Enhanced Text Communication System Using Sentiment-Based Prompting for End-User

Deepshikhar Tyagi[a], Ishwer Kumar[a], Manish Kumar Singh[a], Rishabh Mehrotra[a], Sanket Jain[a], Shubham Saha[a] and Yuvasree Pamujula[a]

[a]Indian Institute of Science, Bengaluru, India
deepshikhart@iisc.ac.in, ishwerkumar@iisc.ac.in,manishsingh@iisc.ac.in,
rishabhmehro@iisc.ac.in,sanketjain@iisc.ac.in, shubhamsaha2@iisc.ac.in,yuvasreep@iisc.ac.in

Abstract. *This paper introduces a Chat Application in a Web-UI focusing on Sentiment Analysis. The idea is to enhance user experience for text based communication mediums by providing rich emotion data in real-time. The application utilizes - Fine-tuned MobileBERT Model with BiLSTM for sentiment inferencing, LLM calls for response suggestions to the user, RAG context retriever for relevant past messages to feed to LLM for appropriate responses, multivariate distributions on a buffer of past messages for sentiment analytics and BART model based paraphrasing to improve user responses.*

1 Introduction

Problem Definition: Modern chat applications are increasingly being relied upon for both personal and professional communication. However, these platforms often fail to convey the emotional tone behind textual content, leading to ambiguity and misinterpretation. This issue arises due to the absence of non-verbal cues such as facial expressions or vocal tone. Current solutions attempt to mitigate this problem using emojis or multimedia but still fall short in many real-time scenarios.

This project addresses the core challenge of sentiment misalignment in textual communication. It aims to enhance text-based chats by analyzing sentiment in real time and assisting users in crafting responses that are contextually and emotionally appropriate.

Problem Analysis: Most chat apps rely on static UI features (e.g., emojis, templates) to help users clarify tone. These fail in dynamic conversations or emotionally sensitive contexts. This motivates the need for intelligent, real-time, context-aware sentiment assistance.

2 Methodology / System Architecture

2.1 Overview

The system is designed as a modular architecture that integrates natural language understanding, sentiment detection, and large language model (LLM) capabilities into a real-time chat interface. The core idea is to interpret the sentiment and context of each incoming message and offer appropriate, emotionally aware suggestions before the user replies.

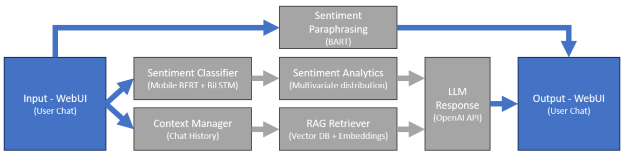

Sentiment Paraphrasing (BART)
Input - WebUI (User Chat)
Sentiment Classifier (Mobile BERT + BiLSTM)
Sentiment Analytics (Multivariate distribution)
Context Manager (Chat History)
RAG Retriever (Vector DB + Embeddings)
LLM Response (OpenAI API)
Output - WebUI (User Chat)

Figure 1: System Architecture Diagram: Flow from chat input to prompt suggestion

2.2 Component Descriptions

2.2.1 Sentiment Classifier

The sentiment analysis model uses HuggingFace's MobileBERT architecture, fine-tuned for classification in 6 different emotions (joy, sadness, angry, love, fear and surprise). The model outputs a class label based on input text. No regression head is used. Sequence modeling is done on these messages using a BiLSTM on the output of MobileBERT that allows emotion prediction of input text incorporating context from previous messages.

2.2.2 Sentiment Analytics

Message sentiments are converted to meaningful analytics using using multivariate normal distributions. Below data is generated:

- User's & Group's emotional state based on recent messages
- User's & Group's emotional state based on entire history
- User's & Group's emotional state trend plot
- User's emotional volatility

In B2B and B2C use-cases, the analytics can be used to make business decisions such as PR evaluations, customer profiling, service agent evaluation, etc.

2.2.3 Sentiment based Paraphrasing

The module uses BART (encoder-decoder) transformer model to paraphrase the user's messages into a positive or neutral tone. The model is trained using a small LLM generated dataset and further finetuned on a larger non-labeled dataset using Reinforcement Learning. Paraphrasing module can be a low cost and privacy sensitive replacement for LLM

2.2.4 *Context Manager*

Chat context is maintained dynamically by storing previous user messages, sentiment detected for each message and the most recent message.

2.2.5 *RAG Context Retriever*

The system integrates Retrieval-Augmented Generation (RAG) using a Chroma vector store and sentence-transformer embeddings (all-MiniLM-L6-v2). Incoming user messages are used to retrieve semantically similar historical chat entries or relevant past examples.

2.2.6 *LLM Prompt Generator*

Retrieved messages from RAG in the form of - last message, own messages & other's messages, along with the last message emotion are passed to a LLM in a prompt to generate user response suggestions.

2.2.7 *Chat Interface and Backend*

The backend is built using FastAPI that exposes endpoints for:

- Performing sentiment analysis using the fine-tuned classifier
- Generating suggestions using an LLM based on current message + chat history

The back-end takes care of all pre-processing steps, along with response handling and feeds it back to the front-end UI. The front-end displays the chat interface, sentiment tags, and suggestions in real time.
Inferencing and data formatting is done in the back-end before calling model functions in order to keep the payload light, thus allowing faster response times.

3 Implementation Details

3.1 *Sentiment Classification*

This section details our two–stage training recipe, the datasets, the evaluation protocol, sequence modeling using BiLSTM and how the models adapt from Twitter prose to chat-style conversation.

3.1.1 *Two-Stage Training Pipeline*

Figure 2. visualises the workflow. **Stage 1** fine-tunes eight BERT-family backbones—each in three variants (MLP, LoRA, full)—on Tweet-Emotion. A validation leaderboard retains the six best weighted-F_1 models. **Stage 2** evaluates those checkpoints zero-shot on Chat-2k (*pre*), then runs a three-epoch incremental update (*post*).

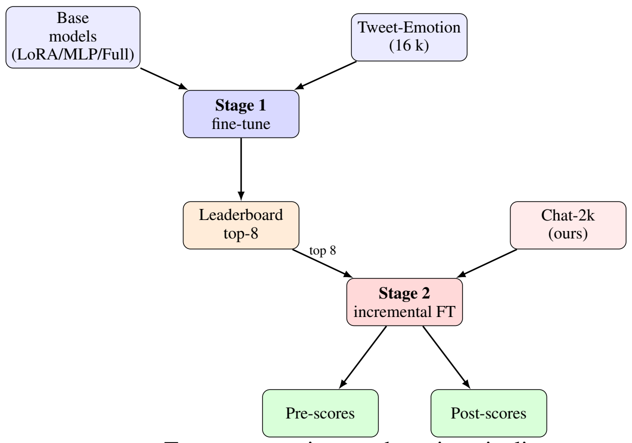


Figure 2: Two-stage sentiment adaptation pipeline.

3.1.2 *BiLSTM Training and Inference*

Sequential flow of messages in a texting application contain temporal information that should be utilised to predict emotion better. For this purpose, a BiLSTM neural net is appended to the trained BERT model. This provides a way to capture messages in a sequence, and embed each message's context to derive the final message's emotion. BiLSTM is selected over a Transformer model since it is lightweight, can be trained with limited data, and suits the application given hysteresis over only the last few messages is sufficient.
Example
"I just got fired from my job."
"But it's fine, I never liked it anyway."
"Best day ever."
The last message may be predicted as "joy" using BERT inferencing in isolation, but with a BiLSTM added context from previous messages will indicate that the predicted emotion is closer to "sadness". This is a classic example of sarcasm that is difficult to capture when inferencing on a single message.

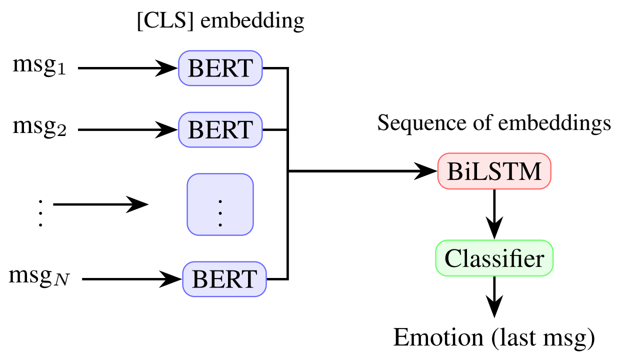


Figure 3: Two-stage sentiment adaptation pipeline.

3.2 *Sentiment Analytics*

The backbone of sentiment analytics is multivariate normal distributions. Each identities (user/group) emotion state is maintained through a multivariate normal distributions. Each distribution is expressed using a 6 x 1 dimension μ vector and 6 x 6 dimension Σ matrix.
As an optimisation, the distribution for entire conversation history is updated using Welford's method, thereby removing the need to maintain entire history of emotion vectors.

For the distribution relevant to each buffer, the distribution is recalculated from the stored $history$ everytime the user sends a new message.
The pseudocode for the implementation is shared below:

Algorithm 1 SentimentAnalytics Function

```
Require: Chat History (min. 1) chat_history, buffer sizes (min. 1) buffers
Ensure: Updated user emotion distribution user_emo_dist
 1: Extract last message and username from chat_history
 2: Update user's $\mu$ and $\Sigma$ for entire history using Welford's method
 3: Compute $CV = \frac{\sigma}{\mu}$
 4: Update history buffer with new emotion vector while maintaining max buffer size
 5: for each buffer size $k$ in buffers do
 6:     Compute $\mu_k$ = mean over last $k$ emotion vectors
 7:     Compute $\Sigma_k$ = covariance over last $k$ vectors
 8:     Compute coefficient of variation $CV_k = \frac{\sigma_k}{\mu_k}$
 9: end for
10: Set/Reset volatile flag using $(CV_k)$ for largest k
11: Update $\mu$ and $\mu_k$ for __group__ by averaging across users
12: return user_emo_dict        ▷ contains user, group analytics
```

3.3 *Sentiment based Paraphrasing*

Sentiment paraphrasing requires the input text to be rephrased with a positive/neutral tone. A suitable architecture for this is encoder-decoder transformers. Based on experiments between variants of BART and T5, BART-base was finalized. The training of paraphrasing model was performed in 2-stages as shown in Figure 4a.

3.3.1 *Stage-1 Training*

Stage-1 is a supervised finetuning approach. The pre-trained BART model `facebook/bart-base` is loaded and trained on the LLM generated Paraphrase-3k dataset (7.3). The loss function used is token level cross-entropy loss

$$\mathcal{L} = -\sum_{t=1}^{T} \log P(y_t \mid y_{<t}, \mathbf{x}) \tag{1}$$

Where:

- $\mathbf{x}$ is the input sequence.
- $y_{<t}$ are the previously generated tokens.
- y_t is the target token at time step t.
- $P(y_t \mid y_{<t}, \mathbf{x})$ is the probability of generating y_t given the input and past outputs.

3.3.2 *Stage-2 Training*

Stage-2 is a Reinforcement Learning approach that allows finetuning on a larger dataset in the absence of labeled data, i.e. paraphrased version of the input text. It involves the use of a DistilBERT sentiment classification model as the reward model for finetuning the BART model (policy model). Refer Fig. 4b. The BART generated text is passed through the DistilBERT model to get classification scores and output embedding vectors for reward calculation as mentioned in 5. For stage-2 trainings of BART model, two implementations were tried as below:

1. A simple policy+reward model training: loss function of BART was chosen as `-reward` from DistilBERT
2. RL using PPO algorithm
   - Policy model: BART
   - Reference model: BART
   - Reward model: DistilBERT (classification + similarity)
   - Value model: DistilBERT+linear_layers

3.4 *RAG*

Vector Store ChromaDB is used with cosine similarity.

RAG The RAG model was tested using `as_retriever`, `similarity_search_with_score`, and cosine similarity for vector database storage, with `SIMILARITY_THRESHOLD` values of 0.9, 0.95, and 0.8. The final setup with a threshold of 0.8 improved retrieval precision and response relevance, as evaluated using both relevant and irrelevant queries.

3.5 *LLM*

The LLM module is implemented as a backend Python service that programmatically builds prompts from the most recent message, detected emotion, and conversation context, tried with Zero-Shot and Few-Shot Prompting. It supports integration with OpenAI, Gemini, Mistral, HuggingFace, and Ollama models, with the provider chosen by configuration. The service calls the selected LLM API and parses the generated output to produce up to three concise, empathetic reply suggestions, following formatting rules for brevity and anonymity. The implementation is modular, allowing for easy extension to new model providers and seamless switching between cloud and local inference. An attempt was made to implement LLM Guardrails using an XML-based schema to enforce input and output constraints—such as message safety, sentiment validity, and exclusion of personal or sensitive information—using the Guardrails library. This mechanism was designed to validate both user inputs and LLM outputs against ethical, privacy, and length requirements before delivering suggestions to end users. This validation pipeline is not present in the codebase, practical integration challenges prevented full deployment, and the Guardrails component is not currently active in production.

4 Experiments and Results

4.1 *Sentiment Classification*

4.1.1 *Metrics*

: In Annexure

4.1.2 *Results*

: In Annexure

4.1.3 *Observations & model choice*

Insights.

1. **Severe domain loss.** Models retain on average only **28%** of their Twitter F_1 when evaluated on chat data without further training.
2. **Incremental fine-tune pays off.** A short three-epoch update recovers ≈46% of the lost performance, pushing BERT-*large* from 0.30 to 0.68 F_1.

3. **Size vs speed.** TinyBERT-6L (68 M params) reaches 0.60 F_1—only 0.08 below BERT-*base*—but is $> 3\times$ faster on CPU, making it an attractive deployment target.
4. **Adapter efficiency.** A DistilBERT LoRA adapter (just 2 M trainable weights) retains 35% out-of-domain accuracy and recovers another 33% after tuning, confirming that cheap, on-device updates are viable.
5. **Why two stages?** Training *only* on the small 2 k-chat set would have been unstable; pre-training on sentiment-rich Twitter texts provides robust lexical priors, while Stage-2 injects conversational pragmatics—striking.

Final-Selected model -MobileBERT

Now we tested our final candidate models TinyBERT-6L , MobileBERT and ALBERT-base and from Refer Table 4. contains the probability for each model and it shows the MobileBERT is more confident in its prediction and is accurate while TinyBERT-6L also have correct prediction but the probabilities are low and is unstable and ALBERT-base classified wrong, 2 of the sentences so based on this analysis we finalized MobileBERT to be our base model for sentiment analysis of the user queries.

4.1.4 *Sequence Modeling using BiLSTM*

Training the BiLSTM model was done using sequence of messages (length 6-8 with padding before input to model), with classified emotion (ground truth) for the last message in the sequence. The messages are fed in to the fine-tuned BERT model, and the embeddings are sequentially passed through BiLSTM.
Training metrics as seen in **Table 9** are average due to the lack of clean data-sets, but inferencing still works for basic message sequences.
Synthetic datasets are generated using a LLM in two ways:

1. Generator function: Some messages are split into multiple lists and sequences are randomly generated (Claude)
2. Organic data: Data is directly provided by LLM (6000 sequences) (GPT4o)

With a coherent flow of messages in organic data from LLM as compared to the generator function, the model performs better post training.
An example is given in in section
Evidently, with the context of previous messages used for classification allow for a better sentiment prediction.

4.2 *Sentiment based Paraphrasing*

4.2.1 *Model Choice*

Experiments were performed with T5 and BART models. T5 model has been majorly pre-trained for English-German translation task. As a result the tuning for paraphrasing did not show good results. In some cases, the T5 model was generating translated texts as paraphrased versions. BART is trained for a wide variety of tasks and performed much better thus making it the default choice for further experiments

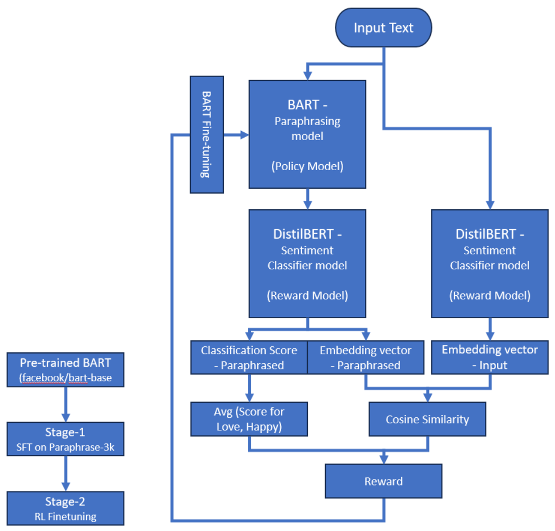
4.2.2 *Evaluation*

Three evaluation schemes are explored:

1. Sentiment Classification: Generated response should have a positive sentiment
2. BERTScore: High context similarity between input and paraphrased texts
3. LLM Evaluation: An LLM is provided the input and paraphrased texts and asked to evaluate naturalness and relevance

Refer Table 7 to check the distribution of sentiment classification for paraphrased text. We see most of the texts have a positive tone. Refer Table 8 to check the evaluation results for BERTScore and LLM Evalutaion. We see that LLM rates the responses high on *Naturalness* and medium on *Relevance*. Overall BERTScore of *0.584* shows the texts have similarity but are not the exact same. Refer Table 11 to see sample outputs.

Team Member	Contribution Area
Deepshikhar Tyagi	• **Architecture Design**: Initiated the architecture design for the application • **Sentiment Analytics Engine** (2.2.2): Created the multivariate distribution based analytics method to convert per message sentiments into user and group emotion states and trend plots • **Sentiment based Paraphrasing** (3.3): Created the module that paraphrases the input texts with a positive sentiment – Worked on data creation using LLM – Devised and performed 2-stage training (Supervised Finetuning + Reinforcement Learning finetuning) – Prepared train, inference and evaluation scripts • **Sentiment Classification**: (2.2.1) – Prepared train, test, inference scripts – Performed stage-1 trainings for MobileBERT and DistilBERT models
Ishwer Kumar	• Integrated the UI template with real-time chat functionality using Supabase • Developed sentiment analysis components for the user interface • Created an initial proof-of-concept with dummy sentiment data and Gemini LLM integration • Designed and incorporated relevant charts for data visualization • Connected all backend APIs to the frontend interface • Worked on LLM evaluation using the LLM-as-judge methodology
Manish Kumar Singh	• **Two-stage domain adaptation**: pre-train on *Tweet-Emotion* (16 k) then apply a 3-epoch incremental fine-tune on *Chat-2k*, recovering 46 % of the zero-shot F1 loss. • **Comprehensive model sweep**: evaluated 8 BERT-family backbones in three modes (MLP, LoRA, full) for 24 distinct checkpoints, giving a controlled view of architecture vs. tuning strategy. • **Cross-domain performance audit**: quantified a 72 % weighted-F1 drop when moving fro m Twitter prose to chat conversations, emphasising the need for conversational adaptation. • **Efficient recovery with minimal training**: showed that three extra chat epochs lift MobileBERT from 0.23 $\rightarrow$ 0.57 F1 and BERT-large to 0.68, avoiding full retraining costs. • **Deployment-oriented trade-off study**: reported parameters, chat F1, and CPU throughput; identified MobileBERT (25 M params, 6.3 steps/s, 0.57 F1) as the “elbow” for real-time use. • **Adapter viability on edge devices**: demonstrated that a DistilBERT LoRA adapter (2 M trainable weights) preserves 35 % zero-shot accuracy and regains another 33 % after tuning.
Rishabh Mehrotra	Worked on LLM Part which incudes - Designed and implemented the LLM prompt generation(Zero-Shot and Few-Shot) and response module supporting OpenAI, Gemini, HuggingFace, and Ollama models, - Integrated environment variable and configuration-based switching between cloud and local LLM providers, Prototyped and tried Guardrails for input and output validation to enforce safety, privacy, and ethical constraints on LLM interactions, - Contributed to robust error handling and modular backend design for seamless extension to future model providers, - Collaborated on ensuring system-generated replies are concise, anonymized, and emotionally supportive
Sanket Jain	• Worked on lightweight language models, including Phi-2 and Falcon RW, to generate context-aware responses for chat messages.
Shubham Saha	• **BiLSTM Model in Sentiment Classification** Trained and added a BiLSTM model on top of the fine-tuned MobileBERT model. This completes the chat application using sequence modeling for emotion classification, improving performance since previous message context is also considered for latest message emotion classification. • **Backend** Developed the complete back-end from scratch, using FastAPI to create various end-points integrating Sentiment Inferencing, Analysis, RAG & Sentiment Paraphrasing. The API service is hosted using ’Uvicorn’ and forwarded using ’Ngrok’. • **RAG Vector Store** Created the context manager using ChromaDB for storing incoming messages via the RAG API, for efficient storage and retrieval for further processing.
Yuvasree Pamujula	• Chroma DB Integration and Vectorization: Developed a vector database using Chroma DB to store and retrieve chat messages efficiently. Utilized MiniLM from HuggingFaceEmbeddings to convert chat messages into vector embeddings and explored various similarity configurations, identifying cosine similarity as the most effective. Also added functionality to clear and reset the database to support iterative testing and refinement. • RAG Pipeline Integration: Developed a Retrieval-Augmented Generation (RAG) module by integrating Chroma DB for efficient context retrieval. Implemented `as_retriever` and `similarity_search_with_score` methods to fetch relevant information based on user queries. • Context Relevance Enhancement: Identified and addressed the issue of irrelevant context being returned for unrelated queries. Introduced similarity score thresholds (0.9 and 0.8) within the retrieval logic to filter out low-relevance results, significantly improving the precision and contextual accuracy of the RAG system.


Input Text
BART Fine-tuning
BART - Paraphrasing model (Policy Model)
DistilBERT - Sentiment Classifier model (Reward Model)
DistilBERT - Sentiment Classifier model (Reward Model)
Classification Score - Paraphrased
Embedding vector - Paraphrased
Embedding vector - Input
Avg (Score for Love, Happy)
Cosine Similarity
Reward
Pre-trained BART (facebook/bart-base
Stage-1 SFT on Paraphrase-3k
Stage-2 RL Finetuning

(a) Training Process (b) RL Finetuning Process (Stage-2)

Figure 4: a shows the overall training procedure, and b details the RL finetuning process.

5 Sentiment based Paraphrasing - Equations

Given:

- x : the original input text
- $\hat{y}$: the generated response
- $H_x \in \mathbf{R}^{T_x \times d}$: hidden states from DistilBERT for x
- $H_{\hat{y}} \in \mathbf{R}^{T_{\hat{y}} \times d}$: hidden states from DistilBERT for $\hat{y}$
- $\bar{h}_x = \frac{1}{T_x} \sum_{t=1}^{T_x} H_x^{(t)}$: mean-pooled embedding for x
- $\bar{h}_{\hat{y}} = \frac{1}{T_{\hat{y}}} \sum_{t=1}^{T_{\hat{y}}} H_{\hat{y}}^{(t)}$: mean-pooled embedding for $\hat{y}$
- $p = f(\bar{h}_{\hat{y}})$: predicted probability vector over emotion classes (e.g., output of softmax)

$$r = 0.5 \cos(\bar{h}_x, \bar{h}_{\hat{y}}) + 0.5 \left(\frac{p_{i_h} + p_{i_l}}{2} \right)$$

Where:

$$\cos(\bar{h}_x, \bar{h}_{\hat{y}}) = \frac{\bar{h}_x \cdot \bar{h}_{\hat{y}}}{||\bar{h}_x|| \cdot ||\bar{h}_{\hat{y}}||}$$

The cosine similarity over DistilBERT embeddings is same as generating BERTScore.

6 BiLSTM examples

Example for BiLSTM based classification and performance

1. "I had the strangest dream last night."
2. "What happened?"
3. "I saw my parents"
4. "Oh wow"
5. "They crossed the line this time!"

Model	**Predicted Emotion**
BERT	Fear
BERT+BiLSTM	Anger

7 Datasets

7.1 *Tweet-Emotion (Stage 1).*

16,000 English tweets labelled with the six emotions {*anger, fear, joy, love, sadness, surprise*}. We use the cleaned release from the Tweet-Emotion corpus. [1] The corpus was gathered via emotion–specific hashtags, then pre-processed (user mentions, URLs and hashtags removed, lower-cased). We adopt a **70 / 15 / 15 %** train / validation / test split (16,000 / 2,000 / 2,000 tweets).

7.2 *Chat-2k (Stage 2).*

2 000 two-to-four-turn snippets generated with `claude-opus-4`. Prompts enforce (i) conversational style, (ii) exactly one target emotion per snippet, (iii) lexical diversity (slang, code-switching, sarcasm). Ten per-cent were manually sanity-checked.

7.3 *Paraphrase-3k*

3500 samples are generated for sentiment based paraphrasing dataset using `claude-sonnet-4`. This dataset is used for supervised finetuning of paraphrasing model. The dataset consists of input texts and their paraphrased versions with a neutral or positive tone. The LLM is instructed to randomly select the target sentiment for the paraphrased

7.4 *Sentiment Classification*

7.4.1 *Metrics*

We conducted the experiment as shown in Fig 2. In the **stage 1** we took 8 models as shown in Table 1. and tried [full, LORA and MLP] based training and had 24 distinct model+finetunning variants and based on stage 1 result we took top 8 models and used to further do incremental finetunning on Chat-2k data.
We report (i) **accuracy** and (ii) **weighted-F$_1$** across the six labels. Weighted F$_1$ is needed because Chat-2k is moderately imbalanced ($12-24$ % per class).

7.4.2 *Results*

Accuracy before vs. after Stage 2 : Refer: Table 1
Result-1 Incremental gains. Refer Table 1. All finalists improve on chat data: *MobileBERT* +0.31 (0.26 $\rightarrow$ 0.57), *TinyBERT-6L* +0.31, and even the tiny *TinyBERT-4L* jumps +0.24 despite its size.

Weighted-F$_1$ before vs. after Stage 2 : Refer Table 2.
Result-2 F$_1$ recovery. Refer Table 2. Zero-shot chat F$_1$ averages only 0.26; the incremental update lifts this to 0.56 (+0.30). BERT-large again tops the chart (0.68), but MobileBERT and TinyBERT-6L closes in at 0.60 with one-third the parameters.

Checkpoint trade-offs (size / quality / speed) Refer Table 2.
Result-3 Deployment trade-off. Refer Refer Table 3. Throughput spans almost an order of magnitude—from 2.9 steps / s (BERT-large) to 23.7 (TinyBERT-4L). MobileBERT offers the best "elbow": 25 M params, 0.57 chat-F$_1$, 6.3 steps / s.

Table 1: Per-model accuracy on Tweet-Emotion and on Chat-2k before (*pre*) and after (*post*) incremental fine-tuning.

Backbone (variant)	$\mathrm{Acc}_{\mathrm{Twitter}}$	$\mathrm{Acc}_{\mathrm{Chat\text{-}pre}}$	$\mathrm{Acc}_{\mathrm{Chat\text{-}post}}$
BERT-large (full)	**0.94**	**0.38**	**0.68**
BERT-base (full)	0.94	0.34	0.65
TinyBERT-6L (full)	0.93	0.29	0.60
DistilBERT (full)	0.93	0.31	0.59
MobileBERT (full)	0.92	0.26	0.57
ALBERT-base (full)	0.93	0.31	0.55
TinyBERT-4L (full)	0.87	0.26	0.42
DistilBERT (LoRA)	0.76	0.34	0.42

Table 2: Weighted-F_1 on Tweet-Emotion and Chat-2k (*pre* vs. *post*)

Backbone (variant)	$F_{1\,\mathrm{Twitter}}$	$F_{1\,\mathrm{Chat\text{-}pre}}$	$F_{1\,\mathrm{Chat\text{-}post}}$
BERT-large (full)	**0.94**	0.31	**0.68**
BERT-base (full)	0.94	0.29	0.65
TinyBERT-6L (full)	0.93	0.27	0.60
DistilBERT (full)	0.94	0.25	0.59
MobileBERT (full)	0.93	0.23	0.57
ALBERT-base (full)	0.93	0.26	0.55
TinyBERT-4L (full)	0.87	0.20	0.42
DistilBERT (LoRA)	0.75	0.26	0.42

Sample Chat Screenshots

GitHub

Link to full source code and demo README. https://github.com/Helinskii/dost-com

BERT MODEL STATS Wandb

We trained the model and used WANDB to log the training and validation stats which can be accessed using the link provided below , along with screenshot of the report.

i) BERT MODEL- STAGE 1 WANDB link

Interactive Report Access

Full Interactive Report: STAGE-1-WANDB
Click here to access WandB report

Note: The complete report includes interactive charts, detailed metrics, and real-time training logs.

i) BERT MODEL- STAGE 2 WANDB link

Interactive Report Access

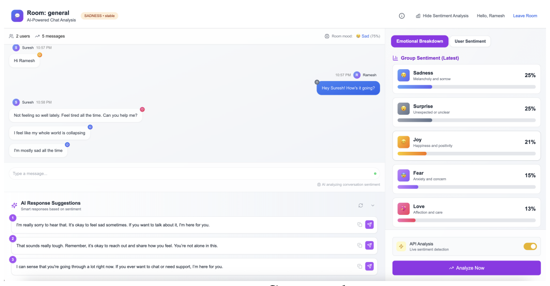
Full Interactive Report: STAGE-2-WANDB
Click here to access WandB report

Note: The complete report includes interactive charts, detailed metrics, and real-time training logs.

[1] Original repository: dair-ai/emotion_dataset.

Table 3: Model-selection trade-off: parameters, final chat accuracy F_1, and validation throughput.

Backbone (variant)	Params (M)	$\mathrm{Acc}_{\mathrm{Chat\text{-}post}}$	$F_{1\,\mathrm{Chat\text{-}post}}$	Throughput (steps/s)
TinyBERT-4L (full)	**15**	0.42	0.42	**23.7**
MobileBERT (full)	**25**	**0.57**	**0.57**	6.3
ALBERT-base (full)	<u>12</u>	0.55	0.55	7.9
TinyBERT-6L (full)	67	0.60	0.60	8.8
DistilBERT (full)	66	0.59	0.59	10.3
DistilBERT (LoRA)	66(+2*)	0.42	0.42	9.2
BERT-base (full)	110	0.65	0.65	9.7
BERT-large (full)	**335**	**0.68**	**0.68**	2.9


Room: general
AI-Powered Chat Analysis
SADNESS • stable
Hide Sentiment Analysis   Hello, Ramesh   Leave Room
2 users   5 messages   Room mood: Sad (75%)
Suresh 10:57 PM
Hi Ramesh
10:57 PM Ramesh
Hey Suresh! How's it going?
Suresh 10:58 PM
Not feeling so well lately. Feel tired all the time. Can you help me?
I feel like my whole world is collapsing
I'm mostly sad all the time
Type a message...
AI analyzing conversation sentiment
AI Response Suggestions
Smart responses based on sentiment
1 I'm really sorry to hear that. It's okay to feel sad sometimes. If you want to talk about it, I'm here for you.
2 That sounds really tough. Remember, it's okay to reach out and share how you feel. You're not alone in this.
3 I can sense that you're going through a lot right now. If you ever want to chat or need support, I'm here for you.
Emotional Breakdown   User Sentiment
Group Sentiment (Latest)

Sadness	Melancholy and sorrow	25%
Surprise	Unexpected or unclear	25%
Joy	Happiness and positivity	21%
Fear	Anxiety and concern	15%
Love	Affection and care	13%

API Analysis
Live sentiment detection
Analyze Now

Figure 5: Screenshot

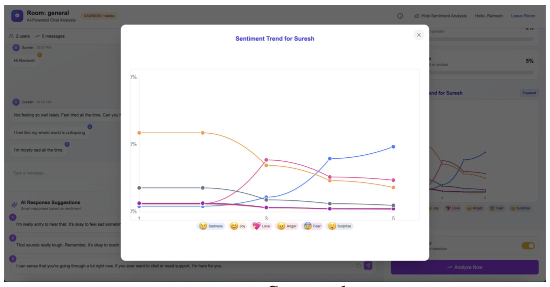
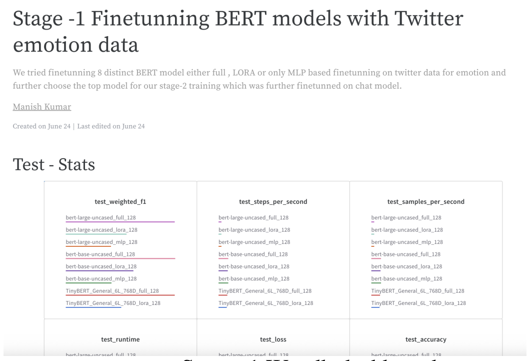


Figure 6: Screenshot


Stage -1 Finetunning BERT models with Twitter emotion data

We tried finetunning 8 distinct BERT model either full , LORA or only MLP based finetunning on twitter data for emotion and further choose the top model for our stage-2 training which was further finetunned on chat model.

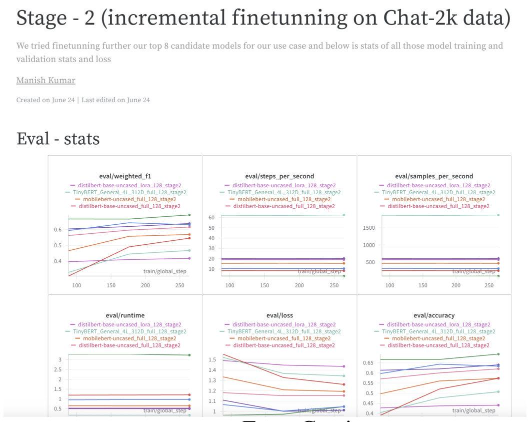
Manish Kumar

Created on June 24 | Last edited on June 24

Test - Stats

test_weighted_f1
test_steps_per_second
test_samples_per_second
bert-large-uncased_full_128
bert-large-uncased_lora_128
bert-large-uncased_mlp_128
bert-base-uncased_full_128
bert-base-uncased_lora_128
bert-base-uncased_mlp_128
TinyBERT_General_6L_768D_full_128
TinyBERT_General_6L_768D_lora_128
test_runtime
test_loss
test_accuracy

Figure 7: Stage -1 Wandb dashboard


Stage - 2 (incremental finetunning on Chat-2k data)

We tried finetunning further our top 8 candidate models for our use case and below is stats of all those model training and validation stats and loss

Manish Kumar

Created on June 24 | Last edited on June 24

Eval - stats

Figure 8: Enter Caption

Table 4: Six-way Emotion Classification Probabilities – All Stage-2 Checkpoints

Model	Text Sample	Sad	Joy	Love	Anger	Fear	Surp
TinyBERT-6L	*"I totally love this!"*	0.6	35.8	**62.6**	0.3	0.1	0.7
	"Exam makes me nervous"	3.7	0.2	1.1	0.4	**94.2**	0.5
	"App crashing – furious"	1.3	0.3	0.4	**96.3**	1.5	0.3
	"Help meant a lot"	0.5	26.9	**72.1**	0.1	0.1	0.3
ALBERT-base	*"I totally love this!"*	1.4	42.3	**48.8**	3.0	1.4	3.3
	"Exam makes me nervous"	**42.1**	2.4	2.9	6.4	41.2	4.9
	"App crashing – furious"	**29.0**	4.9	3.6	23.4	22.8	16.2
	"Help meant a lot"	0.6	4.8	**94.0**	0.2	0.1	0.3
MobileBERT	*"I totally love this!"*	0.7	4.3	**91.5**	0.9	0.5	2.1
	"Exam makes me nervous"	2.5	0.2	1.2	0.2	**95.4**	0.6
	"App crashing – furious"	1.2	1.1	2.4	**89.1**	5.2	1.0
	"Help meant a lot"	1.4	16.0	**81.9**	0.2	0.1	0.4

Note: Highest values highlighted in red. Background colors group text samples for cross-model comparison. Values are percentages.

Model	Prompt	Relevance	Naturalness	Diversity	Quality	Time (s)
gemini-2.5-flash	base	7.05	7.29	5.67	7.38	4.68
gemini-2.5-flash	no_positivity	6.72	7.03	5.35	7.28	4.64
gemini-2.5-flash	no_sentiment	7.58	7.98	6.42	7.77	4.72
gemini-2.0-flash	base	8.23	8.47	7.00	7.90	4.98
gemini-2.0-flash	no_positivity	8.17	8.50	7.00	7.93	4.98
gemini-2.0-flash	no_sentiment	8.01	8.37	6.98	7.90	6.02
gpt-4.1-mini	base	8.08	8.08	7.98	8.06	1.48
gpt-4.1-mini	no_positivity	8.08	8.06	7.98	8.02	1.56
gpt-4.1-mini	no_sentiment	8.05	8.04	8.00	8.02	1.56
gpt-4o-mini	base	8.03	8.03	8.00	8.02	1.23
gpt-4o-mini	no_positivity	8.10	8.09	8.00	8.06	1.26
gpt-4o-mini	no_sentiment	8.05	8.05	7.98	8.06	1.34

Table 5: Mean values for each model and prompt variant.

Model	Prompt	Relevance	Naturalness	Diversity	Quality	Time (s)
gemini-2.5-flash	base	2.97	3.00	2.86	2.86	4.68
gemini-2.5-flash	no_positivity	3.19	3.24	3.03	3.03	4.64
gemini-2.5-flash	no_sentiment	2.05	1.95	1.95	1.95	4.72
gemini-2.0-flash	base	0.92	0.60	1.72	1.72	4.98
gemini-2.0-flash	no_positivity	0.90	0.58	1.59	1.59	4.98
gemini-2.0-flash	no_sentiment	1.09	0.78	1.33	1.33	6.02
gpt-4.1-mini	base	0.86	0.85	0.86	0.86	1.48
gpt-4.1-mini	no_positivity	0.84	0.83	0.85	0.85	1.56
gpt-4.1-mini	no_sentiment	0.86	0.86	0.89	0.89	1.56
gpt-4o-mini	base	0.95	0.94	0.97	0.97	1.23
gpt-4o-mini	no_positivity	0.88	0.86	0.88	0.88	1.26
gpt-4o-mini	no_sentiment	0.99	0.93	0.96	0.96	1.34

Table 6: Standard deviation for each model and prompt variant.

Table 7: Paraphrased Text: Emotion Distribution

Emotion	Count
sadness	11
joy	148
love	2
anger	17
fear	22
surprise	0

Table 8: Evaluation Metrics

Metric	Evaluator	Value
Naturalness	LLM	5.00
Relevance	LLM	3.06
BERT_Score	DistilBERT	0.584

Table 9: BiLSTM Training Metrics

Metric	Score
Training Accuracy	72.64%
Precision	76.39%
Recall	72.56%
F1	71.09%
Validation Accuracy	91.94%

Table 11: Sentiment Paraphrasing Examples and Ratings

Original Text	Paraphrased Text	Naturalness	Relevance	BERT_Score	Sentiment (Input-Paraphrased
This project is a complete disaster and nothing is working.	This project is teaching me patience and patience is key.	5	4	0.674	sadness-joy
My boss never listens to my ideas and always shuts them down.	My boss listens to my ideas and encourages me to share more.	5	5	0.425	anger-joy
The meeting was boring and a waste of time.	The meeting covered important topics and topics worth noting.	5	3	0.406	sadness-joy
I hate working with this difficult team.	I'm finding peace in working with this team.	5	4	0.486	sadness-joy
The deadline is impossible to meet.	The deadline requires significant time management and planning.	5	3	0.817	sadness-joy