phí gán nhãn

Active Learning giúp mô hình chủ động lựa chọn các mẫu mà nó chưa chắc chắn để đưa tới người gán nhãn. Điều này giúp giảm đáng kể chi phí so với việc gán nhãn toàn bộ dữ liệu, đồng thời vẫn cải thiện rõ rệt hiệu năng. Phương pháp này phù hợp với bài toán sentiment, nơi nhiều câu mang sắc thái trung tính hoặc ẩn ý khó nhận diện.

Mở rộng sang phân tích cảm xúc đa chiều (Emotion Analysis)

Thay vì ba lớp Positive–Neutral–Negative, mô hình có thể được mở rộng để nhận diện các cảm xúc chi tiết hơn như *joy*, *anger*, *frustration*, *excitement*. Điều này tăng giá trị ứng dụng thực tế, đặc biệt trong các hệ thống phản hồi người chơi, phân tích trải nghiệm game, hoặc các chatbot trong lĩnh vực Gaming.

Tích hợp mô hình hiểu ngữ cảnh nâng cao

Để khắc phục hạn chế trong xử lý sarcasm, irony và hội thoại đa lượt, mô hình có thể chuyển sang các kiến trúc mạnh hơn như T5, BART hoặc các LLM được tinh chỉnh chuyên biệt. Các kiến trúc này cho phép học quan hệ liên câu, hiểu ý định và suy luận ngữ dụng tốt hơn so với mô hình encoder-base như RoBERTa.

Đa dạng hóa nguồn dữ liệu để giảm Domain Bias

Dù đều xuất phát từ Reddit, tập Kaggle mang phong cách được tuyển chọn trong khi Reddit raw rất đa dạng về chủ đề và văn phong. Việc mở rộng thêm dữ liệu từ Facebook, TikTok, review game hoặc hội thoại thực giúp mô hình tiếp xúc với nhiều dạng biểu đạt khác nhau. Điều này làm giảm rủi ro Domain Shift và tăng khả năng tổng quát hóa sang các bối cảnh ngoài Reddit và Kaggle.

Tài liệu tham khảo

[1] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

[2] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[3] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," *The VLDB Journal*, vol. 29, no. 2, pp. 709–730, 2017.

[4] B. Liu, *Sentiment analysis and opinion mining*. Morgan & Claypool Publishers, 2012.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.