TOWARDS EMOTIONALLY INTELLIGENT (MOOD-SENSING) MESSENGER APPLICATIONS USING MACHINE LEARNING

Shailesh D. Wakpaijan
Student, Department of Computer Science and Application, Shri Shivaji Science College Amravati
shaileshwakpaijan@gmai.com
Sundeep S. Gawande
Assistant Professor, Department of Computer Science and Application, Shri Shivaji Science College Amravati
gawande.sundeep@gmail.com

Abstract
Emotion-aware messaging can make digital conversations more helpful, supportive, and context-sensitive. This paper presents a feasible architecture for an emotionally intelligent messenger module, which infers a user's mood from short text messages to improve conversational support. The system combines lightweight, text-based emotion recognizer with a straightforward context-aggregation layer that stabilizes short, casual chat signals into significant mood indicators. The system is proposed to be privacy-aware, modular, and compatible with integration into conventional chat platforms or as an optional add-on service. To demonstrate feasibility, the study offers a conceptual validation approach as well as some usage scenarios that indicate how the system may support higher-level features including empathetic suggestions, moderated notifications, and UI changes. The discussion covers practicality, limits, and ethical protections, as well as future research possibilities such as greater contextual modeling, personalization, and multimodal fusion.

1. Introduction

Instant messaging is rapidly becoming the most common form of daily contact. Brief chat messages, rapid replies, and emojis convey not only information but also emotion. Inferring those emotions automatically could make messaging systems more useful— for example, by proposing empathetic responses, modifying the interface to lower pressure, or notifying a user of a friend who might require attention. But brief, casual messages present special difficulties for automatic emotion detection: they are short, frequently lack overt emotional vocabulary, and employ slang, punctuation, or emojis individually.

Previous research in affective computing has shown that text classifiers, multimodal fusion, and conversation-aware models are excellent at detecting emotions in longer texts or highly annotated data. Most public datasets and high-performing models are simultaneously trained using data sources such as social media posts, long-form text, and multimodal recordings. Those resources do not share the style and constraint of one-to-one chat: brief utterances, quick turns, and heavy reliance on near context. As a result, models trained on general corpora often lose accuracy when applied directly to messenger-style text.

This paper proposes a practical, modular design for an emotionally intelligent messenger component that focuses on the chat setting. The proposed design separates concerns into four modules: message intake and light preprocessing, a single-utterance text classifier, a context aggregation layer that combines the most recent messages for a given conversation, and a small state store that smooths changes before exposing mood labels to the user interface. The aggregation stage is intended to reduce noise from single messages and to capture short-term emotional trends without requiring long histories or heavy computation.

Privacy and practical deployment guided the design choices. The architecture supports both server-side and on-device inference, permits optional anonymized logging for research, and treats mood labels as soft signals rather than definitive judgments. The user interface is expected to present emotion information in a gentle, reversible form — for example, an emoji plus short label — and to offer users the ability to opt out or correct predictions.

To illustrate feasibility without claiming an implemented system, the paper outlines a conceptual validation framework and example usage scenarios that show how the proposed modules would interact and how their outputs could support features such as empathetic suggestions and UI adaptations. The aim is to present a well-defined method and an evaluation plan rather than to report a completed production system.

The remainder of the paper reviews related work, explains the system and aggregation algorithm in detail, outlines an evaluation plan and expected analyses, and concludes with limitations and directions for future work.

2. Literature Review

Automatic emotion recognition from language has attracted sustained interest in natural language processing and affective computing. Large, well-annotated corpora have been central to progress in text-only emotion classification. The GoEmotions dataset provides a large-scale, human-annotated benchmark of 58k Reddit comments across 27 emotion categories plus neutral; it has become the standard resource for training and evaluation for fine-grained emotion classifiers. [1]

Emotion recognition in conversation is different from utterance classification because dialogue requires context modeling, turn-taking, and short-term dynamics. The MELD dataset, Multimodal EmotionLines Dataset, extends earlier dialog corpora with audio and visual channels and provides utterance-level emotion labels that are drawn from multi-party TV-dialogues, showing that contextual and multimodal cues perform better compared to text-only baselines. [2]

Multimodal resources have given impetus to research that exploits voice, facial expression, or physiological signals. Perhaps the most widely used resource has been the IEMOCAP corpus, Interactive Emotional Dyadic Motion Capture, which provides acted conversational exchanges along with synchronized audio-visual recordings; work based on IEMOCAP for speech-and video-informed emotion models reflects consistent gains when systems exploit acoustic and visual signals in addition to text. [3]

Beyond datasets, a number of application-focused studies have investigated how emotion signals can be surfaced to real users. EmotionPush is an early deployed prototype that annotates incoming instant messages with emotion indicators (presented as colored push notifications); its deployment study provides practical insight into user acceptance, notification design, and operational issues such as latency and domain mismatch. Such applied work motivates conservative, privacy-aware integration strategies for messaging clients. [4]

Recent modeling advances combine contextual encoders with external knowledge or hierarchical structures to better capture implicit affective cues in conversations. The Knowledge-Enriched Transformer (KET) augments transformer-based encoders with commonsense and affective knowledge to improve emotion detection in textual conversations, illustrating that knowledge integration and context modeling can help resolve ambiguous or implicit emotional signals at the utterance level. [5]

Taken together, these strands suggest three practical lessons for chat-focused mood sensing. First, domain mismatch matters: models trained on long-form or social-media text do not transfer directly to short, informal chat. Second, context and multimodality improve detection but add engineering and privacy costs. Third, application integration — how emotion signals are smoothed, presented, and controlled by users — is crucial for real-world usefulness. These observations motivate a design that emphasizes modularity, short-term context aggregation, and clear privacy safeguards for messenger-style emotion sensing.

3. Methodology

This section describes the practical design and operational flow of the proposed emotionally intelligent messenger module. The goal is to explain how messages are processed, how short-term context is used to infer mood, and how mood signals are smoothed and exposed to the user interface. The description focuses on modules, the aggregation algorithm, data flow, and user controls.

3.1 System overview

The system is divided into four primary components: (1) message intake and light pre-processing, (2) single-message emotion prediction, (3) short-window aggregation with hysteresis, and (4) state storage. The backend server integrates these components and communicates with both a cache server for short-window vectors and a persistent database for storing messages and committed mood states. This modular design permits replacing individual components without changing the overall flow.

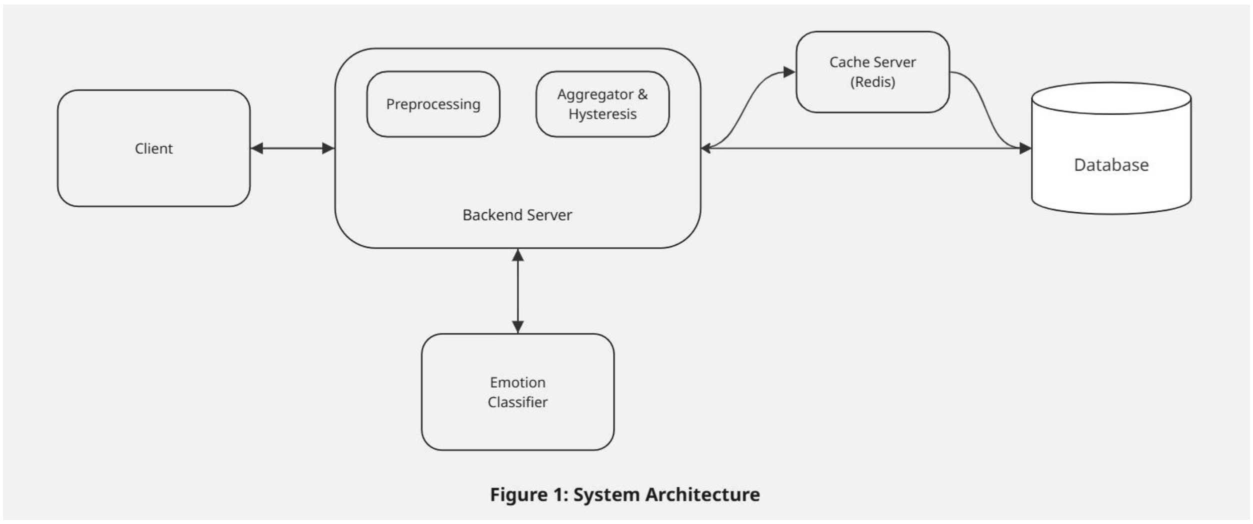

Figure 1. System architecture of the proposed mood-sensing module showing interactions between the client, backend server, cache server (Redis), and database. The backend server integrates preprocessing, emotion classification, and the aggregator–hysteresis mechanism to determine and store the final emotion.

3.1.1 Message intake and preprocessing

Incoming messages are received by the backend API and stored in the message store as usual. A minimal preprocessing step normalizes text for the classifier: trimming whitespace, normalizing repeated punctuation, and preserving emoji characters. The preprocessing step is intentionally lightweight so that informal chat features are retained.

3.1.2 Single-message emotion inference

Every preprocessed message is sent to a text-based classifier that yields a probability distribution on six emotion labels: Happy, Sad, Angry, Fear, Surprise, and Neutral. The classifier is a pluggable component; different models or encoders may be used according to deployment needs. The per-message probability vector is recorded in the emotion log and is not shown directly to users.

3.1.3 Short-window aggregation and hysteresis

To reduce noise from single short messages, an aggregation stage considers the classifier outputs for a recent window of messages within the same conversation. The aggregator computes the element-wise mean probability vector over the last window_size messages and identifies the top two scores (top1 and top2). The decision rule is:

```
if (top1 - top2) <= delta then
   label = combine_labels(emotion1, emotion2) // e.g., "Sad / Fear"
else
   label = emotion1
end if
```

A candidate label (single or mixed) is only committed to the user’s persistent state after it appears for hysteresis_count consecutive aggregation steps. This hysteresis prevents rapid flipping of the displayed mood and provides stable UI behavior.

3.1.4 Storage and data flow

1) The sender posts a message to the chat service.
2) The backend stores the incoming message in the persistent database.
3) The backend forwards a copy of the message to the recipient and marks the mood indicator as pending.
4) The backend performs light preprocessing and forwards the preprocessed text to the emotion classifier.
5) The classifier returns a probability vector, which is appended to an emotion log stored in the cache server.
6) The aggregator retrieves the cached emotion vectors for the most recent message window.
7) The aggregator computes the mean probability vector, applies the delta rule and hysteresis logic, and determines the final emotion label (single or mixed).
8) The backend writes the confirmed final emotion back to the persistent database.
9) The backend sends the final emotion annotation to the recipient’s client for display.

During operation, each incoming message is processed by the backend: it is stored persistently, analyzed for emotion, and logged temporarily in the cache for aggregation. The aggregator computes the stabilized emotional state, which is then committed to the database when hysteresis conditions are met. When hysteresis conditions are met, the backend updates user_state and sets the message final_mood field.

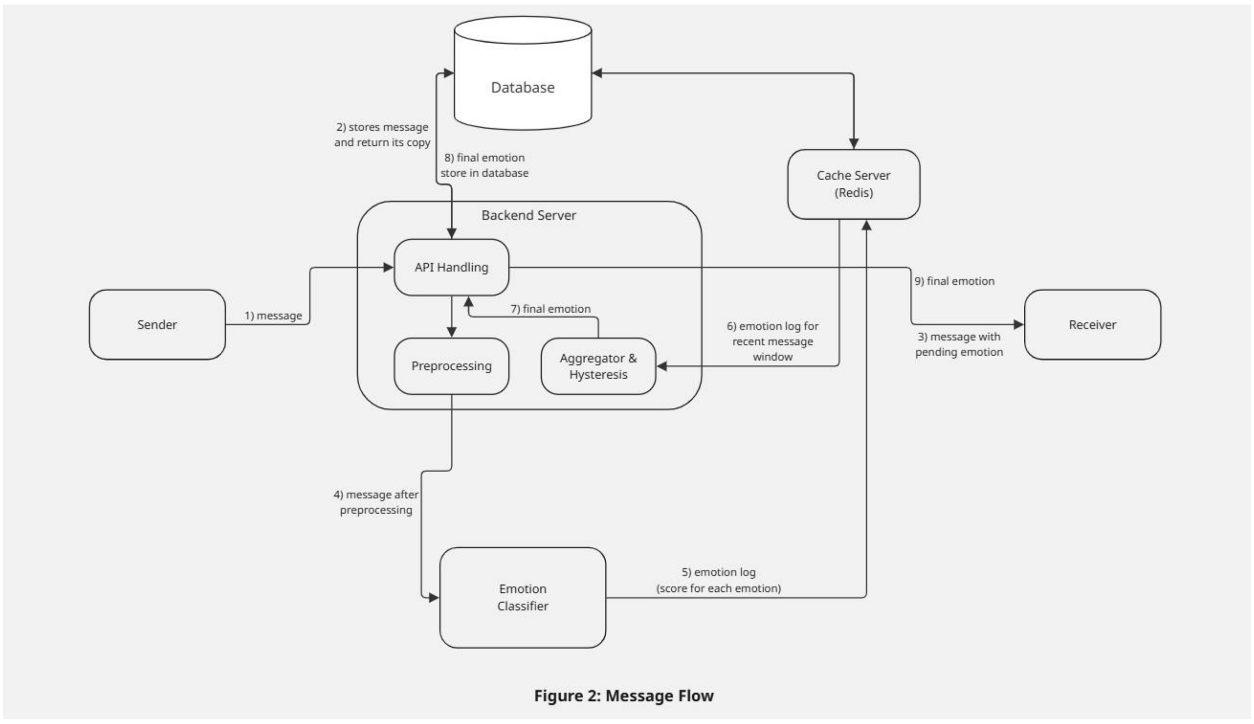
Database
2) stores message and return its copy
8) final emotion store in database
Cache Server (Redis)
Backend Server
API Handling
Sender
1) message
7) final emotion
9) final emotion
Receiver
6) emotion log for recent message window
3) message with pending emotion
Preprocessing
Aggregator & Hysteresis
4) message after preprocessing
Emotion Classifier
5) emotion log (score for each emotion)
Figure 2: Message Flow

Figure 2. Message flow from sender to receiver illustrating the sequence of operations involved in emotion processing. Each message undergoes preprocessing and emotion classification, followed by aggregation and hysteresis for final emotion determination before being stored in the database and displayed to the receiver.

3.2 Database design

The persistent storage is intentionally simple and keeps only the canonical records needed by the messenger and the mood module. The database contains a small set of collections/tables that hold users, conversations, messages, emotion logs, and the current user mood. Each collection has a single, clear purpose:

Users: stores account and privacy settings for each person (unique id, display name, opt-in flags).

Conversations: represents a chat thread and its participants (one-to-one or group).

Messages: stores each message content, sender, timestamp, current status (pending/final), and the finalized mood label when available.

Emotion_logs: keeps per-message classifier outputs (timestamped probability vectors and model id) when persistence is needed for auditing or later analysis.

User_state: holds the current committed mood for a user and a last-updated timestamp for quick UI reads.

Contacts: stores friend links and notification consent for features like close-contact alerts.

These collections are linked in a straightforward way: messages belong to conversations and reference their sender (user); emotion logs attach to specific messages; and the aggregator writes the confirmed mood back to the corresponding message record and updates the user_state table. Persistent records remain the source of truth; transient working data (for fast aggregation in an implementation) is not shown in the diagram.

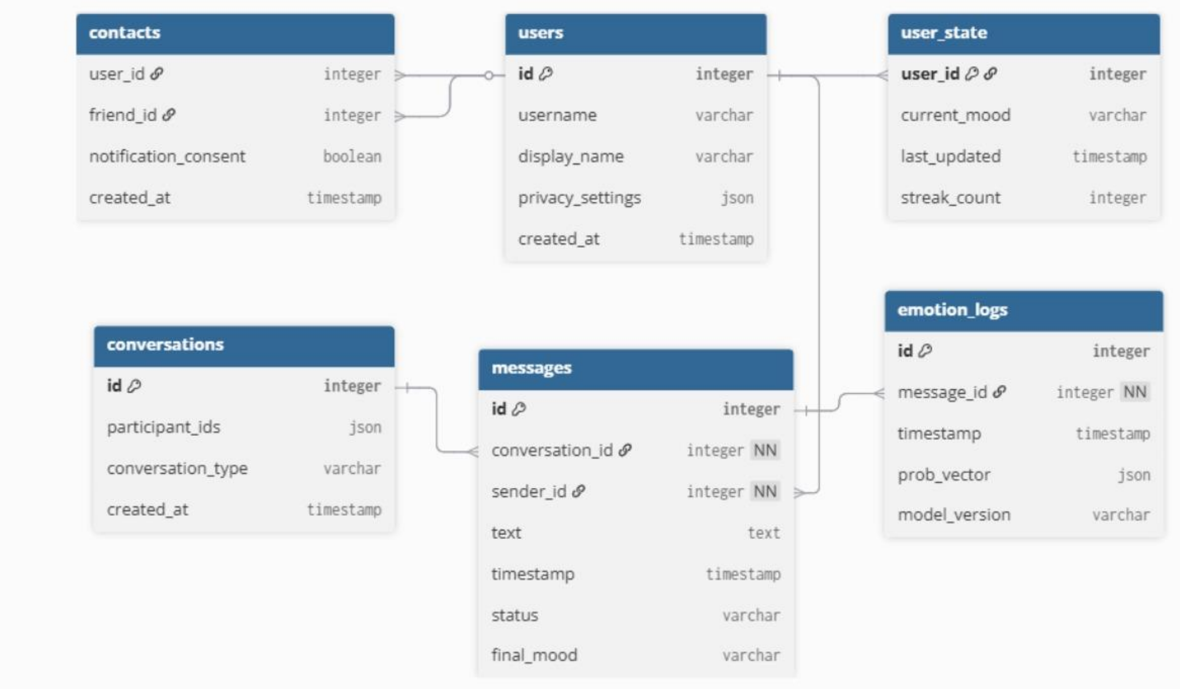

Figure 3: Database Design

Figure 3. Persistent data model for the mood-sensing module, showing primary collections and their main relations (messages → conversations → users; emotion_logs attached to messages; user_state for quick mood access).

3.3 UI integration and user controls

Mood indicators are presented in a simple manner: an emoji and a brief label near the chat header. For mixed emotional states, combined labels can be displayed (e.g., 😊 Happy / 😮 Surprise). Users should have the option to disable mood indicators if they prefer not to display emotional information during conversations.

3.4 Privacy, deployment, and safeguards

The system supports two deployment modes: server-side inference (with clear retention policies and opt-in) and on-device inference (no server transfer of raw text). All persistent logs should be minimized and retained only under clear policy. Any mechanism that sends mood information to third parties (for example, notifying a close contact) must be explicit and opt-in.

4. Results and Discussion

This section describes the planned presentation of empirical findings and guidance for interpreting likely outcomes. As the current work emphasizes a proposed design and evaluation plan, the following layout clarifies which metrics, visualizations, and analyses will be used once experiments are performed, making results comparable and reproducible.

4.1 Presentation of results

Results will be organized into three parts:

4.1.1 Quantitative classifier performance.

Report classification metrics for each emotion class and overall scores: precision, recall, F1-score (macro- and micro-averaged). Overall accuracy will be reported but interpreted cautiously with imbalanced classes. A classification-report table will appear here.

4.1.2 Confusion analysis.

A confusion matrix will show frequent misclassifications between emotion pairs, normalized by row for comparison. Typical patterns, e.g., Sad vs Neutral or Fear vs Surprise, will be highlighted.

4.1.3 Aggregation effect and latency.

Compare single-message predictions with short-window aggregation. Evaluate whether aggregation increases per-class F1, reduces spurious label flips, and frequency of mixed labels. Record latency and memory use to assess deployability.

4.2 Interpreting common patterns

- **High precision, low recall:** conservative predictions, fewer false positives but may miss true occurrences.
- **Low precision, high recall:** more false positives, may reduce user trust.

- **Confusion between related emotions** (Surprise vs Fear, Sad vs Neutral) reflects limits of short text.
- **Aggregation improvements** (higher macro-F1, fewer rapid label changes) support the proposed stabilizing effect.

4.3 Practical considerations and limitations

Small datasets or domain mismatch: results are preliminary; report confidence intervals when possible.

User experience trade-offs: balance recall and precision according to privacy-first or feature-rich deployments.

4.4 Summary of expected contribution

Planned evaluation will show whether short-window aggregation improves mood-sensing in chat, highlighting strengths and limits of text-only, short-message emotion inference, and assessing aggregation/hysteresis effects on stability and user-facing behavior.

5. Conclusion

This paper presented a realistic, modular approach to adding mood sensing to messenger apps. The suggested approach mixes a light-weight text-based emotion recognizer with a short-window aggregation phase and a hysteresis mechanism to mitigate temporary errors and stabilize user-facing mood labels. The design promotes privacy and adaptability: the emotion classifier is a plugin component and the platform is compatible with both server-side and on-device deployment modes. The design prioritizes careful exposure of mood cues through subtle UI and explicit user controls. Generally, the proposal seeks to facilitate emotion-sensitive features—like empathetic recommendations and adaptive interfaces—without compromising user control and data minimization.

6. Future Scope

The proposed design provides a foundation for several promising extensions and further studies:

1. **Emoji-aware processing:** Current text preprocessing preserves emoji characters but does not fully exploit their semantic value. Future work can incorporate emoji-aware tokenization and embedding schemes so that emoji content directly informs emotion inference.
2. **Group conversation support:** The present proposal focuses on one-to-one chats. Extending the approach to group conversations requires handling multiple participants, turn-taking, and aggregated group-level mood signals. Group-aware aggregation policies and privacy-preserving summaries are natural directions for expansion.
3. **Longer-term mood monitoring and notifications:** Beyond per-message mood, the system could monitor persistent changes in a user's emotional state. With ethical and rigorous consent protection, users are able to opt to notify close contacts if there are recurring patterns of distress, worry, or sadness. Any notification functionality must be opt-in, clear, and subject to policies so that users are not misused.
4. **Multimodal fusion;** Merging other signals (e.g., short voice clips, pictures, or interaction metadata) can improve accuracy where text on its own is vague. Accuracy gains must be weighed by research against privacy, latency, and implementation cost.
5. **Personalization and calibration:** Make the system learn how each person writes so it reads emotions more accurately for that user. For example, it will learn if someone often says "I'm fine" but actually means the opposite. Personalization can run on the device or use privacy-friendly methods so raw messages don't have to leave the phone.
6. **On-device optimization and resource-aware deployment:** Make the emotion model small and fast so it can run on phones without draining battery or memory. This reduces delay and keeps user data private because processing happens on the device instead of on a server.
7. **Multilingual and sarcasm detection:** Extend the system to handle more languages and spot sarcasm or ironic messages. This improves accuracy for users who write in other languages or who use sarcasm, which often breaks simple text-only detectors.

References

[1] D. Demszky, D. Movshovitz-Attias, J. Ko, A. Cowen, G. Nemade, and S. Ravi, "GoEmotions: A Dataset of Fine-Grained Emotions," in *Proc. 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020, pp. 4040–4054.

[2] S. Poria, D. Hazarika, N. Majumder, G. Naik, E. Cambria, and R. Mihalcea, "MELD: A Multimodal Multi-Party Dataset for Emotion Recognition in Conversations," in *Proc. Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[3] C. Busso, M. Bulut, C.-C. Lee, A. Kazemzadeh, E. Mower, S. Kim, J. N. Chang, S. Lee, and S. S. Narayanan, "IEMOCAP: Interactive Emotional Dyadic Motion Capture Database," *Language*

Resources and Evaluation, vol. 42, no. 4, pp. 335–359, 2008.

[4] S.-M. Wang, C.-H. Li, Y.-C. Lo, T.-H. K. Huang, and L.-W. Ku, "Sensing Emotions in Text Messages: An Application and Deployment Study of EmotionPush," *arXiv preprint* arXiv:1610.04758, Oct. 2016; demo/ACL workshop materials also available.

[5] P. Zhong, D. Wang, and C. Miao, "Knowledge-Enriched Transformer for Emotion Detection in Textual Conversations," in *Proc. EMNLP-IJCNLP (Workshops)*, 2019, pp. 165–176.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. North American Chapter of the Association for Computational Linguistics — Human Language Technologies (NAACL-HLT)*, 2019, pp. 4171–4186.

[8] B. Felbo, A. Mislove, A. Søgaard, I. Rahwan, and S. Lehmann, "Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm," in *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017, pp. 1625–1635.

[9] C. J. Hutto and E. Gilbert, "VADER: A parsimonious rule-based model for sentiment analysis of social media text," in *Proc. International AAAI Conference on Web and Social Media (ICWSM)*, 2014.

[10] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," Stanford University, CS224N Project Report, 2009.

[11] A. Zadeh, M. Liang, S. Poria, E. Cambria and L.-P. Morency, "CMU-MOSEI: Multimodal sentiment, emotion and opinion intensity dataset," in *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[12] D. Ghosal, N. Majumder, S. Poria, N. Chhaya, and A. Gelbukh, "DialogueGCN: A graph convolutional neural network for emotion recognition in conversation," in *Proc. Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 154–164.