HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION COMMUNICATION TECHNOLOGY

PROJECT REPORT:

Sentiment Analysis for Video Game Reviews

***Supervised by*:**

Ph.D. Nguyen Kiem Hieu

***Presented by*:**

Au Trung Phong - 20225455

Nguyen Nam Khanh - 20225448

Vu Duc Minh - 20225514

Tran Sy Minh Quan - 20225521

Bui Van Huy - 20225497

Hanoi - Vietnam

2025

Contents

1 Introduction

1.1 Context and Motivation

The video game industry has evolved into a dominant force in global entertainment, surpassing both the film and music industries in revenue. According to recent market intelligence by Newzoo, the global games market is projected to generate approximately $187.7 billion in 2024, with the number of active players worldwide reaching nearly 3.4 billion [9]. Unlike passive forms of media, gaming is deeply interactive, fostering passionate communities that generate massive volumes of feedback across social media, forums, and review aggregators.

This explosion of User-Generated Content (UGC) has fundamentally shifted how games are marketed and consumed. Research indicates that up to 93% of consumers find UGC helpful when making purchasing decisions, and approximately 70% of gamers consult reviews or ratings before buying a title [11]. Platforms like Metacritic have emerged as central hubs for this ecosystem, where a game's aggregate score (Metascore) can determine its commercial success, influence stock prices, and even dictate the financial bonuses of development teams [15].

However, the sheer velocity and volume of this data present a significant challenge. A single popular game release can generate tens of thousands of reviews in a matter of days. These reviews range from detailed, multi-paragraph critiques to slang-filled one-liners (e.g., "This game is buggy but the combat is fire"). Manual analysis of this unstructured text is impossible at scale. Consequently, automated "Opinion Mining"—or Sentiment Analysis—has transitioned from an academic curiosity to an industrial necessity. It allows developers and publishers to monitor brand health, identify critical bugs, and quantify player satisfaction in real-time.

1.2 Problem Statement

The core technical problem addressed in this project is the automated classification of sentiment in unstructured game reviews. While general sentiment analysis is a well-studied field, the gaming domain presents unique linguistic challenges that general-purpose models often fail to capture.

Formal Problem Definition Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ of N game reviews, where each review x_i is a sequence of tokens $x_i = (w_1, w_2, \ldots, w_{|x_i|})$ and $y_i \in \mathcal{Y}$ is its corresponding sentiment label from the set $\mathcal{Y} = \{\text{Positive}, \text{Mixed}, \text{Negative}\}$ with $|\mathcal{Y}| = 3$, the objective is to learn a classification function:

$$f : \mathcal{X} \to \mathcal{Y}, \quad \text{where } |\mathcal{Y}| = 3 \tag{1}$$

that maps an arbitrary review text $x \in \mathcal{X}$ to one of the three sentiment classes $y \in \mathcal{Y}$, minimizing the expected classification error:

$$\mathcal{L}(f) = \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\mathbb{1}(f(x) \neq y)\right] \tag{2}$$

where $\mathbb{1}(\cdot)$ is the indicator function that equals 1 when the condition is true and 0 otherwise.

Problem Characteristics

- **Input:** A dataset of natural language reviews collected from online gaming platforms. These inputs are characterized by high variance in length, informal grammar, and heavy usage of domain-specific terminology (e.g., "nerf," "buff," "grindy," "framerate," "pay-to-win").
- **Output:** A predicted sentiment label for each review (Positive, Mixed, Negative).
- **Key Challenges:**
    1. **Domain Adaptation:** Standard NLP models trained on formal corpora (like Wikipedia) often misinterpret gaming slang. For instance, the word "sick" is positive in a gaming context, whereas "broken" can refer to a bug (negative) or an overpowered mechanic (context-dependent).
    2. **Data Quality:** User-generated game reviews are often polarized or sarcastic, complicating sentiment detection.
    3. **Scalability:** The solution must be efficient enough to process large datasets without excessive

computational overhead.

1.3 Project Objectives

The primary goal of this capstone project is to develop a comprehensive pipeline for gaming sentiment analysis, spanning from raw data collection through model evaluation and public release. The specific objectives are defined as follows:

1. **Dataset Construction:** To create a valuable benchmark dataset by scraping, aggregating, and cleaning thousands of reviews from Metacritic. This involves using advanced web scraping tools to handle dynamic content and filtering specifically for English-language feedback.
2. **Methodological Comparison:** To implement and strictly evaluate three distinct modeling approaches to understand the trade-off between complexity and performance:
    - **Baseline Model (LSTM):** Implementing a Long Short-Term Memory network with a standard trainable embedding layer. This serves as a conventional baseline to measure performance improvements without reliance on pre-trained external vectors.
    - **Fine-tuning Large Language Model Approach (RoBERTa):** Fine-tuning a RoBERTa transformer model to leverage deep contextual understanding and attention mechanisms.
    - **Dense Embedding with Machine Learning Approach (EmbeddingGemma + XGBoost):** Proposing a modern approach that combines state-of-the-art dense vector representations (EmbeddingGemma-300M) with gradient boosting (XGBoost) to achieve high accuracy with lower training costs.
3. **Performance Analysis and Model Release:** To provide a comprehensive evaluation using standard metrics including per-class and aggregate measures (Accuracy, Precision, Recall, F1-Score, Confusion Matrix), along with efficiency metrics (Training Time, Throughput) to analyze which architecture offers the best performance for the specific nuances of game reviews while maintaining practical computational feasibility. Additionally, to release all trained models and code artifacts to the HuggingFace Hub for reproducibility, accessibility, and potential integration into downstream applications.

2 Background

2.1 Sentiment Analysis in Domain-Specific Contexts

Sentiment Analysis (SA), or opinion mining, is a pivotal task in Natural Language Processing (NLP) that aims to determine the polarity of subjective text. While general SA models have achieved significant success on standard datasets like IMDb (movies) or Yelp (restaurants), they often struggle when applied to the gaming domain due to its unique linguistic characteristics.

One major challenge in gaming reviews is the high prevalence of domain-specific slang and evolving terminology. Research by Panwar and Bhatnagar [10] highlights that generic sentiment lexicons fail to

capture the nuanced meaning of terms like “frag” (kill), “grind” (repetitive gameplay), or “camp” (waiting for enemies), which can be neutral or negative depending on context. Furthermore, gaming communities frequently employ sarcasm and irony, which remain persistent hurdles for traditional machine learning models. A survey by Hussein [6] identifies sarcasm detection as a critical bottleneck, noting that gamers often use ostensibly positive words (e.g., “Great job crashing every 5 minutes”) to express severe dissatisfaction.

Another significant issue is the structural complexity of game reviews. Unlike product reviews that focus on a single tangible object, game reviews often discuss multiple heterogeneous aspects such as graphics, gameplay mechanics, story, and technical performance. Viggiato et al. [14] demonstrated that off-the-shelf sentiment classifiers trained on non-gaming data perform poorly on game reviews (achieving AUC scores as low as 0.53), largely because they cannot distinguish between sentiment directed at a specific buggy feature versus the overall game experience. This necessitates the development of domain-adapted models capable of understanding these specific contexts.

2.2 Recurrent Neural Networks and Long Short-Term Memory

Recurrent Neural Networks (RNNs) are designed to process sequential data but often struggle with the vanishing gradient problem, limiting their ability to learn long-term dependencies in text. To address this, Hochreiter and Schmidhuber [5] introduced Long Short-Term Memory (LSTM) networks, which utilize gating mechanisms to selectively retain or discard information over long sequences.

The information flow within an LSTM cell is regulated by the following transition functions:

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) && \text{(Forget Gate)} \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) && \text{(Input Gate)} \\
C_t &= f_t \odot C_{t-1} + i_t \odot \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) && \text{(Cell State)} \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) && \text{(Output Gate)} \\
h_t &= o_t \odot \tanh(C_t) && \text{(Hidden State)}
\end{aligned}
\tag{3}
$$

where σ denotes the sigmoid function and $\odot$ denotes element-wise multiplication.

To capture context from both preceding and succeeding words, this project utilizes a Bidirectional LSTM (BiLSTM), which processes input sequences in both directions. The specific architectural implementation for this study is adapted from Dhanjal and Sinha [8], who demonstrated the efficacy of BiLSTM networks for sentiment classification tasks.

2.3 Large Language Models (LLMs) and Fine-tuning Strategies

While LSTMs improved sequential modeling, they remain constrained by their sequential nature, which precludes parallelization and limits the retention of context in very long sequences. The introduction of the Transformer architecture by Vaswani et al. [12] revolutionized NLP by utilizing the **Self-Attention Mechanism**, allowing models to weigh the importance of all words in a sentence simultaneously, regard-

less of their distance from each other.

2.3.1 RoBERTa Architecture

For this project, we utilize **RoBERTa** (Robustly Optimized BERT Pretraining Approach), introduced by Liu et al. [7]. RoBERTa is an encoder-only Transformer model that builds upon BERT's architecture but introduces critical optimizations that make it highly effective for complex sentiment analysis:

- **Dynamic Masking:** Unlike BERT, which masks tokens once during data preprocessing, RoBERTa generates a unique masking pattern for every training sequence, effectively augmenting the data and preventing the model from overfitting to specific masking patterns.
- **Removal of Next Sentence Prediction (NSP):** Research indicated that the NSP objective used in BERT was unnecessary for performance; removing it allows RoBERTa to focus entirely on the Masked Language Modeling (MLM) objective.
- **Larger Contexts and Batches:** RoBERTa is trained on significantly larger mini-batches and more diverse datasets, resulting in strictly better performance on downstream tasks compared to the original BERT.

2.3.2 Fine-Tuning Strategy

Instead of training a Transformer from scratch—which requires massive computational resources—we employ **Transfer Learning**. We initialize the model with pre-trained weights learned from a massive corpus and fine-tune it on our game review dataset.

Mathematically, given the final hidden vector h_{CLS} corresponding to the special classification token (representing the entire sentence), the probability distribution over the sentiment classes y is computed by passing h_{CLS} through a linear classification head:

$$P(y|x) = \text{softmax}(W_{task} \cdot h_{CLS} + b_{task}) \tag{4}$$

where W_{task} and b_{task} are the parameters specific to the sentiment classification layer that are updated during fine-tuning, while the underlying RoBERTa weights are fine-tuned at a lower learning rate.

2.4 Dense Vector Representations and Gradient Boosting

While end-to-end deep learning models like RoBERTa are powerful, they are computationally expensive. An alternative approach is to decouple the *understanding* of text from the *classification* task. This involves using a pre-trained Large Language Model purely as a feature extractor to generate high-quality "embeddings," which are then fed into a highly efficient structured data classifier.

2.4.1 Semantic Embeddings via EmbeddingGemma

Text embeddings map discrete text sequences into a continuous vector space

$$\mathbb{R}^d$$

, where d is the dimension of the embedding (typically $d = 768$ for this model). In this space, semantic similarity is preserved as geometric proximity; reviews with similar meanings are located closer together.

For this project, we utilize **EmbeddingGemma (google/embeddinggemma-300m)** [13]. This model is a compact, 300-million-parameter embedding model derived from the Gemma 3 family. Unlike larger monolithic models, EmbeddingGemma is optimized for resource-constrained environments while maintaining state-of-the-art performance. It employs contrastive learning and Matryoshka Representation Learning to produce robust dense vectors across over 100 languages.

We selected this model for two primary reasons:

1. **Resource Efficiency:** With only 300M parameters, the model fits comfortably within the memory constraints of the Kaggle environment (specifically the NVIDIA Tesla P100 16GB GPU), allowing for larger batch sizes and fast inference.
2. **High Performance:** Despite its small size, it achieves competitive rankings on the Massive Text Embedding Benchmark (MTEB), ranking **11th in the Borda count** and **16th in Classification tasks** [2].

Given an input sentence x, the embedding model E produces a fixed-size vector v:

$$v = E(x), \quad v \in \mathbb{R}^d \tag{5}$$

This vector v serves as a dense numerical representation of the review's semantic content, capturing nuances that simple Bag-of-Words models miss.

2.4.2 Gradient Boosting with XGBoost

Once the text is converted into dense vectors, the sentiment analysis task becomes a standard tabular classification problem. We employ **XGBoost (eXtreme Gradient Boosting)** [1], an ensemble learning method based on decision trees.

XGBoost constructs a strong classifier by sequentially adding "weak" learners (decision trees). Each new tree is trained to correct the residual errors made by the previous ensemble of trees. The prediction $\hat{y}_i$ for a given instance i at step K is the sum of predictions from K trees:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(v_i), \quad f_k \in \mathcal{F} \tag{6}$$

where $\mathcal{F}$ is the space of regression trees. XGBoost is particularly chosen for its ability to handle dense numerical features effectively, its regularization techniques to prevent overfitting, and its computational efficiency compared to fine-tuning massive Transformers.

3 Methodology

3.1 Phase 1: Data Acquisition

The foundation of this project is the construction of a novel, domain-specific dataset derived from Metacritic, a leading review aggregator for the gaming industry. The data acquisition process was designed to be robust, scalable, and capable of handling dynamic web content.

3.1.1 Game Discovery Module

The first step involved identifying a diverse list of games to ensure the dataset covers various genres, platforms, and time periods. We implemented a custom Python script, `discover_games.py`, which employs standard HTTP requests and BeautifulSoup for HTML parsing. This script iterates through Metacritic's game listings, utilizing pagination handling to traverse multiple pages of search results, extracting key metadata for each game:

- **Game Title:** The canonical name of the video game.
- **Platform:** The specific console or system (e.g., PlayStation 5, PC, Xbox Series X) to account for performance differences that might affect sentiment.
- **Metascore:** The aggregate critic score, collected for dataset completeness.
- **Release Date:** The game's original release date for temporal analysis.
- **URL:** The direct link to the game's user review page.

The output of this module is a structured list of game URLs stored in `data/discovered_games/`, serving as the input queue for the review scraper.

3.1.2 Review Scraping Engine

To extract the actual review content, we developed a sophisticated scraping engine using **Scrapy**, a high-level web crawling framework, integrated with **Playwright**. This combination was essential because Metacritic relies heavily on JavaScript to render user reviews dynamically. Standard HTTP requests (used by libraries like `requests` or `BeautifulSoup`) would fail to load the full content.

The spider, located in `metacritic_scraper/spiders/`, operates as follows:

1. **Request Handling:** It accepts the list of discovered game URLs and initiates asynchronous requests with configurable concurrency (`CONCURRENT_REQUESTS = 2-4`).
2. **Dynamic Rendering:** Playwright launches a headless browser instance to render the DOM (Document Object Model) fully, triggering the necessary JavaScript events to load all reviews.
3. **Platform Discovery:** The spider automatically detects all available platforms for multi-platform games by parsing the platform filter dropdown, then scrapes reviews for each platform separately to ensure comprehensive coverage.
4. **Progressive Loading:** To load all available reviews, Playwright automatically scrolls through the page and clicks "Load More" buttons, with timeout protection (300 seconds per platform) and du-

plicate detection to prevent infinite loops.

5. **Data Extraction:** The spider parses the rendered HTML to extract comprehensive review and game metadata:
    - **Review Text:** The body of the user's comment.
    - **User Score:** The numerical rating (0-10 scale, normalized to 0-100 for consistency).
    - **Review Category:** Extracted directly from Metacritic's color-coded sentiment badges—positive (green, $\geq$75), mixed (yellow, 50-74), or negative (red, $<$50)—which represent Metacritic's domain-expert classification of review sentiment.
    - **Review Platform:** The specific platform for which the review was written, extracted at the individual review level.
    - **Date:** The submission date of the review.
    - **Username:** Identifier for the reviewer.
    - **Game Metadata:** Title, platform, metascore, release date, genre, developer, and publisher for contextual analysis.
6. **Rate Limiting:** To adhere to ethical scraping guidelines and avoid IP bans, we configured `DOWNLOAD_DELAY = 3` seconds, enabled auto-throttling (`AUTOTHROTTLE_ENABLED = True`), and limited concurrent requests in both the spider's custom settings and `settings.py`, ensuring a respectful request pace that adapts to server response times.

The raw output is saved as JSON Lines (`.jsonl`) files in the `data/review_data/` directory, with each review stored as a separate JSON object on its own line, facilitating efficient line-by-line processing in subsequent stages.

3.2 Phase 2: Data Preparation

Raw web data is inherently noisy and unstructured. Phase 2 focused on transforming the raw collection into a clean, machine-learning-ready benchmark dataset.

3.2.1 Aggregation and Cleaning

We developed an aggregation pipeline (`aggregate_dataset.py`) to consolidate the scattered JSONL files from Phase 1. Key preprocessing steps included:

- **Language Filtering:** Since the scope of this project is English sentiment analysis, we applied the `langdetect` library to identify and retain only English reviews. While individual reviews may contain occasional non-English words (e.g., game titles or character names), the library's n-gram-based detection ensures the overall text is predominantly English, maintaining vocabulary consistency for the embedding layers and preventing multilingual interference.
- **Deduplication:** Duplicate reviews (e.g., spam or multiple scrapes of the same page) were identified via normalized text comparison and removed to prevent data leakage between training and test sets. The system tracks unique reviews using lowercase, trimmed text as a hash key.

- **Field Selection:** From the rich metadata collected during scraping, we extracted only the essential fields for sentiment analysis: review text and review category. User scores, while collected as part of comprehensive dataset construction, were not utilized in the classification task but remain available for potential future correlation studies or multi-task learning experiments.

Note that text normalization (whitespace standardization) was performed during the scraping phase itself, ensuring clean input to the aggregation pipeline.

3.2.2 Sentiment Labels

Sentiment labels were directly extracted from Metacritic's platform during Phase 1. Metacritic employs a three-tier classification system that color-codes user reviews based on their scores:

- **Positive:** Reviews marked with green badges.
- **Mixed:** Reviews marked with yellow badges, representing reviews with balanced or ambivalent sentiment.
- **Negative:** Reviews marked with red badges.

These pre-labeled categories were scraped directly from the HTML structure, eliminating the need for manual annotation or score-based heuristics. This approach ensures that our labels align precisely with Metacritic's domain expertise in game review sentiment classification. All three categories were retained in the final dataset to enable both binary (positive vs. negative) and ternary (including mixed) classification experiments.

3.2.3 Dataset Splitting

To guarantee rigorous evaluation, the processed dataset was deterministically split using a fixed random seed (`random.seed(42)`) into three subsets:

- **Training Set (80%):** Used for model parameter updates during backpropagation.
- **Validation Set (10%):** Used for hyperparameter tuning, early stopping, and model selection during training.
- **Test Set (10%):** Held out completely until the final evaluation phase to report unbiased performance metrics.

The splitting was performed after shuffling to ensure random distribution across all sentiment categories and prevent temporal or game-specific biases. The final dataset was formatted as a HuggingFace `DatasetDict` and uploaded to the HuggingFace Hub using `prepare_and_upload_hf_dataset.py`, enabling seamless integration with modern NLP libraries such as Transformers and Datasets.

3.3 Phase 3: Model Training

The final phase involves the development and evaluation of three distinct sentiment classification models. The training pipeline implements a rigorous two-phase strategy to ensure optimal hyperparameter selection and unbiased performance evaluation.

Two-Phase Training Strategy: All models follow a systematic approach to hyperparameter optimization and final model training, ensuring reproducible and unbiased performance assessment.

Phase 1: Grid Search (Hyperparameter Exploration) The first phase conducts exhaustive hyperparameter search using a subset of the training data to efficiently identify optimal model configurations:

- **Data Partitioning:** Grid search uses only training and validation splits (typically 10% for computational efficiency). The test set remains completely untouched to prevent information leakage.
- **Evaluation Protocol:** Models are trained on the training subset and evaluated exclusively on the validation set using F1-macro score as the primary optimization metric, ensuring equal importance for all sentiment categories (positive, mixed, negative).
- **Hyperparameter Space:** Each model explores its relevant hyperparameter space—learning rates, regularization strengths, architectural choices—with results automatically saved to `best_config.txt`. These exploration models are not uploaded to HuggingFace Hub.
- **Output:** The configuration achieving the highest validation F1-macro score is identified and logged, along with comprehensive grid search results in `gridsearch_results.json`.

Phase 2: Final Training (Production-Ready Model) Using the optimal hyperparameters discovered in Phase 1, the second phase trains the final production-ready model:

- **Full Dataset Training:** The model is retrained on 100% of the training data (not the 10% subset) with the best hyperparameters, maximizing the model's exposure to training examples.
- **Test Set Evaluation:** The held-out test set is evaluated for the **first and only time** during this phase, providing unbiased performance metrics that accurately reflect the model's generalization capability on truly unseen data.
- **Model Release:** The final trained model, along with all necessary artifacts (tokenizers, label encoders, configuration files, and training logs), is saved locally and uploaded to the HuggingFace Hub for reproducibility, sharing, and potential integration into downstream applications or further development.

This two-phase methodology ensures that hyperparameter selection is both data-driven and computationally efficient, while final performance reporting maintains scientific rigor through strict test set isolation.

3.3.1 Baseline Model: Bidirectional LSTM

To establish a robust benchmark for sentiment analysis without relying on external pre-trained models, we constructed a deep learning architecture based on the Bidirectional Long Short-Term Memory (Bi-LSTM) network. As described in [8], this architecture extends the traditional RNN by processing sequence data in both forward and backward directions, allowing the model to capture context from both past and future states, thereby effectively handling long-term dependencies in text.

Data Preprocessing and Tokenization The text data is processed using a custom `SimpleTokenizer` designed to build a domain-specific vocabulary directly from the training corpus.

- **Vocabulary Construction:** The vocabulary is built by counting word frequencies in the training set and selecting the top 73,738 most common words ($|V| = 73,738$).
- **Sequence Formatting:** Each review is tokenized, converted to integer indices, and standardized to a fixed length of $L = 200$ tokens. Sequences shorter than this limit are padded with zeros (index 0), while longer sequences are truncated.
- **Label Encoding:** Target labels are encoded into integer format (e.g., negative $\rightarrow$ 0, mixed $\rightarrow$ 1, positive $\rightarrow$ 2) using a Label Encoder fitted on the training data.

Network Architecture The model architecture consists of four primary components arranged sequentially:

1. **Embedding Layer:** The input layer is a trainable embedding matrix $E \in \mathbb{R}^{|V| \times d_{emb}}$, where the embedding dimension d_{emb} is set to 128. This layer learns dense vector representations for each token during the training process.
2. **Stacked Bi-LSTM Layers:** We employ a stack of 2 Bidirectional LSTM layers. Each layer has a hidden dimension of $h_{dim} = 128$ per direction. Consequently, the output at each timestep is the concatenation of the forward and backward hidden states, resulting in a feature vector of size 256 (128×2).
3. **Fully Connected Classification Head:** The extracted features (the final hidden states) are passed through a series of dense layers with Rectified Linear Unit (ReLU) activation to perform classification. The dimensionality reduction proceeds as follows:

$$256 \xrightarrow{FC1} 64 \xrightarrow{FC2} 64 \xrightarrow{FC3} 16 \xrightarrow{Output} N_{classes}$$

4. **Regularization:** To prevent overfitting, Dropout regularization with a rate of $p = 0.5$ is applied after the LSTM layers and between each fully connected layer.

The total number of trainable parameters for this configuration is approximately 10.2 million.

Training Configuration The model training follows a two-stage approach: hyperparameter optimization via grid search, followed by full-scale training with the optimal configuration. Both stages share common training infrastructure detailed in Table 1.

Stage 1: Grid Search for Learning Rate Optimization. To identify the optimal learning rate while maintaining computational efficiency, we conduct a grid search on a 10% subset of the training data:

- **Search Space:** Six learning rate candidates: $\eta \in \{10^{-5}, 5 \times 10^{-5}, 10^{-4}, 5 \times 10^{-4}, 10^{-3}, 5 \times 10^{-3}\}$.
- **Validation Frequency:** Every 250 training steps (adapted for smaller dataset).
- **Selection Metric:** Macro-averaged F1 score on the validation set, ensuring balanced performance across all sentiment classes (negative, mixed, positive).

Table 1: Common Hyperparameters for LSTM Training

Parameter	Value
Embedding Dimension	128
Hidden Dimension (per direction)	128
LSTM Layers	2 (Bidirectional)
Dropout Rate	0.5
Max Sequence Length	200 tokens
Vocabulary Size	73,738
FC Architecture	$256 \to 64 \to 64 \to 16 \to N_{classes}$
Batch Size	64
Epochs	20
Optimizer	Adam
Loss Function	NLLLoss (3-class)

Stage 2: Final Training with Optimal Configuration. Using the best learning rate η^* identified in Stage 1, we train on the complete dataset (100% of training data):

- **Validation Frequency:** Every 500 training steps (standard interval for full-scale training).
- **Checkpoint Management:** When validation F1 improves, a checkpoint is saved containing model state, optimizer state, and performance metrics. Only the top 2 checkpoints are retained to conserve storage.
- **Model Selection:** After training completes, the checkpoint with the highest validation F1 score is automatically loaded for final test set evaluation, serving as an implicit early stopping mechanism.

3.3.2 Proposed Approach 1: Fine-Tuned RoBERTa

To capture the deep semantic context required for understanding game reviews, we employed the **RoBERTa** (Robustly Optimized BERT Pretraining Approach) architecture. Specifically, we utilized the `FacebookAI/roberta-base` variant, which consists of a 12-layer bidirectional Transformer encoder with 125M parameters. Unlike the standard BERT model, RoBERTa is pre-trained with dynamic masking and without the Next Sentence Prediction (NSP) objective, making it particularly effective for downstream sequence classification tasks where context density varies.

Network Architecture We leveraged the pre-trained `roberta-base` model from HuggingFace's Transformers library [3]. The architecture consists of:

- **Embedding Layer:** Byte-Pair Encoding (BPE) tokenizer with a vocabulary size of 50,265 tokens, projecting each token to a 768-dimensional space.
- **Encoder:** 12 stacked Transformer layers, each containing 12 self-attention heads. The total hidden dimension is 768, with an intermediate feed-forward dimension of 3072.
- **Classification Head:** A linear layer projects the pooled representation of the `<s>` token (start-of-sequence) to our three sentiment classes: negative, mixed, and positive ($N_{classes} = 3$).

The model was initialized with pre-trained weights learned on large-scale corpora, allowing it to leverage broad linguistic knowledge before fine-tuning on our domain-specific game review dataset.

Data Preprocessing and Tokenization Raw text reviews undergo preprocessing through the RoBERTa tokenizer, which implements Byte-Pair Encoding (BPE):

- **Tokenization:** Each review is tokenized into subword units using the pre-trained BPE vocabulary. Special tokens `<s>` (start) and `</s>` (end) are automatically inserted to delimit the sequence.
- **Sequence Length Management:** Reviews are processed with a maximum sequence length of $L = 256$ tokens. Sequences exceeding this limit are truncated, while shorter sequences are padded with `<pad>` tokens to ensure uniform tensor dimensions across batches.
- **Attention Masking:** Attention masks are generated to prevent the model from attending to padding tokens, ensuring that only meaningful content influences the predictions.
- **Label Encoding:** Sentiment labels are mapped to integer indices (e.g., negative $\rightarrow$ 0, mixed $\rightarrow$ 1, positive $\rightarrow$ 2) for compatibility with the classification head.

Training Configuration The model fine-tuning follows a two-stage approach: hyperparameter optimization via grid search, followed by full-scale training with the optimal configuration. Both stages share common training infrastructure detailed in Table 2.

Table 2: Common Hyperparameters for RoBERTa Fine-Tuning

Parameter	Value
Base Model	`FacebookAI/roberta-base`
Model Parameters	125M
Encoder Layers	12 (Bidirectional Transformer)
Hidden Dimension	768
Attention Heads	12
Max Sequence Length	256 tokens
Batch Size	64
Epochs	5
Weight Decay	0.01
Warmup Steps	0
LR Scheduler	Linear
Optimizer	AdamW
Loss Function	Cross-Entropy
Mixed Precision (FP16)	Enabled

Stage 1: Grid Search for Learning Rate Optimization. To identify the optimal learning rate for the fine-tuning process, we conduct a grid search on a 10% subset of the training data:

- **Search Space:** Four learning rate candidates: $\eta \in \{5 \times 10^{-6}, 1 \times 10^{-5}, 2 \times 10^{-5}, 5 \times 10^{-5}\}$.
- **Validation Frequency:** Every 500 training steps (adapted for smaller dataset).
- **Selection Metric:** Macro-averaged F1 score on the validation set, ensuring balanced performance across all sentiment classes (negative, mixed, positive).

Stage 2: Final Training with Optimal Configuration. Using the best learning rate η^* identified in Stage 1, we fine-tune on the complete dataset (100% of training data):

- **Validation Frequency:** Every 1000 training steps (standard interval for full-scale training).
- **Checkpoint Management:** When validation F1 improves, a checkpoint is saved containing model

state, optimizer state, and performance metrics. Only the top 2 checkpoints are retained to conserve storage.

- **Model Selection:** After training completes, the checkpoint with the highest validation F1 score is automatically loaded for final test set evaluation, serving as an implicit early stopping mechanism.

3.3.3 Proposed Approach 2: EmbeddingGemma-300m with XGBoost

To leverage pre-trained contextual embeddings while maintaining computational efficiency, we employed a two-component pipeline: **EmbeddingGemma-300m** for feature extraction and **XGBoost** for classification. This approach decouples representation learning from classification, enabling efficient hyperparameter tuning on the classifier without re-computing expensive embeddings.

Embedding Generation with EmbeddingGemma-300m We utilized Google's `embeddinggemma-300m` model [4], a 300M parameter encoder specifically designed for generating high-quality text embeddings. This model is built on the Gemma architecture and fine-tuned for semantic similarity tasks, making it particularly suitable for capturing the nuanced sentiment expressions in game reviews.

Architecture and Processing:

- **Model Specifications:** The `embeddinggemma-300m` model employs a Transformer-based architecture with 300M parameters. Unlike traditional masked language models, it is optimized specifically for embedding generation, producing dense 768-dimensional vector representations.
- **Tokenization:** Reviews are tokenized using the model's native tokenizer (SentencePiece-based) with a maximum sequence length of $L = 256$ tokens. This length balances capturing sufficient context while maintaining computational efficiency.
- **Checkpoint System:** To handle potential session interruptions during large-scale embedding generation, we implemented a checkpoint-based system that saves embeddings incrementally every 200 batches. This allows resumption from the last checkpoint without re-computing completed work.
- **Precision:** Unlike many modern models, EmbeddingGemma-300m does not support mixed-precision (FP16) inference. All computations are performed in FP32, which is optimal for the model's architecture and ensures numerical stability.

The embedding generation process produces fixed-size 768-dimensional vectors for each review, regardless of the original text length. These embeddings serve as input features for the downstream XGBoost classifier.

XGBoost Classification We employed **XGBoost** (eXtreme Gradient Boosting) [1], a highly efficient gradient boosting framework known for its strong performance on tabular data and robustness to class imbalance. XGBoost constructs an ensemble of decision trees through iterative boosting, where each tree corrects the errors of previous trees.

Class Imbalance Handling: Given the inherent class imbalance in sentiment data, we implemented several strategies:

- **Sample Weighting:** Applied balanced class weights, inversely proportional to class frequencies, to ensure minority classes receive appropriate attention during training.
- **Early Stopping:** Monitored validation loss with a patience of 200 rounds to prevent overfitting while allowing sufficient training for minority classes.
- **Regularization:** Employed both L1 (α) and L2 (λ) regularization to control model complexity and improve generalization.

Training Configuration The training follows a two-stage approach: grid search for hyperparameter optimization, followed by full-scale training. Both stages share common infrastructure detailed in Table 3.

Table 3: Common Hyperparameters for XGBoost Training

Parameter	Value
Embedding Model	`google/embeddinggemma-300m`
Embedding Dimension	768
Max Sequence Length	256 tokens
Batch Size (embedding)	128
Loss Function	Cross-Entropy
Eval Metric	Log Loss
Tree Method	Histogram-based
Early Stopping Patience	200 rounds
Class Weighting	Balanced

Stage 1: Grid Search for Hyperparameter Optimization. To identify the optimal XGBoost hyperparameters, we conduct a grid search on a 10% subset of the training data (using pre-computed embeddings):

- **Search Space:**
  - Learning rate: $\eta \in \{0.1, 0.2\}$
  - Max depth: $\{4, 6\}$
  - Min child weight: $\{1, 3\}$
  - Subsample ratio: $\{0.8, 1.0\}$
  - Column subsample ratio: $\{0.8, 1.0\}$
  - L2 regularization (λ): $\{1, 10\}$
- **Fixed Parameters:** Number of estimators fixed at 5000 with early stopping to find optimal iteration count.
- **Selection Metric:** Macro-averaged F1 score on the validation set, ensuring balanced performance across all sentiment classes.

Stage 2: Final Training with Optimal Configuration. Using the best hyperparameters identified in Stage 1, we train on embeddings from the complete dataset (100% of training data):

- **Training Protocol:** XGBoost's built-in early stopping monitors validation performance every boosting round, terminating training when validation loss fails to improve for 200 consecutive rounds.
- **Model Selection:** The model at the iteration with lowest validation loss is automatically retained for final evaluation.

3.3.4 Evaluation Metrics

To evaluate the performance of our game review sentiment classifier, we utilize standard metrics adapted for a 3-class problem ($C = 3$: Positive, Negative, Mixed). Let TP_i, FP_i, and FN_i denote the True Positives, False Positives, and False Negatives for class i, respectively.

Per-Class Metrics These metrics assess how well the model handles specific sentiments (e.g., distinguishing *Mixed* reviews from *Negative* ones).

- **Precision (P_i):** The accuracy of specific sentiment predictions.

$$P_i = \frac{TP_i}{TP_i + FP_i} \tag{7}$$

  Meaning: For the *Negative* class, high precision ensures that reviews flagged as negative are indeed complaints, minimizing false alarms for developers.

- **Recall (R_i):** The coverage of actual sentiment instances.

$$R_i = \frac{TP_i}{TP_i + FN_i} \tag{8}$$

  Meaning: High recall on the *Mixed* class indicates the model successfully identifies nuanced feedback without oversimplifying it into purely positive or negative.

- **F1-Score ($F1_i$):** The harmonic mean of P_i and R_i.

$$F1_i = 2 \cdot \frac{P_i \cdot R_i}{P_i + R_i} \tag{9}$$

  Meaning: Provides a balanced view for a specific sentiment, ensuring neither valid reviews are missed nor incorrect labels are applied.

- **Confusion Matrix:** A 3×3 matrix showing misclassifications. *Meaning:* It visually reveals specific confusion patterns, such as whether the model frequently misinterprets sarcasm in *Mixed* reviews as *Positive*.

Overall Metrics These provide a global summary of the classifier's effectiveness.

- **Accuracy (Acc):** The overall ratio of correct predictions.

$$\text{Acc} = \frac{\sum_{i=1}^{3} TP_i}{N} \tag{10}$$

  Meaning: A general indicator of success, though potentially misleading if game reviews are heavily skewed toward one sentiment (e.g., *Positive*).

- **Macro Averages (M-Prec, M-Rec, M-F1):** Unweighted averages across the three classes.

$$\text{M-F1} = \frac{1}{3} \sum_{i \in \{Pos, Neg, Mix\}} F1_i \tag{11}$$

Meaning: Critical for ensuring the minority class (e.g, *Mixed*) is not ignored. A high Macro F1 implies the model performs well on distinct, nuanced reviews, not just the majority.

- **Weighted Averages (W-Prec, W-Rec, W-F1):** Averages weighted by support S_i.

$$\text{W-F1} = \frac{1}{N} \sum_{i=1}^{3} S_i \cdot F1_i \tag{12}$$

Meaning: Reflects the "real-world" user experience, prioritizing performance on the most frequent review types.

- **Training Time:** The total wall-clock time required to complete the model training phase. *Meaning:* Measures the computational cost and resource intensity. A reasonable training time is essential for the feasibility of frequent model updates and retraining cycles.
- **Throughput:** The processing speed defined as N/T_{total} calculated from *test set*. *Meaning:* Determines if the system can handle high-volume streams of reviews during peak events (e.g., game launch days) in real-time.

4 Dataset Characteristics

The final dataset comprises 629,884 game reviews collected from Metacritic through automated web scraping, representing a comprehensive collection of user-generated sentiment data across the gaming domain. This dataset captures diverse player perspectives spanning multiple gaming platforms, genres, and release periods, providing a robust foundation for sentiment analysis modeling.

4.1 Class Distribution and Imbalance Analysis

The dataset exhibits a multi-class sentiment distribution with three distinct categories: positive, negative, and mixed reviews. The distribution demonstrates notable class imbalance characteristic of real-world user review data:

- **Positive reviews:** 368,669 samples (58.5%)
- **Negative reviews:** 159,847 samples (25.4%)
- **Mixed reviews:** 101,368 samples (16.1%)

The positive class represents the majority with approximately 58.5% of the total reviews, establishing a 2.31:1 ratio compared to negative reviews and a 3.64:1 ratio compared to mixed reviews. This imbalance reflects realistic user behavior patterns where satisfied players are more likely to leave reviews. At the same time, negative reviews account for over a quarter of the dataset, ensuring substantial representation of critical feedback. Mixed reviews, which contain nuanced or conflicting sentiments, constitute the

smallest but analytically most complex class at approximately 16% of the total samples.

The class distribution presents several implications for model training and evaluation:

1. The imbalance necessitates careful consideration of evaluation metrics beyond simple accuracy, favoring metrics such as precision, recall, and F1-score.
2. Stratified sampling is employed during train–validation–test splits to maintain proportional class representation across all data partitions.
3. The presence of three sentiment classes increases classification complexity compared to binary sentiment analysis, requiring models to distinguish between strong sentiments and nuanced mixed opinions.

4.2 Textual Characteristics and Statistical Properties

A comprehensive analysis of review text properties reveals distinct characteristics across sentiment categories. Aggregate statistics are summarized in Table 4.

Table 4: Comprehensive Text Statistics by Review Category

Metric	Positive	Negative	Mixed	Overall
Character Length Statistics				
Mean	551	609	826	610
Median	304	343	484	337
Std. Deviation	706	754	938	766
Minimum	3	3	4	3
Maximum	8,995	8,978	8,860	8,995
Word Count Statistics				
Mean	100	111	150	111
Median	56	63	89	62
25th Percentile	30	33	44	33
75th Percentile	114	129	185	128
Std. Deviation	128	137	169	139

Length Distribution Patterns: Reviews exhibit substantial variation in length, averaging 610 characters (111 words) overall. The minimum review length is 3 characters, while the maximum reaches 8,995 characters. The interquartile range spans from 178 to 698 characters and from 33 to 128 words, reflecting wide diversity in user expression.

Variability Analysis: The high standard deviation of 766 characters indicates pronounced heterogeneity in review verbosity. The dataset includes extremely brief comments as well as extensive, essay-style critiques. This variability reflects diverse user writing styles and necessitates models capable of handling both concise expressions and long-form discourse.

Distribution Characteristics: The median review length (337 characters) is substantially lower than the mean (610 characters), indicating a right-skewed distribution. Approximately 50% of reviews contain fewer than 62 words, while 25% exceed 128 words, demonstrating the predominance of short-form feedback typical of online review platforms.

4.3 Representative Sample Analysis

To illustrate the distinct linguistic and stylistic patterns across sentiment categories, representative examples from each class are presented below with analytical commentary.

Sample Selection Methodology

Representative samples were selected through a systematic stratified random sampling approach:

1. **Random sampling:** One hundred reviews were randomly selected from each category.
2. **Statistical filtering:** Outliers were excluded (reviews with fewer than 10 words or more than 300 words).
3. **Category-specific filtering (mixed reviews only):** Priority was given to samples containing contrast markers (e.g., "but", "however").
4. **Stratified selection by length:** Five samples per category were selected across the length spectrum (short, medium, long).

This procedure ensures that selected samples are representative of both the statistical properties and linguistic diversity of each sentiment category.

Positive Review Samples

1. *Short, enthusiastic (23 words, 121 characters):* `Enjoyed this game, pretty much it was the best game I ever played this year, however I hate it being an exclusive experience.`

   Analysis: Expresses strong positive sentiment using superlative language ("best game I ever played"), while briefly acknowledging a negative aspect (platform exclusivity).

2. *Medium, detailed praise (64 words, 363 characters):*

   `Mario Kart World marks the new beginning of the Mario Kart series, and thanks to the power of Switch 2, it does so in style. A quality open world, graphically very valid and with fun gameplay that never tires. The immense roster, also thanks to the alternative skins, and the tracks reworked to guarantee replayability, make it a must have for every Nintendo fan.`

   Analysis: Highlights multiple positive dimensions, including graphics, gameplay, and replayability, and targets a specific audience segment.

3. *Long, nuanced appreciation (129 words, 694 characters):*

   `The base game is good and for my first campaign it was played without any bugs. The plot is ok and forgettable with the cliché of our terrible politics. I miss the multiplayer game choice. My impression is that every faction is balanced in its own way. Why not a Hotseat mode or even online play? The 2D art is good, but the same cannot be said about the 3D graphics. I still remember the great animations in Brig-`

andine: Legend of Forsena Grand Edition on the PlayStation 1. I hope more games like this come to new generations of consoles or PCs.

Analysis: Combines praise with constructive criticism and nostalgic references, typical of long-form positive reviews.

4. *Very long, in-depth critique (252 words, 1,393 characters):*

   Dead Space 2, on the surface, appears to be more of the same. After the hard slog through Dead Space one, it seems the sequel would repeat the formula. But instead, Dead Space 2 features a number of sequences that emphasize intensity over repetition. Overall, the game is more intense and better paced than the original, with stronger storytelling and character development. While some design choices remain questionable, the game ultimately delivers a fulfilling experience.

   Analysis: Provides a comprehensive evaluation with comparisons to earlier titles, addressing pacing, mechanics, and narrative depth.

5. *Long, balanced with caveats (175 words, 1,016 characters):*

   As an original player of FFXIV 1.0, I went into ARR with low expectations. I was surprised by the sheer overhaul of the game. The environments, sound, and gameplay all improved significantly. That said, launch issues such as server congestion negatively impacted the experience. Once these problems were resolved, the game proved to be excellent overall.

   Analysis: Demonstrates how positive reviews can include significant criticism while maintaining an overall favorable assessment.

Negative Review Samples

1. *Short, direct criticism (18 words, 99 characters):* The story is boring and online is disgusting. I think it has the most toxic community in video games.

   Analysis: Uses strong negative language and direct condemnation, typical of brief negative reviews.

2. *Medium, comparative disappointment (97 words, 545 characters):*

   I have been playing this since launch and can honestly say it is average. MKX was far more enjoyable. The story mode is exciting, but online play is poorly balanced and repetitive.

   Analysis: Contrasts the title with previous entries, acknowledging minor strengths but emphasizing overall dissatisfaction.

3. *Long, detailed critique (147 words, 770 characters):*

   The game is below average in nearly every aspect. While controls are functional and visuals are acceptable in places, the plot is dull and overstays its welcome. For such a weak narrative, the game length feels

`excessive.`

Analysis: Enumerates multiple flaws with minimal praise, reinforcing the negative sentiment.

4. *Very long, franchise critique (204 words, 1,085 characters):*

   `This game abandons the survival horror roots of the franchise in favor of repetitive action mechanics. While some familiar characters return, the experience ultimately feels exhausting and unfocused.`

   Analysis: Critiques gameplay direction and narrative consistency, combining nostalgia with disappointment.

5. *Extensive, technical and emotional critique (259 words, 1,391 characters):*

   `FIFA 15 starts strong visually but deteriorates quickly due to technical glitches and gameplay imbalance. Online issues, repeated bugs, and perceived bias significantly diminish enjoyment, making this one of the weakest entries in the series.`

   Analysis: Combines technical complaints with emotional frustration, characteristic of long-form negative reviews.

Mixed Review Samples

1. *Short, historical context (88 words, 451 characters):*

   `It was one of the pioneers of RTS games and highly innovative in 1997. However, having played more modern titles first, I found it boring. Despite its historical importance, it did not fully resonate with me.`

   Analysis: Balances historical appreciation with personal disappointment, using contrastive language.

2. *Medium, comparative evaluation (119 words, 677 characters):*

   `Liberation HD is solid overall. The mechanics work well, and the protagonist is compelling. However, the game lacks some features expected from the franchise and feels shorter than ideal.`

   Analysis: Presents both strengths and limitations, concluding with a balanced assessment.

3. *Long, detailed contrast (152 words, 813 characters):*

   `The game excels in visuals and movement, but the narrative is bland and repetitive. While enjoyable at first, the experience gradually becomes tedious.`

   Analysis: Strongly contrasts positive mechanics with narrative weaknesses.

4. *Very long, repetitive gameplay critique (181 words, 981 characters):*

   `The gameplay loop is initially fun but quickly becomes repetitive. While the ending is satisfying, much of the experience feels unnecessarily prolonged.`

   Analysis: Uses multiple contrast markers to articulate both enjoyment and fatigue.

5. *Extensive, balanced assessment (225 words, 1,199 characters):*

   ```
   The core gameplay mechanics are solid, but the game lacks innovation.
   Despite several strong elements, the overall experience fails to stand
   out, justifying a moderate score.
   ```

   Analysis: Provides a reasoned justification for a middle-ground rating, characteristic of mixed sentiment reviews.

4.4 Linguistic Patterns and Vocabulary Analysis

A detailed vocabulary analysis across all 629,884 reviews reveals category-specific lexical preferences and semantic patterns characteristic of each sentiment class.

Positive Review Lexicon: Positive reviews are dominated by affirmative and superlative language, signaling strong endorsement:

- *Quality indicators:* "great" (150,546), "best" (122,688), "amazing" (68,706), "good" (144,757)
- *Enjoyment markers:* "fun" (120,103), "love" (frequent)
- *Narrative elements:* "story" (164,517), "characters" (65,135), "world" (69,870)
- *Technical aspects:* "gameplay" (94,128), "graphics" (75,759)

Negative Review Lexicon: Negative reviews emphasize dissatisfaction, value concerns, and causal explanation:

- *Direct criticism:* "bad" (40,331), "boring", "disappointing"
- *Economic concerns:* "money" (28,064)
- *Temporal qualifiers:* "after" (29,286), "then" (28,351)
- *Causality markers:* "because" (40,281)

Mixed Review Lexicon: Mixed reviews display balanced sentiment vocabulary and comparative reasoning:

- *Balanced sentiment terms:* "good" (62,804), "fun" (44,575), "great" (32,117), "bad" (26,632)
- *Comparative language:* "better" (24,681), "still" (27,706)
- *Qualification markers:* "feel", "way" (23,596)

Contrasting Discourse Patterns: Analysis of mixed reviews reveals sophisticated discourse structures:

1. Co-occurrence of positive and negative sentiment markers within the same review.
2. Frequent use of contrastive conjunctions such as "but", "however", "although", and "despite".
3. Qualified sentiment intensity, often expressing overall approval tempered by acknowledged shortcomings.

These findings demonstrate that mixed reviews represent genuinely nuanced opinions rather than neutral sentiment, characterized by explicit evaluation of both strengths and weaknesses.

4.5 Domain-Specific Characteristics

The gaming review dataset exhibits several domain-specific linguistic patterns:

Common evaluation dimensions: Users consistently assess gameplay mechanics, narrative quality, visual presentation, and entertainment value, reflecting the multidimensional nature of gaming experiences.

Comparative tendencies: Frequent references to other games indicate that evaluations are often contextual and comparative, complicating sentiment interpretation.

Temporal dynamics: Terms such as "first", "still", and "after" suggest longitudinal evaluation, capturing changes in perception over time due to gameplay depth, updates, or post-release support.

5 Training Process Report

This section details the training pipeline and the intermediate results obtained at each stage. It is important to note that the metrics presented here are derived from the validation set and are distinct from the final test evaluation. Consequently, the analysis focuses on a selected subset of key performance indicators rather than the comprehensive suite described previously.

5.1 Baseline Model: Bidirectional LSTM

5.1.1 Phase I: Hyperparameter Tuning (Grid Search)

A grid search was conducted to determine the optimal learning rate using a 10% subset of the dataset. Six configurations were evaluated, ranging from 1×10^{-5} to 5×10^{-3}.

Selection Process **Configuration 4 (5×10^{-4})** was selected for the final training run. This configuration offered the best balance for the model, achieving the highest Validation F1-Macro score of 0.6872 among the stable configurations.

Table 5: Grid Search Results for Hyperparameter Tuning

Config	Learning Rate	Val F1-Macro	Val Accuracy	Training Time (s)
1	1×10^{-5}	0.2473	0.5899	1335.76
2	5×10^{-5}	0.5369	0.7622	1355.36
3	1×10^{-4}	0.6611	**0.7707**	1374.95
4	5×10^{-4}	**0.6872**	0.7681	1399.33
5	1×10^{-3}	0.6867	0.7589	1409.00
6	5×10^{-3}	0.6665	0.7366	1414.07

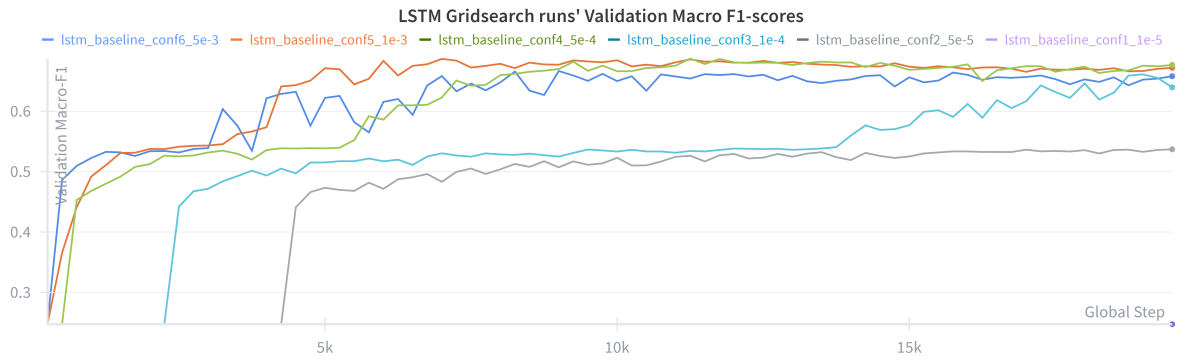

Figure 1: Validation F1-Macro Scores across Grid Search Configurations

5.1.2 Phase II: Training Execution & Validation Performance

The final model was trained on the full dataset using the learning rate derived from the grid search. The training process took a total of 33,635.10 seconds (approx. 9.34 hours) to complete.

Training Configuration

- **Learning Rate:** 0.0005
- **Batch Size:** 64
- **Epochs:** 20
- **Sequence Length (Max):** 200
- **Device:** NVIDIA Tesla P100 16GB

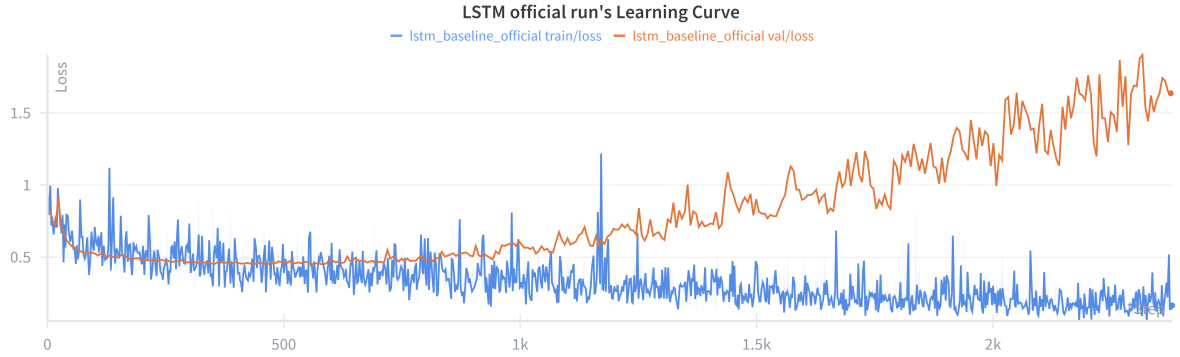

Figure 2: Training and Validation Learning Curves (Loss/Accuracy) during Final Training. *Note: The x-axis is labeled as "Step" rather than "Global Step" due to a minor implementation issue; however, the correctness of visualization remains the same.*

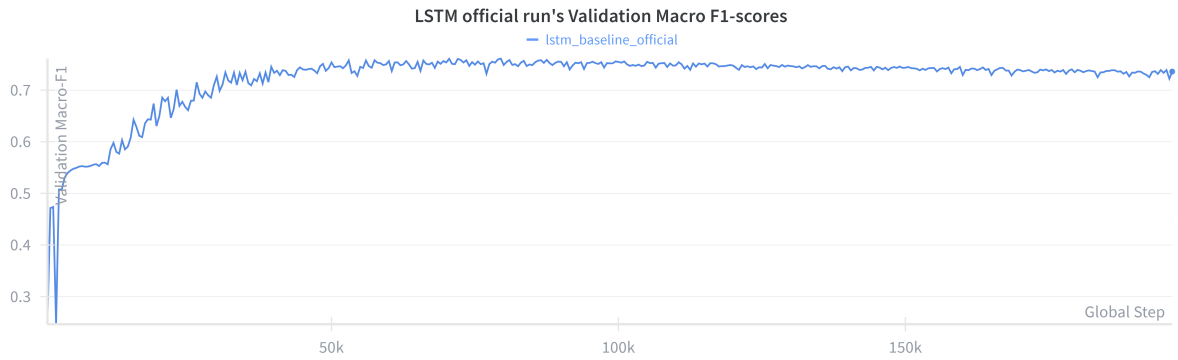

Figure 3: Progression of Validation F1-Score over Epochs

Validation Metrics Post-training evaluation on the validation set resulted in an overall **Accuracy of 82.78%** and a **Macro F1-Score of 0.7612**.

5.2 Proposed Approach 1: Fine-Tuned RoBERTa

5.2.1 Phase I: Hyperparameter Tuning (Grid Search)

A grid search was performed on December 23, 2025, using a 10% subset of the dataset to identify the optimal learning rate. Four configurations were tested, focusing on smaller learning rates appropriate for fine-tuning a pre-trained transformer.

Selection Process Configuration 2 (1×10^{-5}) was selected as the optimal hyperparameter setting. It achieved the highest performance across both key metrics, yielding a **Validation F1-Macro of 0.8706** and a **Validation Accuracy of 0.8729**.

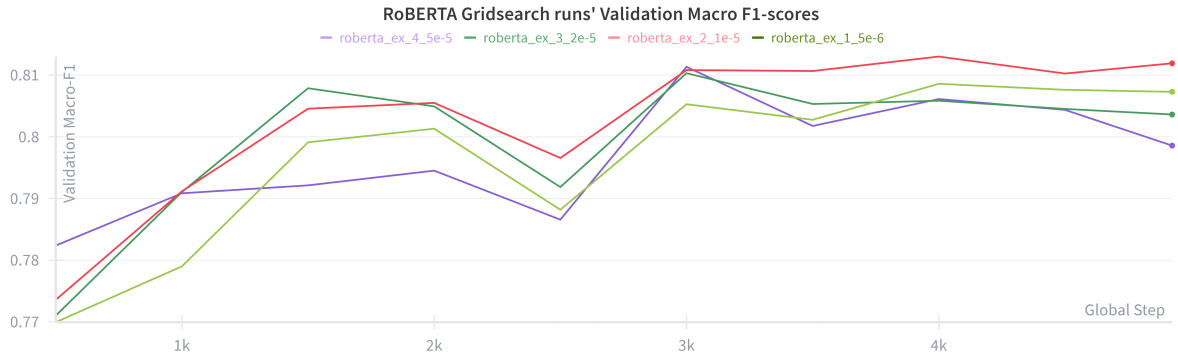

Figure 4: Comparison of Validation F1-Macro Scores across RoBERTa Grid Search Configurations

Table 6: Grid Search Results for RoBERTa Hyperparameter Tuning

Config	Learning Rate	Val F1-Macro	Val Accuracy	Training Time (s)
1	5×10^{-6}	0.8676	0.8706	7795.46
2	1×10^{-5}	**0.8706**	**0.8729**	7795.71
3	2×10^{-5}	0.8649	0.8665	7794.86
4	5×10^{-5}	0.8610	0.8624	7803.05

5.2.2 Phase II: Training Execution & Validation Performance

The final model was fine-tuned on the full dataset using the parameters selected from the grid search. The training process required a total of 13,961.28 seconds (approx. 3.88 hours).

Training Configuration

- **Model Architecture:** FacebookAI/roberta-base
- **Learning Rate:** 1×10^{-5}
- **Batch Size:** 64
- **Epochs:** 5
- **Max Sequence Length:** 256
- **Weight Decay:** 0.01
- **Device:** NVIDIA Tesla P100 16GB

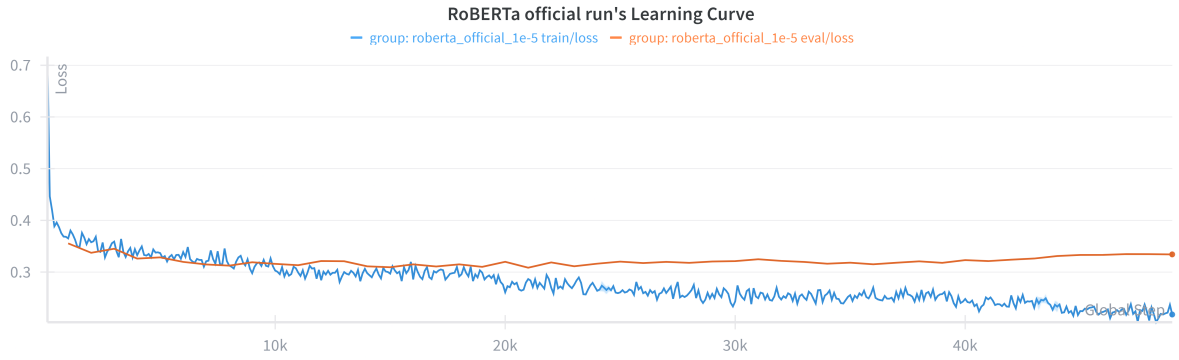

Figure 5: Training and Validation Learning Curves (Loss/Accuracy) for RoBERTa

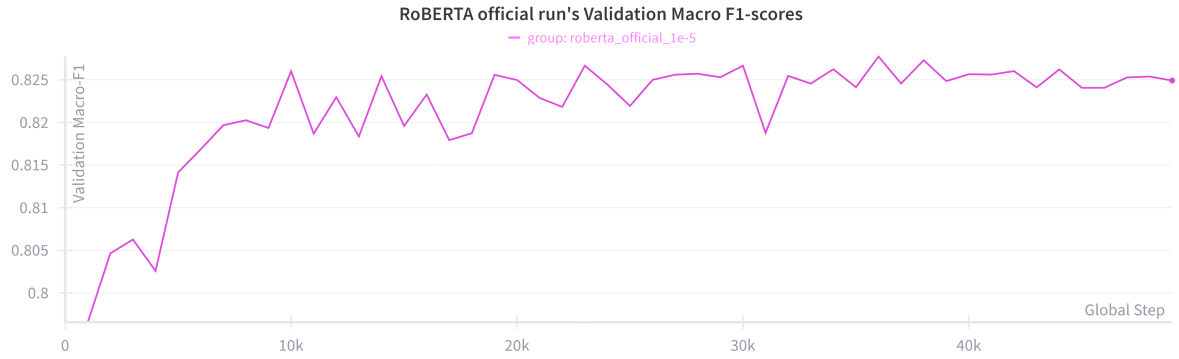

Figure 6: Progression of Validation F1-Score over Epochs (RoBERTa)

Validation Metrics Post-training evaluation on the validation set demonstrated a significant improvement over the baseline. The model achieved an overall **Accuracy of 88.23%** and a **Macro F1-Score of 0.8277**.

5.3 Proposed Approach 2: EmbeddingGemma-300m with XGBoost

5.3.1 Phase I: Feature Extraction (Embedding Generation)

Following the architecture defined previously, the first stage involved converting the raw text reviews into high-dimensional vector representations using the `embeddinggemma-300m` model.

We processed the full training dataset of 629,884 samples. To manage the computational load, the checkpointing system described earlier was utilized to save embeddings incrementally every 200 batches, ensuring the process was robust to interruptions.

While this phase represented the heaviest initial computational cost, it was a one-time operation. Once computed, the fixed embeddings allowed the downstream XGBoost model to be trained and iterated upon rapidly.

5.3.2 Phase II: Hyperparameter Tuning (Grid Search)

With the features pre-computed, a grid search was conducted to optimize the XGBoost classifier using a 10% subset of the data. The search evaluated 64 configurations, exploring learning rates ($\eta \in \{0.1, 0.2\}$), tree depths ($\{4, 6\}$), minimum child weights ($\{1, 3\}$), subsample ratios ($\{0.8, 1.0\}$), column sample ratios ($\{0.8, 1.0\}$), and regularization terms ($\lambda \in \{1, 10\}$).

Selection Process The analysis revealed that a shallower tree depth (`max_depth`=4) combined with a conservative learning rate ($\eta = 0.1$) and moderate regularization ($\lambda = 1$) yielded better generalization than more complex models. The optimal configuration achieved a Validation F1-Macro of 0.7042 on the 10% subset.

Table 7: Top XGBoost Hyperparameter Configurations

N Est.	Max D.	LR	Min CW	Subsamp.	Colsamp.	Reg λ	Acc	F1	Traing Time (s)
5000	4	0.1	3	0.8	0.8	1	0.7831	**0.7042**	34.00
5000	4	0.1	1	0.8	0.8	10	0.7813	0.7040	34.08
5000	4	0.1	3	0.8	1.0	1	0.7817	0.7028	32.52
5000	4	0.1	3	1.0	0.8	10	0.7809	0.7028	36.16
5000	4	0.1	1	0.8	1.0	10	0.7801	0.7025	30.27
5000	4	0.1	3	0.8	1.0	10	0.7801	0.7014	32.40
5000	4	0.1	3	1.0	1.0	10	0.7795	0.7012	33.37
5000	4	0.1	3	1.0	0.8	1	0.7786	0.7005	32.24
5000	4	0.1	1	1.0	0.8	1	0.7761	0.7005	28.40
5000	4	0.1	1	1.0	0.8	10	0.7778	0.6994	34.33
					...				

5.3.3 Phase III: Classifier Training & Validation Performance

The final model was trained on the full dataset using the optimal hyperparameters. A significant advantage of this decoupled approach was the training speed; the XGBoost training phase completed in just 513.97 seconds (approx. 8.5 minutes), markedly faster than the deep learning baselines.

Training Configuration

- **Number of Estimators:** 5000 (with early stopping)
- **Maximum Depth:** 4
- **Learning Rate:** 0.1
- **Subsample Ratio / Column Sample by Tree:** 0.8
- **Minimum Child Weight:** 3
- **Device:** NVIDIA Tesla P100 16GB

Validation Metrics The XGBoost model demonstrated robust performance, achieving an overall **Accuracy of 84.74%** and a **Macro F1-Score of 0.7695** on the full validation set.

6 Results

All three models were evaluated on the same held-out test set ($n = 78{,}737$) with the same class distribution: Positive 58.24% ($45{,}859$), Negative 25.63% ($20{,}181$), Mixed 16.13% ($12{,}697$).

6.1 Overall metrics (global)

Metric	LSTM	EmbeddingGemma + XGBoost	RoBERTa
Acc	0.8266	0.8495	**0.8827**
W-Prec	0.8293	0.8415	**0.8805**
W-Rec	0.8266	0.8495	**0.8827**
W-F1	0.8279	0.8435	**0.8815**
M-Prec	0.7560	0.7824	**0.8313**
M-Rec	0.7617	0.7683	**0.8247**
M-F1	0.7587	0.7717	**0.8278**
Throughput	**∼1,913 samples/s**	∼176 samples/s	∼74.35 samples/s
Training time	33,635 (9h21m)	**513 (9m)**	112,245 (31h10m)

Table 8: Overall test-set performance and inference efficiency. Acc = Accuracy (micro-F1). W-* = weighted average. M-* = macro average.

Overall, the results show a clear trade-off between predictive quality (Acc, weighted and macro Precision/Recall/F1) and efficiency (throughput and training time). RoBERTa consistently leads on the performance metrics—both weighted averages and the more class-balanced macro averages—suggesting stronger generalization and better robustness across classes, but it is also the least efficient in both inference speed and training time. Its inference throughput is only 74 samples/s and fine-tuning takes about 31 hours on our hardware, compared with 9 hours for the LSTM and 9 minutes for EmbeddingGemma + XGBoost. EmbeddingGemma + XGBoost provides a solid middle ground, with competitive scores on the same metrics while remaining much faster to train and reasonably efficient at inference. LSTM lags behind the other two models on predictive quality, but it stands out in inference throughput, processing nearly 1,900 samples/s. The training-time comparison should still be interpreted with caution, since the models were trained with different epoch budgets (20 epochs for LSTM vs. 5 for RoBERTa), so the reported times reflect both the computational cost per update and the chosen optimization budget.

6.2 Per-class analysis (Metrics + Confusion matrices)

Per-class performance (Precision / Recall / F1). Table 9 reports per-class Precision, Recall, and F1 on the test set for all models. Overall, all models perform strongly on **Positive** and **Negative** sentiment, while **Mixed** remains the most challenging class with noticeably lower Precision/Recall/F1. RoBERTa is the most consistent across classes, achieving the best overall balance—especially improving Mixed-class F1—while EmbeddingGemma + XGBoost shows a stronger trade-off between precision and recall depending on the class (notably weaker recall for Mixed). LSTM is competitive on the major classes but struggles most on Mixed, suggesting it benefits less from nuanced or ambiguous examples compared to RoBERTa.

Model	Metric	Positive	Negative	Mixed
LSTM	P	0.9148	0.8232	0.5301
	R	0.8990	0.8366	0.5493
	F1	0.9068	0.8298	0.5395
EmbeddingGemma + XGBoost	P	0.9168	0.8089	0.6215
	R	0.9333	0.8916	0.4802
	F1	0.9249	0.8482	0.5418
RoBERTa (fine-tuned)	P	0.9379	0.8770	0.6791
	R	0.9494	0.8830	0.6417
	F1	0.9436	0.8800	0.6599

Table 9: Per-class performance on the test set (P = Precision, R = Recall, F1 = F1-score).

Confusion matrices (per-class error patterns). For all confusion matrices, rows correspond to *Actual* classes and columns to *Predicted* classes, using the class order: **[Mixed, Negative, Positive]**. Across models, **Mixed** is the hardest class and is frequently confused with both **Negative** and **Positive**, consistent with its lower per-class F1. RoBERTa shows the cleanest separation (stronger diagonal), especially improving Mixed, while EmbeddingGemma + XGBoost tends to be conservative on Mixed and shifts ambiguous cases into Negative or Positive.

(A) LSTM

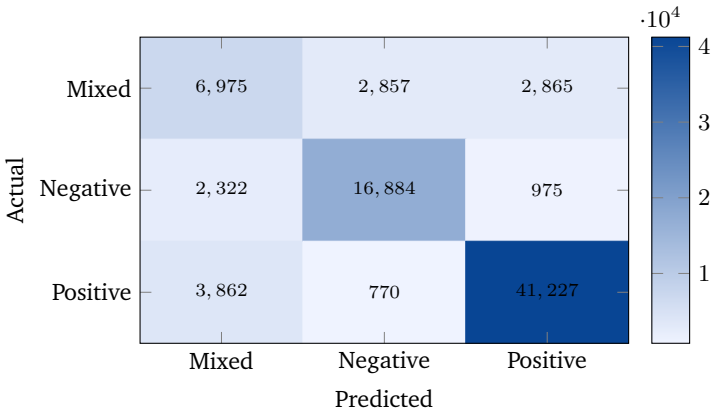

Table 10: Confusion matrix for the LSTM model (test set).

Mixed 16.71% predicted vs. 16.13% actual (+0.59 pp), Negative 26.05% vs. 25.63% (+0.42 pp), Positive 57.24% vs. 58.24% (−1.01 pp). It classifies Positive well, but Mixed is frequently confused with both

Negative and Positive, indicating Mixed is the hardest class. The predicted class distribution slightly overestimates Mixed/Negative and underestimates Positive.

(B) EmbeddingGemma + XGBoost

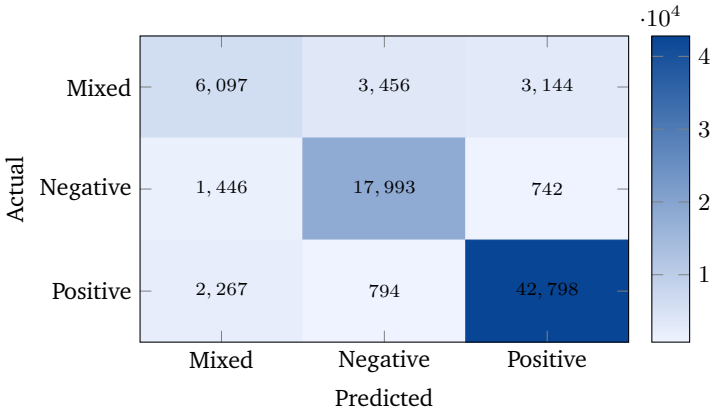

Table 11: Confusion matrix for the XGBoost model (test set).

Mixed 12.46% predicted vs. 16.13% actual (-3.67 pp), Negative 28.25% vs. 25.63% ($+2.62$ pp), Positive 59.29% vs. 58.24% ($+1.05$ pp). It strongly under-predicts Mixed, shifting many Mixed samples into Negative and Positive, and tends to over-predict Negative. Overall, it is conservative on Mixed and biased toward labeling ambiguous cases as Negative.

(C) RoBERTa (fine-tuned)

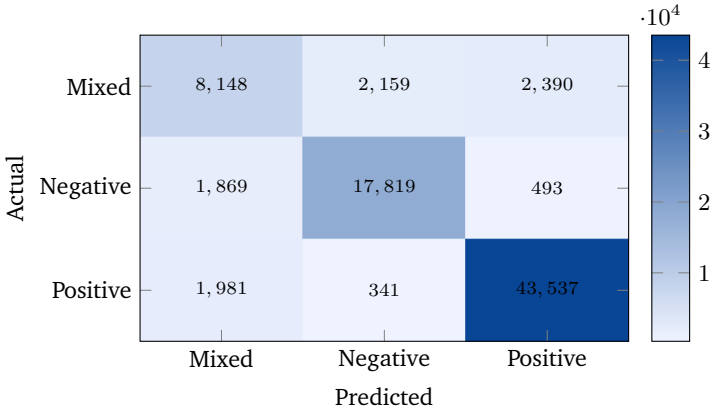

Table 12: Confusion matrix for the RoBERTa (fine-tuned) model (test set).

Mixed 15.24% predicted vs. 16.13% actual (-0.89 pp), Negative 25.81% vs. 25.63% ($+0.18$ pp), Positive 58.96% vs. 58.24% ($+0.71$ pp). It shows the cleanest separation overall, with a stronger diagonal—especially for Mixed—and fewer cross-class confusions than the other models. Its predicted distribution is closest to the true distribution, although Mixed remains the most challenging class.

7 Discussion

7.1 LSTM vs RoBERTa and EmbeddingGemma + XGBoost

True label	Error count	Share of all LSTM errors (%)
positive	3,124	43.03
mixed	2,365	32.58
negative	1,771	24.39
Total	**7,260**	**100.00**

Table 13: Error distribution for the LSTM model on examples where RoBERTa is correct.

True label	Error count	Share of all LSTM errors (%)
positive	2,830	45.45
negative	1,896	30.45
mixed	1,500	24.09
Total	**6,226**	**100.00**

Table 14: Error distribution for the LSTM model on examples where EmbeddingGemma + XGBoost is correct.

Key confusion patterns:

- **True *positive* reviews:** In the RoBERTa comparison, 3,124 true *positives* are misclassified, with 85.95% predicted as *mixed* and 14.05% as *negative*. In the EmbeddingGemma + XGBoost comparison, 2,830 true *positives* are misclassified, with 87.07% predicted as *mixed* and 12.93% as *negative*.
- **True *negative* reviews:** In the RoBERTa comparison, 1,771 true *negatives* are misclassified, with 70.86% predicted as *mixed* and 29.14% as *positive*. In the EmbeddingGemma + XGBoost comparison, 1,896 true *negatives* are misclassified, with 73.05% predicted as *mixed* and 26.95% as *positive*.
- **True *mixed* reviews:** In the RoBERTa comparison, 2,365 true *mixed* reviews are misclassified, with 52.39% predicted as *negative* and 47.61% as *positive*. In the EmbeddingGemma + XGBoost comparison, 1,500 true *mixed* reviews are misclassified, with 52.87% predicted as *positive* and 47.13% as *negative*, yielding an almost 50/50 split in both cases.

On the subsets of reviews where LSTM failed but the stronger baselines succeeded, LSTM's predictions were distributed as follows:

- **RoBERTa-correct subset:**
  - 54.27% predicted as *mixed*
  - 23.11% predicted as *negative*
  - 22.62% predicted as *positive*
- **EmbeddingGemma + XGBoost-correct subset:**
  - 61.82% predicted as *mixed*
  - 20.94% predicted as *positive*
  - 17.23% predicted as *negative*

Across both comparisons, LSTM demonstrates a strong over-reliance on the *mixed* label, applying it to 54.27% of cases in the RoBERTa subset and 61.82% of cases in the EmbeddingGemma + XGBoost subset. This represents a severe miscalibration in handling sentiment nuance: LSTM systematically fails to recognize clearly positive reviews (86–87% of true *positives* incorrectly classified as *mixed*) and similarly fails to recognize clearly negative reviews (around 71–73% of true *negatives* defaulted to *mixed*).

Main issue The primary failure mode of LSTM is **systematic over-prediction of the *mixed* sentiment category**, especially when reviews contain any combination of positive and negative language (caveats, minor criticisms, comparisons, acknowledgments of the other side), regardless of the overall sentiment direction. This behavior is consistent across both diagnostic slices (RoBERTa-correct and EmbeddingGemma + Xgboost-correct), indicating a stable structural weakness rather than a comparator-specific artifact.

Concretely, LSTM exhibits three repeating patterns:

1. **Positive → Mixed collapse**: it misclassifies ≈86–87% of true *positive* reviews as *mixed* when the review contains even minor criticisms, reservations, or balanced discussion alongside an overall positive stance.
2. **Negative → Mixed collapse**: it fails to recognize true *negative* reviews ≈71–73% of the time, defaulting to *mixed* when negative sentiment is expressed through measured, analytical language or specific criticisms rather than purely emotional condemnation, and when the reviewer acknowledges small positives.
3. **Mixed → Forced polarity**: for genuinely mixed reviews, LSTM fails to detect the balanced nature that defines the class and instead splits them nearly evenly into *positive* vs *negative* (roughly 50/50), suggesting it detects opposing signals but cannot represent "balance" as its own decision.

Overall, LSTM appears to treat sentiment as a shallow aggregation of positive/negative tokens, while RoBERTa and EmbeddingGemma + Xgboost successfully capture contextual weighting, rhetorical structure, and overall argumentative flow. LSTM struggles to distinguish:

- "good but with flaws" (positive),
- "strengths and weaknesses are roughly balanced" (mixed),
- "bad despite some redeeming qualities" (negative).

How this problem manifests LSTM's over-reliance on *mixed* predictions manifests in several specific ways:

- **For true *positive* reviews:** LSTM is triggered into predicting *mixed* by any mention of flaws, limitations, caveats, or comparisons, even when these are minor relative to overall praise. Reviews that acknowledge weaknesses while concluding with a clear recommendation are systematically misclassified. The model struggles to separate "positive with honest assessment" from "genuinely balanced mixed."
- **For true *negative* reviews:** When negative reviews use measured or analytical language ("the gameplay is slow and very basic… the biggest problem is…") or acknowledge small positives while maintaining an overall negative stance, LSTM often defaults to *mixed*. It can also be vulnerable to tone or rhetorical tricks (e.g., sarcasm/irony) where surface-level phrasing does not match the underlying sentiment intent.
- **For genuinely mixed reviews:** LSTM fails to recognize balanced sentiment as its own category. Instead, it forces nuanced reviews into binary *positive* or *negative* buckets almost randomly (ap-

proximately 50/50 split). Reviews with explicit balancing cues (e.g., "mixed bag," structured PROS/CONS, or evenly weighted trade-offs) are misclassified because LSTM cannot integrate competing sentiments into a stable "balanced" decision.

The underlying issue is that LSTM detects mixed signals but fails to integrate them into a coherent global assessment, unlike RoBERTa and EmbeddingGemma + Xgboost which successfully navigate the same nuanced cases.

Practical implications

- **Production risk: biased sentiment distribution.** Using the LSTM in a game-review sentiment pipeline would materially distort outputs, especially due to systematic over-prediction of *Mixed* (by roughly 54–62% across both comparisons).
- **False-neutral "sentiment void".** Many genuinely **Positive** reviews (useful for recommendations and conversions) and clearly **Negative** reviews (useful for quality monitoring and alerts) are diluted into an ambiguous middle class that is less actionable.
- **Error composition is business-relevant.** In the XGBoost comparison, about **45%** of the LSTM's errors are missed positives and about **30%** are missed negatives, implying:
  - loved games are under-surfaced (weaker discovery/recommendation signals),
  - problematic releases are insufficiently flagged (weaker QA / trust-and-safety signals).
- **Unreliable aggregate metrics on high-value content.** Sentiment summaries derived from the LSTM become least trustworthy on nuanced, thoughtful reviews—the very content most valuable for product and market insights.
- **Architecture implication.** Since both RoBERTa and EmbeddingGemma + XGBoost handle these same cases better, the LSTM setup here appears ill-suited for document-level sentiment:
  - it likely follows local word-level polarity shifts without integrating them into a stable global judgment,
  - stronger baselines better exploit contextual cues and feature interactions to resolve ambiguity.

7.2 EmbeddingGemma + XGBoost vs RoBERTa

True label	Error count	Share of all XGBoost errors (%)
mixed	2,875	53.51
positive	1,602	29.81
negative	896	16.68
Total	**5,373**	**100.00**

Table 15: XGBoost errors on examples where RoBERTa is correct.

Key confusion patterns:

- **True *mixed* (2,875 errors)**: 57.18% predicted as *negative*, 42.82% as *positive*
- **True *positive* (1,602 errors)**: 76.03% predicted as *mixed*, 23.97% as *negative*

- **True *negative* (896 errors)**: 72.66% predicted as *mixed*, 27.34% as *positive*

On reviews where EmbeddingGemma + XGBoost failed but RoBERTa succeeded, embeddingGemma's predictions were:

- 37.74% *negative*
- 34.79% *mixed*
- 27.47% *positive*

EmbeddingGemma + Xgboost shows a relatively balanced distribution across categories, but critically fails on genuinely *mixed* reviews, misclassifying them as binary sentiments 100% of the time. This represents the model's primary weakness: complete inability to recognize balanced sentiment.

Main issue The primary failure mode of EmbeddingGemma + Xgboost is its **catastrophic inability to correctly identify genuinely *mixed* reviews**, which account for 53.51% of all its errors. Unlike LSTM which over-predicts *mixed*, embeddingGemma does the opposite: it forces genuinely balanced reviews into binary *positive* (43%) or *negative* (57%) categories, unable to recognize when a review expresses balanced criticism or praise. For true *positive* and negative reviews, the model shows a secondary failure pattern of over-predicting *mixed* (76% and 73% respectively), but this is less severe than its complete failure on actual mixed sentiment. The model appears to use a simplistic decision boundary: if it detects both positive and negative signals and one seems slightly dominant, it classifies to that extreme; if signals seem equally balanced, it defaults to *mixed* even when the overall sentiment is clearly positive or negative. This suggests the embedding-based approach captures semantic similarity but lacks the nuanced contextual understanding needed to weigh competing sentiments and detect genuine balance versus qualified positive/negative sentiment.

How this problem manifests

- **For true *mixed* reviews**, EmbeddingGemma + Xgboost completely fails to recognize balanced sentiment, instead forcing every genuinely mixed review into either *positive* or *negative* categories. Reviews with explicit PROS/CONS structure, phrases like "mixed bag," or statements like "not as good as... but still decent" are systematically misclassified. For example, a review about Senran Kagura stating "mechanics are pretty decent... however missions are incredibly boring... not the worst game... but missions structure is boring and repetitive" is classified as *negative* when it's clearly expressing balanced mixed sentiment. Similarly, a detailed review of SWTOR stating "deeply flawed, incredibly addictive... The story is great, but game mechanics are a deal-killer. Cautiously recommend" is misread as *positive* despite explicitly stating mixed feelings. The model cannot detect the rhetorical markers of balanced criticism.
- **For true *positive* reviews**, embeddingGemma defaults to *mixed* when reviews contain any negative language or qualifications, even when the overall recommendation is clearly positive. Reviews stating "pretty good. If you're an old Spyro fan... you will enjoy this" are misclassified as *mixed* because the model detects the qualification "if" and interprets minor reservations as balanced criticism. Re-

views praising games while acknowledging limitations like “good but not perfect” or “I enjoyed it despite some flaws” trigger the *mixed* classification incorrectly.

- **For true *negative* reviews**, embeddingGemma similarly over-predicts *mixed* when negative reviews contain any positive acknowledgments or measured language. A scathing Silent Hill HD Collection review explaining in detail why it’s terrible but framed as an analytical comparison (“if we were reviewing SH2 and 3 you’d see nines and tens, this however...”) is misclassified as *positive*, showing the model’s inability to track negation and contrastive rhetoric. Reviews using phrases like “Flaming pile of ****” but followed by explanatory context are classified as *mixed* when the sentiment is unambiguously negative.

Root cause analysis The underlying issue is that EmbeddingGemma + Xgboost, despite its semantic embedding capabilities, lacks the sophisticated contextual reasoning that RoBERTa’s transformer architecture provides. It cannot properly aggregate competing sentiments, track rhetorical flow, or distinguish between “qualified positive,” “genuinely balanced mixed,” and “acknowledged negative.”

Practical implications

- **High deployment risk for nuanced sentiment.** The model largely fails to correctly identify genuinely *Mixed* reviews; around **53% of all errors** are due to Mixed cases, so it is not reliable when nuance is required.
- **False-binary behavior (loss of nuance).** Instead of producing a false-neutral middle (as with the LSTM), EmbeddingGemma + Xgboost tends to force balanced reviews into **Positive** or **Negative**, collapsing the most informative feedback (both pros and cons) into extreme labels.
- **Impact on developer/publisher insight.** Because mixed reviews often contain the most actionable detail (what works vs. what breaks), this failure directly reduces the usefulness of sentiment analytics for prioritizing fixes and improvements.
- **Noise from class imbalance in predictions.** For **Positive** and **Negative** reviews, the model’s reported **76%** and **73%** over-prediction of *Mixed* adds noise to aggregate reporting, though it is less damaging than the Mixed-class failure itself.
- **Rhetorical/linguistic vulnerability.** Weak handling of **negation** and **contrastive rhetoric** (e.g., “good idea, but terrible execution”) can produce harmful errors, including cases where strongly negative reviews are misclassified as **Positive**.

8 Conclusion

Project Summary This project successfully established an end-to-end pipeline for sentiment analysis in the video game domain. The team constructed a robust benchmark dataset of 629,884 reviews scraped from Metacritic to address the unique linguistic challenges of gaming feedback, such as slang ("nerf," "buff") and sarcasm. Three distinct modeling approaches were implemented and evaluated: a baseline

Bidirectional LSTM, a fine-tuned RoBERTa transformer, and a hybrid approach using EmbeddingGemma + XGBoost classification.

Key Findings The comparative analysis revealed a distinct trade-off between predictive performance and computational efficiency:

- **Best Overall Performance (RoBERTa):** The fine-tuned RoBERTa model proved to be the superior architecture for understanding semantic nuance. It achieved the highest global metrics (Accuracy: 88.27%, Macro F1: 0.8278) and demonstrated the "cleanest separation" between classes. It was particularly effective at identifying "Mixed" reviews, a category where other models struggled significantly.
- **Best Efficiency (EmbeddingGemma + XGBoost):** This hybrid approach offered a strong middle ground. It drastically reduced training time (approx. 9 minutes compared to RoBERTa's $\sim$31 hours) while maintaining competitive accuracy (84.95%). However, it suffered from a "false-binary" behavior, where it failed to recognize nuanced "Mixed" sentiment, forcing those reviews into Positive or Negative categories.
- **Highest Throughput (LSTM):** While the LSTM baseline lagged in predictive quality (Accuracy: 82.66%) and suffered from a "false-neutral" bias (often misclassifying clear opinions as Mixed), it offered the highest inference speed ($\sim$1,913 samples/second), making it potentially useful for high-volume, real-time streams where depth of understanding is secondary.

Final Verdict The study concludes that while dense embedding approaches (EmbeddingGemma + XGBoost) provide a viable, resource-efficient alternative for general sentiment classification, **fine-tuned Transformers (RoBERTa) remain essential for capturing the complex, nuanced opinions** typical of the gaming community. Future work could focus on improving the efficiency of the Transformer model or addressing the specific inability of the embedding-based model to detect balanced, mixed sentiments.

9 Group Contributions & Resources

9.1 Group Contributions

Name	Student ID	Contributions
Au Trung Phong	20225455	Team Leader, Data Preparation, System Refinement
Nguyen Nam Khanh	20225448	LSTM Baseline Implementation, Data Analysis
Vu Duc Minh	20225514	Final Result Analysis
Tran Sy Minh Quan	20225521	RoBERTa Approach Implementation
Bui Van Huy	20225497	Dense Embedding + XGBoost Implementation

Table 16: Group Members and Contributions

9.2 Resources

1. **Source Code:** GitHub Repository
2. **Dataset:** HuggingFace Dataset
3. **Final Models:**
    - LSTM Baseline: Model Link
    - Embedding Gemma + XGBoost: Model Link
    - Finetuned RoBERTa: Model Link

References

[1] Tianqi Chen. Xgboost: A scalable tree boosting system. *Cornell University*, 2016.

[2] Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzemiński, Genta Indra Winata, Saba Sturua, Saiteja Utpala, Mathieu Ciancone, Marion Schaeffer, Gabriel Sequeira, Diganta Misra, Shreeya Dhakal, Jonathan Rystrøm, Roman Solomatin, Ömer Çağatan, Akash Kundu, Martin Bernstorff, Shitao Xiao, Akshita Sukhlecha, Bhavish Pahwa, Rafał Poświata, Kranthi Kiran GV, Shawon Ashraf, Daniel Auras, Björn Plüster, Jan Philipp Harries, Loïc Magne, Isabelle Mohr, Mariya Hendriksen, Dawei Zhu, Hippolyte Gisserot-Boukhlef, Tom Aarsen, Jan Kostkan, Konrad Wojtasik, Taemin Lee, Marek Šuppa, Crystina Zhang, Roberta Rocca, Mohammed Hamdy, Andrianos Michail, John Yang, Manuel Faysse, Aleksei Vatolin, Nandan Thakur, Manan Dey, Dipam Vasani, Pranjal Chitale, Simone Tedeschi, Nguyen Tai, Artem Snegirev, Michael Günther, Mengzhou Xia, Weijia Shi, Xing Han Lù, Jordan Clive, Gayatri Krishnakumar, Anna Maksimova, Silvan Wehrli, Maria Tikhonova, Henil Panchal, Aleksandr Abramov, Malte Ostendorff, Zheng Liu, Simon Clematide, Lester James Miranda, Alena Fenogenova, Guangyu Song, Ruqiya Bin Safi, Wen-Ding Li, Alessia Borghini, Federico Cassano, Hongjin Su, Jimmy Lin, Howard Yen, Lasse Hansen, Sara Hooker, Chenghao Xiao, Vaibhav Adlakha, Orion Weller, Siva Reddy, and Niklas Muennighoff. Mmteb: Massive multilingual text embedding benchmark. *arXiv preprint arXiv:2502.13595*, 2025.

[3] Facebook AI. Facebookai/roberta-base. `https://huggingface.co/FacebookAI/roberta-base`, 2019.

[4] Google. google/embeddinggemma-300m. `https://huggingface.co/google/embeddinggemma-300m`, 2025.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[6] Doaa Mohey El-Din Mohamed Hussein. A survey on sentiment analysis challenges. *Journal of King Saud University-Engineering Sciences*, 30(4):330–338, 2018.

[7] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[8] UB Mahadevaswamy and P Swathi. Sentiment analysis using bidirectional lstm network. *Procedia Computer Science*, 218:45–56, 2023.

[9] Newzoo. The global games market report 2024, 2024. Available at: https://newzoo.com/resources/blog/global-games-market-revenue-estimates-and-forecasts-in-2024.

[10] Arvind Panwar and Vishal Bhatnagar. Sentiment analysis of game review using machine learning in a hadoop ecosystem. In *Research Anthology on Implementing Sentiment Analysis Across Multiple Disciplines*, pages 463–483. IGI Global Scientific Publishing, 2022.

[11] Stackla. The consumer content report: Influence in the digital age, 2019. Available at: https://www.nosto.com/wp-content/uploads/2019/02/Stackla-Consumer-Marketer-Data-Report-2019_FINAL.pdf.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[13] Henrique Schechter Vera, Sahil Dua, Biao Zhang, Daniel Salz, Ryan Mullins, Sindhu Raghuram Panyam, Sara Smoot, Iftekhar Naim, Joe Zou, Feiyang Chen, et al. Embeddinggemma: Powerful and lightweight text representations. *arXiv preprint arXiv:2509.20354*, 2025.

[14] Markos Viggiato, Dayi Lin, Abram Hindle, and Cor-Paul Bezemer. What causes wrong sentiment classifications of game reviews? *IEEE Transactions on Games*, 14(3):350–363, 2021.

[15] Nick Wingfield. High scores matter to game makers, too. *The Wall Street Journal*, 2007. Available at: https://www.wsj.com/articles/SB119024844874433247.