Sentiment Analysis Based on LSTM of Top 20 Play Store Apps Reviews

1st Shenyi Tang

***Abstract*—This paper presents two sentiment analysis models based on LSTM for mobile app reviews. The initial and basic model combines LSTM-based text processing with app-specific features. Based on the performance of the first model, the enhanced version introduces several key improvements: an attention mechanism for better context understanding, SMOTE oversampling for class imbalance, and “FocalLoss” with class weights. Architectural enhancements include deeper bidirectional LSTM layers, increased embedding dimensions, and sophisticated optimization strategies using “AdamW” with “OneCycleLR” scheduling.**

***Index Terms*—Sentiment Analysis, LSTM, Imbalanced Data, SMOTE oversampling**

I. Introduction

Sentiment analysis has become a valuable tool across numerous industries, playing a critical role in social media listening, email feedback analysis, customer support ticket prioritization, product review mining, and personalized marketing. By extracting customer reviews and comments, businesses can gain insights into customer experiences, predict trends, and make data-driven decisions to improve customer satisfaction and engagement.

Given the significant impact of sentiment analysis in practical applications, substantial research efforts have been devoted to improving its accuracy and robustness. BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained language model that utilizes bidirectional context learning, enabling simple fine-tuning for diverse NLP tasks with state-of-the-art results [1]. Lopes et al. proposed an AutoML-based fusion approach for text and image sentiment analysis, achieving 95.19% accuracy on the B-T4SA dataset.” [2]. By incorporating Transformer-XL architecture, Yang et al. proposed XLNet, overcoming BERT’s limitations of masked input dependency while outperforming it across 20 NLP tasks [3].

This project presents an innovative approach to sentiment analysis that combines traditional text processing with app-specific features, addressing the unique challenges of analyzing mobile app reviews.

II. The Basic Model

A. Dataset Description and Data Cleaning

The Original data contains three fields: `content` refers to users’ reviews, `score` refers to users’ ratings, `app` refers to apps’ names. As the original score ranges from 1-5, I simplify the problem into a three-class classification by mapping `score: 1, 2` to -1 (negative), `score: -3` to 0 (neutral) and `score: 4, 5` to 1 (positive). What’s more, I filtered out non-English reviews to make the problem less complex by identifying whether the characters have valid ASCII code, but retained the emojis in the content. Hence, the final dataset used for training is illustrated as Tab. I below:

TABLE I
Data Dictionary

Field	Data Type	Description
content	string	content of users review
score	integer	From 1 to 5
flag	integer	$-1, 0, 1$
cleaned_review	string	content after cleaning

B. Problem Formulation

The sentiment analysis task can be formulated as a supervised multi-class classification problem where the goal is to map app reviews to sentiment categories. Given a set of N reviews $\{x_i, a_i, y_i\}_{i=1}^{N}$, where:

$$\begin{aligned} x_i \in R^L &: \text{represents the review text sequence} \\ &\quad\ \text{of length } L \\ a_i \in R^m &: \text{represents the app-specific feature} \\ &\quad\ \text{vector of dimension } m \\ y_i \in \{0,1,2\} &: \text{represents the sentiment label} \\ &\quad\ \text{(negative, neutral, positive)} \end{aligned} \tag{1}$$

The objective is to learn a function $f : (X, A) \to Y$ that achieving the trade-off between maximizing the classification accuracy while maintaining robust performance across all sentiment classes (minimizing the loss function).

C. Model Architecture

The first step is text embedding. Each review text x_i is tokenized into a sequence of tokens $[w_1, w_2, \ldots, w_l]$, where l is the maximum sequence length. Each token w_i is then mapped to a dense vector $e_j \in R^d$ through an embedding layer: $E(w_j) = e_j,\ where\ E \in R^{V\times d}$ where V is the vocabulary size and d is the embedding dimension (100 in this implementation).

Additionally, as emojis are supposed to carry strong sentiment signals in modern digital communication, all the emojis were preserved in the text processing.

While “App” features are encoded through One-hot encoding: $O(a_i) = [o_1, o_2, \ldots, o_m]$, where $o_i \in \{0, 1\}$.

Then the embedded sequence is processed through a 2-layer LSTM: $h_t = LSTM(e_t, h_{t-1})$, where h_t represents the hidden state at time t. After that, I concatenated the text features processed by LSTM and app features followed by 2 fully connected layers to learn feature interactions. Both of the fully connected layers use `ReLu` to do the non-linear activation, between which there is a dropout layer, randomly dropping a proportion of neurons with probability $p = 0.5$, to prevent over-fitting.

The model is trained to minimize the cross-entropy loss function:

$$L = \sum_{i=1}^{N} y_i \times \log(\hat{y}_i) \tag{2}$$

D. Results

As Fig. 1 shown, the model exhibits fascinating dynamic changes in its performance characteristics. The training can be clearly divided into three phases: initial phase (rounds 1-5), rapid learning phase (rounds 6-10), and stable optimization phase (rounds 11-30).

Round 6 marks a significant turning point as the model enters the rapid learning phase. The training loss drops dramatically from 0.7197 to 0.6469, with the test loss correspondingly decreasing from 0.7183 to 0.5410. This sudden performance improvement likely occurred because the model discovered key feature patterns in the data. In the subsequent rounds 7-10, the model continues to improve rapidly, with training loss further decreasing to 0.4900 and test loss dropping to 0.4851.

From round 11 onwards, the training loss continues to decrease gradually, while the test loss begins to show fluctuations, generally maintaining between 0.48-0.52. The possible explanations are that the model might be starting to overfit to the training data, or the model's performance is not consistent across all types of input data etc.

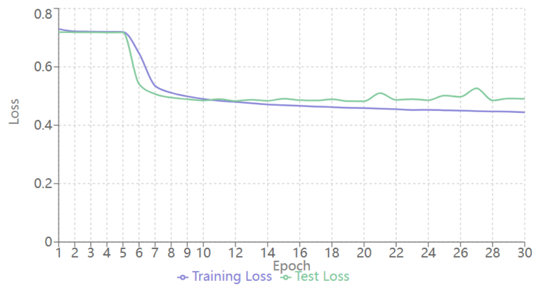

Fig. 1. Training and Testing Curve

Looking at the final classification performance, the model shows significant variations across the three categories. For positive reviews (category 2), the model performs excellently, achieving a 95% recall rate and 86% precision. For negative reviews (category 0), the model shows moderate performance with a 72% recall rate and 78% precision. However, for neutral reviews (category 1), the model performs poorly with an F1 score of just 0.01. This severe performance imbalance primarily stems from the uneven distribution of training data: 27,050 positive reviews, 10,181 negative reviews, but only 2,107 neutral reviews.

TABLE II
CONFUSION MATRIX OF MODEL I

	0	1	2
0	7294	14	2873
1	731	9	1367
2	1383	4	25663

TABLE III
OVERALL AND CLASSIFICATION REPORT OF MODEL I

	Precision	Recall	F1-Score	Support
0	0.78	0.72	0.74	10181
1	0.33	0.00	0.01	2107
2	0.86	0.95	0.90	27050
overall	0.81	0.84	0.81	39338

Overall, the model demonstrates satisfactory general performance, with an accuracy of 83.80% being acceptable in practical application scenarios. However, its poor handling of neutral reviews represents a significant weakness that needs to be addressed especially in the terms of imbalanced data.

III. THE IMPROVEMENT BILSTM MODEL

A. Model Improvements

Based on the output of the Basic LSTM Model, I made some modifications especially for the poor performance in imbalanced data.

First, the model introduces Focal Loss to address class imbalance issues. While traditional cross-entropy loss treats all samples equally, Focal Loss dynamically adjusts the weights of easy and difficult samples to optimize the learning process. The mathematical expression is:

$$Fl(pt) = -\alpha(1 - pt)^{\gamma} \log(pt) \tag{3}$$

where pt is the model's prediction probability for the correct class, α is the class weight, and γ is the modulating factor. This design enables the model to automatically focus on hard-to-classify samples while reducing the impact of easy samples. For our sentiment analysis task, this improvement is particularly effective as it helps the model better learn features of minority classes such as neutral reviews.

In terms of neural network architecture, the new model implements a Bidirectional LSTM (BiLSTM) structure. Compared to unidirectional LSTM, BiLSTM processes sequence data from both forward and backward directions simultaneously, allowing the hidden state at each time step to capture complete contextual information.

The attention mechanism introduced in the model. The attention layer design allows the model to dynamically focus on important parts of the input sequence. Specifically, for the

LSTM output sequence $H = [h_1, h_2, ..., h_T]$, the attention weights are calculated as:

$$\begin{aligned} e_t &= tan^{-1}(Wh_t + b) \\ \alpha_t &= softmax(e_t) \\ c &= \sum \alpha_t h_t \end{aligned} \tag{4}$$

where W is a learnable weight matrix, $alpha_t$ is the attention weight for time step t, and c is the final context vector. This mechanism enables the model to adaptively determine the importance of each word based on context, particularly suitable for identifying key words in sentiment analysis.

For training strategy, the model implements the OneCycleLR learning rate scheduler, which increases and then decreases the learning rate, helping the model converge quickly to better local optima. Additionally, gradient clipping and gradient norm monitoring mechanisms are introduced to effectively prevent gradient explosion problems.

The new version of data processing maintains the advantage of emoji processing while addressing class imbalance through SMOTE oversampling technique to get the training dataset. This method balances data distribution by generating synthetic samples of minority classes in feature space, represented as:

$$x_{new} = x_i + \lambda(x_k - x_i) \tag{5}$$

where x_i is a minority class sample, x_k is one of its k-nearest neighbors, and λ is a random number between 0 and 1.

Furthermore, the model includes a complete checkpoint saving and restoration mechanism, encompassing model states, optimizer states, and learning rate scheduler states, greatly improving the robustness and recoverability of model training.

B. Results

Shown as Fig. 2, the improved model demonstrates faster convergence and lower initial loss values. The training loss consistently decreased from 0.5544 in the first epoch to 0.2400 in the final epoch, showing a stable learning trend. This is significantly lower than the original model's initial loss of 0.7297, indicating enhanced feature extraction capabilities from the start.

During the first 10 epochs, the test loss maintained good consistency with the training loss, decreasing from 0.4697 to 0.2937, demonstrating good generalization capability. However, starting from epoch 11, the test loss began showing noticeable fluctuations, particularly in the later stages (epochs 25-30) where significant oscillations occurred, reaching as high as 1.8120. This phenomenon suggests potential overfitting in the later stages, despite the implementation of regularization techniques such as Dropout and batch normalization.

According to the confusion matrix and other metrics, hhe improved model's overall accuracy is 78.16%, lower than the original model's 83.80%. However, this apparent decline masks an important improvement: the new model shows marked progress in handling neutral reviews (class 1). The F1 score increased from 0.01 to 0.12.

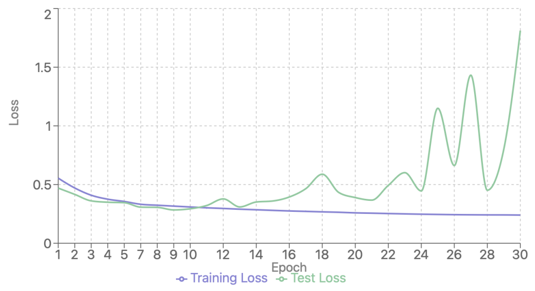

Fig. 2. Training and Testing Curve of Model II

TABLE IV
CONFUSION MATRIX OF MODEL II

	0	1	2
0	6952	740	2489
1	686	289	1132
2	1733	1813	23504

TABLE V
OVERALL AND CLASSIFICATION REPORT OF MODEL II

	Precision	Recall	F1-Score	Support
0	0.74	0.68	0.71	10181
1	0.10	0.14	0.12	2107
2	0.87	0.87	0.87	27050
overall	0.79	0.78	0.79	39338

Overall, there is a distinct improvement in model's ability to identify minor categories, but seems at the expense of overall accuracy and class-specific performance.

IV. GitHub Repo

Shenyi Tang's STATS 507 GitHub Repo Link

References

[1] J. Devlin, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[2] V. Lopes, A. Gaspar, L. A. Alexandre, and J. Cordeiro, "An automl-based approach to multimodal image sentiment analysis," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–9.

[3] Z. Yang, "Xlnet: Generalized autoregressive pretraining for language understanding," *arXiv preprint arXiv:1906.08237*, 2019.