Multilabel Sentiment and Emotion Detection for Bangla YouTube Comments

---

1. Introduction

This document specifies the system requirements, design, and evaluation of an NLP system that performs:

- Sentiment detection on Bangla YouTube comments under:
    - a 3-label scheme (Negative, Neutral, Positive), and
    - a 5-label scheme (Highly Negative, Negative, Neutral, Positive, Highly Positive).
- Emotion detection on the same domain of comments:
    - Emotions: Anger, Joy, Disgust, Fear, Surprise, Sad, None.

The system is built from two main modeling components:

- A sentiment detection module implemented in the sentiment notebook.
- An emotion detection module implemented in the emotion notebook.

Both modules operate on noisy, code-mixed comments in Bangla (BN), English (EN), and Romanized Bangla (RN). The SRD describes:

- Dataset characteristics
- Preprocessing and feature extraction
- Model architectures and training setup
- Quantitative performance and model selection
- Functional and non-functional requirements

No deployment or UI details are included here.

2. Dataset Description

2.1 Source and Context

The system uses the dataset introduced in:

> N. Irtiza Tripto & M. Eunus Ali, “Detecting Multilabel Sentiment and Emotions from Bangla YouTube Comments,” 2018 International Conference on Bangla Speech and Language Processing (ICBSLP), Sylhet, 2018, pp. 1–6. doi: 10.1109/ICBSLP.2018.8554875

Key properties:

- Comments are collected from a variety of Bangla YouTube videos.
- Videos are manually selected based on popularity signals:
    - Number of views
    - Number of likes/dislikes
    - Time range: 2013 to early 2018.
- Up to 50 top-level comments are collected per video.
- Replies are excluded to avoid redundancy and because they often contain less stand-alone sentiment information.
- Comments may include:
    - Abusive and vulgar language
    - Slang, sarcasm, and personal attacks
- To ensure ethical data handling, all annotators were adults.

2.2 Language Detection

Language detection is performed using Google Translator:

- If detected as Bangla → tagged as BN
- If detected as English → tagged as EN
- If not clearly identified → treated as Romanized Bangla (RN)

The RN category therefore implicitly includes some noisy or ambiguous text, reflecting typical social media writing styles.

2.3 Files and Schema

2.3.1 Sentiment.csv

This file is used for multilabel sentiment analysis.

- Separator: `;`
- Encoding: `utf-8-sig`

Columns:

- `id` – unique identifier for a comment
- `text` – raw YouTube comment
- `label` – indicates the sentiment labeling scheme:
    - 1 → comment annotated under 3-label scheme
    - 2 → comment annotated under 5-label scheme
- `score` – numeric sentiment polarity
- `lan` – language of the comment: EN, BN, or RN
- `domain` – video category/domain

3-label scheme (`label = 1`):

- 1 → Positive
- 0 → Neutral
- -1 → Negative

5-label scheme (`label = 2`):

- 2 → Highly Positive
- 1 → Positive
- 0 → Neutral
- -1 → Negative
- -2 → Highly Negative

2.3.2 Emotion.csv

This file is used for emotion detection.

- Separator: ;
- Encoding: `utf-8-sig`

Columns:

- `id` – unique identifier
- `text` – raw comment text
- `emotion` – discrete emotion label
- `lan` – EN, BN, or RN
- `domain` – video category

Emotion labels:

- Anger, Joy, Disgust, Fear, Surprise, Sad, None (no emotion detected).

3. System Overview

3.1 Main Components

The overall system is modular, consisting of:

- **Sentiment Detection Module**
    - 3-label sentiment classification
    - 5-label sentiment classification
- **Emotion Detection Module**

Each module follows a similar processing pipeline:

- Data loading and cleaning
- Language-aware text preprocessing
- Feature extraction:
    - TF–IDF for classical ML
    - Tokenization and padding for deep learning
- Model training, cross-validation, and test evaluation
- Final model comparison and selection

3.2 Modeling Approaches

The system compares:

- **Classical machine learning models:**

    - Logistic Regression
    - Multinomial Naive Bayes
    - Linear SVM
    - Passive Aggressive Classifier
- **Deep learning models:**
    - Simple RNN
    - LSTM / multi-layer LSTM
    - BiLSTM

This allows evaluation of trade-offs between simpler linear models and recurrent neural architectures on short, noisy text.

4. Text Preprocessing

Preprocessing is shared across sentiment and emotion tasks but uses language-specific logic.

4.1 Normalization

For each comment:

- Convert text to lowercase.
- Remove URLs (patterns beginning with `http`, `https`, or `www`).
- Remove user mentions and hashtags.
- Retain only:
    - Bangla characters (`\u0980–\u09FF`)
    - English letters (`a–z`, `A–Z`)
    - Whitespace

- Normalize whitespace (collapse multiple spaces, trim edges).

4.2 Tokenization

- Comments are tokenized into word tokens using NLTK's tokenizer.

4.3 Stopword Handling

Three stopword sets are used:

- **English stopwords** from NLTK.
- **Bangla stopwords** from NLTK (Bengali stopword list).
- **Romanized Bangla stopwords**, manually curated for common conversational tokens (e.g., "ami", "tumi", "onek", "vai", "bro", etc.).

Tokens are filtered according to the `lan` field:

- `BN` → filtered using Bangla stopwords
- `EN` → filtered using English stopwords
- `RN` → filtered using the RN stopword list

4.4 Stemming (English Only)

- English tokens are stemmed with the Porter stemmer to reduce inflectional variants.
- Bangla and RN tokens are not stemmed due to script and orthographic variability.

4.5 Length Filtering

- English tokens of length ≤ 2 are removed.
- Bangla and RN tokens of length ≤ 1 are removed.

4.6 Clean Text Field

- The filtered tokens are joined with spaces to form a `clean_text` feature.
- Rows where `clean_text` is empty after preprocessing are dropped from further analysis.

5. Sentiment Detection Module

All sentiment logic resides in the sentiment notebook / script and is divided into:

- 3-label sentiment classification
- 5-label sentiment classification

5.1 Common Data Preparation

- Load `Sentiment.csv` with correct separator and encoding.
- Drop exact duplicate rows.
- Drop non-essential columns: `id`, `domain`.
- Apply the preprocessing pipeline to obtain `clean_text`.
- Ensure `label` and `score` are stored as integers.
- Split into:
    - `df_sentiment_3label` where `label = 1`
    - `df_sentiment_5label` where `label = 2`

All subsequent steps operate on these filtered subsets.

5.2 3-Label Sentiment Classification

5.2.1 Task Definition

Target:

- `score ∈ {-1, 0, 1}`

Mapping:

- `-1` → Negative
- `0` → Neutral
- `1` → Positive

Input:

- `clean_text` from `df_sentiment_3label`.

A stratified train–test split (80/20) is used to preserve class proportions.

5.2.2 Feature Extraction

- **Classical ML models** use TF–IDF features with a capped vocabulary size of **max_features = 8000**.
- **Deep learning models** use:
    - A token index with a limited vocabulary (matching the deep learning configuration)
    - Padded sequences of fixed length (**150 tokens**)
    - Labels mapped to discrete classes and converted to one-hot vectors.

5.2.3 Models Evaluated

For 3-label sentiment:

- Logistic Regression
- Multinomial Naive Bayes
- Linear SVM
- Passive Aggressive Classifier
- Simple RNN
- LSTM
- BiLSTM

Each model is evaluated with:

- 5-fold cross-validation (training and validation performance)
- Final test set evaluation

5.2.4 Final Test Accuracy Comparison (3-Label)

Table 1. Final Model Comparison – 3-Label Sentiment (Test Accuracy)

Model	Accuracy
BiLSTM	**0.6340**
Logistic Regression	**0.6274**
Naive Bayes	**0.6251**
SVM	**0.6207**
Passive Aggressive	**0.5996**
SimpleRNN	**0.3726**
LSTM	**0.3704**

Key points:

- BiLSTM is the best-performing model with an accuracy of 0.6340.
- Logistic Regression and Naive Bayes are very close behind, showing that linear models are competitive on this task.
- Simple RNN and LSTM perform significantly worse, indicating that naïve recurrent architectures under the chosen hyperparameters underfit the data compared to BiLSTM and the linear baselines.

5.2.5 Cross-Validation Behaviour (3-Label)

From the 5-fold results:

- **Logistic Regression, Naive Bayes, SVM, Passive Aggressive:**
    - Training accuracies are high (≈ 0.88–0.95).
    - Validation accuracies are clearly lower (≈ 0.55–0.59).
    - Test accuracies around 0.60–0.63.
    - → These models show **overfitting**: they fit the training data very well but generalize less strongly.
- **Simple RNN and LSTM:**
    - Both training and validation accuracies are low (~0.35), with similarly low test performance.
    - → These models **underfit**, failing to capture sufficient information from the data.
- **BiLSTM:**
    - Training accuracy is moderate, validation accuracy slightly lower, and test accuracy reaches 0.6340.
    - → BiLSTM offers the best compromise between capacity and generalization.

5.3 5-Label Sentiment Classification

5.3.1 Task Definition

Target:

- `score` ∈ {-2, -1, 0, 1, 2} mapped conceptually to:
    - -2 → Highly Negative
    - -1 → Negative
    - 0 → Neutral
    - 1 → Positive
    - 2 → Highly Positive

Input:

- `clean_text` from `df_sentiment_5label`.

Again, a stratified 80/20 train–test split is used.

5.3.2 Feature Extraction

- **Classical ML:** TF–IDF features with **max_features = 8000** to better capture subtle differences between fine-grained sentiment labels.
- **Deep learning:**
    - Tokenization with a vocabulary appropriate for the 5-label dataset.
    - Padded sequences to a fixed maximum length (consistent with the sentiment deep learning configuration).
    - Labels encoded as one-hot vectors across 5 classes.

5.3.3 Handling Class Imbalance

The 5-label dataset is more imbalanced, especially for the extreme classes (Highly Negative and Highly Positive).

To address this:

- **Targeted class weights** are used during training in:
    - Selected **classical models**: Logistic Regression, Linear SVM, and Passive Aggressive Classifier.
    - All **deep learning models**: Simple RNN, LSTM, and BiLSTM.
- The implementation **up-weights the extreme classes** (Highly Negative, Highly Positive) relative to the middle classes (Negative, Neutral, Positive), encouraging the models to pay more attention to underrepresented sentiment categories while leaving intermediate classes at weight 1.0.

Details of the exact weight values and implementation are kept in the notebook; they are not expanded in this SRD.

5.3.4 Final Test Accuracy Comparison (5-Label)

Table 2. Final Model Comparison – 5-Label Sentiment (Test Accuracy)

Model	Accuracy
BiLSTM	**0.5000**
Logistic Regression	**0.4644**
Naive Bayes	**0.4548**
SVM	**0.4521**
Passive Aggressive	**0.4274**
SimpleRNN	**0.3123**
LSTM	**0.1068**

Key points:

- BiLSTM again outperforms all other models, achieving 0.5000 accuracy on a more difficult, fine-grained sentiment task.
- Classical models (Naive Bayes, SVM, Logistic Regression) remain competitive with scores around 0.45–0.46.
- LSTM performs particularly poorly (0.1068), and Simple RNN is also weak (0.3123), indicating severe underfitting or instability in this configuration.

5.3.5 Cross-Validation Behaviour (5-Label)

- **Logistic Regression, Naive Bayes, SVM, Passive Aggressive:**
    - High training accuracies (up to 0.93–0.95).
    - Lower validation accuracies (~0.43–0.46).
    - Test accuracies in the 0.43–0.46 range.
    - → Clear **overfitting**: models fit training sets extremely well but do not generalize proportionally.
- **Simple RNN and LSTM:**
    - Both training and validation accuracies remain low (~0.18–0.27).
    - Test performance is also low.
    - → These models **underfit**, failing to leverage the additional capacity into effective decision boundaries.
- **BiLSTM:**
    - Training and validation accuracies are modest (~0.26 and ~0.24), but test accuracy rises to 0.5000.
    - → BiLSTM, with class weighting and proper sequence modeling, emerges as the best-performing model for the 5-label task.

6. Emotion Detection Module

The emotion module mirrors the sentiment pipeline but uses `Emotion.csv` and discrete emotion labels.

6.1 Task Definition

Target:

- `emotion ∈ {Anger, Joy, Disgust, Fear, Surprise, Sad, None}`

Input:

- `clean_text` from the emotion dataset after preprocessing.

Data is split into stratified training and test sets (80/20).

6.2 Feature Extraction

- **Classical ML models:**
    - TF–IDF features with **max_features = 7000**.
- **Deep learning models:**
    - Token sequences produced by a tokenizer.
    - Padded to a fixed length of **120 tokens**.
    - Labels one-hot encoded across all emotion classes.

6.3 Models Evaluated

For emotion detection:

- Logistic Regression
- Naive Bayes
- Linear SVM

- Passive Aggressive
- Simple RNN
- Multi-layer LSTM
- BiLSTM

The same evaluation strategy is used: 5-fold cross-validation plus final test evaluation.

6.4 Final Test Accuracy Comparison (Emotion)

Table 3. Final Model Comparison – Emotion Classification (Test Accuracy)

Model	Accuracy
BiLSTM	**0.475836**
Logistic Regression	**0.460967**
Naive Bayes	**0.449814**
SVM	**0.438662**
Passive Aggressive	**0.401487**
SimpleRNN	**0.280669**
LSTM	**0.280669**

Key points:

- BiLSTM achieves the highest test accuracy (~0.476).
- Logistic Regression is close behind (~0.461), showing again that linear models are strong baselines.
- Naive Bayes and SVM also perform reasonably well around 0.44–0.45.

- Simple RNN and LSTM lag significantly with accuracy ~0.28.

6.5 Cross-Validation Behaviour (Emotion)

- **Logistic Regression, Naive Bayes, SVM, Passive Aggressive:**
    - Training accuracies are high (0.74–0.91).
    - Validation accuracies hover around 0.41–0.45.
    - Test accuracies roughly 0.40–0.46.
    - → These models **overfit**: excellent training performance but more modest generalization.
- **Simple RNN and LSTM:**
    - Training and validation accuracies are low (~0.26–0.27), and test accuracy remains at ~0.28.
    - → These models **underfit**, indicating insufficient capacity or suboptimal architecture for this noisy multi-emotion problem.
- **BiLSTM:**
    - Training accuracy ~0.33, validation ~0.32, test ~0.48.
    - → BiLSTM manages to generalize better than other recurrent models and provides the best overall emotion classification performance.

7. Functional Requirements

7.1 Sentiment Module

The system shall:

- Load the sentiment dataset from `Sentiment.csv`.
- Clean and normalize raw text into `clean_text`.

- Apply language-aware stopword removal and English stemming.
- Build and evaluate separate models for:
    - 3-label sentiment classification
    - 5-label sentiment classification
- Train the following model families on each task:
    - Logistic Regression
    - Multinomial Naive Bayes
    - Linear SVM
    - Passive Aggressive Classifier
    - Simple RNN
    - LSTM
    - BiLSTM
- For the 5-label task, **use targeted class weighting in selected classical models (Logistic Regression, Linear SVM, Passive Aggressive) and all deep learning models (Simple RNN, LSTM, BiLSTM)** to mitigate label imbalance, especially for the Highly Negative and Highly Positive classes.
- Compute and report:
    - Accuracy, precision, recall, and F1-score on test data
    - 5-fold cross-validation performance summaries
    - Final comparison tables (Tables 1 and 2).

7.2 Emotion Module

The system shall:

- Load the emotion dataset from `Emotion.csv`.
- Apply the same preprocessing pipeline.
- Train:
    - Logistic Regression
    - Naive Bayes
    - Linear SVM
    - Passive Aggressive
    - Simple RNN
    - LSTM / multi-layer LSTM
    - BiLSTM
- Report:
    - Accuracy, precision, recall, and F1-score on test data
    - 5-fold cross-validation summaries
    - Final comparison table (Table 3).

8. Non-Functional Requirements

8.1 Performance

- Classical models with TF–IDF features should allow low-latency inference on CPU for individual comments.

- Deep models should be computationally feasible on standard CPU or GPU for batch inference.

8.2 Robustness

The preprocessing pipeline must tolerate:

- Misspellings, emojis (removed), and noisy punctuation.
- Code-mixed comments switching between Bangla, English, and RN.

The system should avoid crashing on malformed or partially missing rows (handled via dropping problematic lines).

8.3 Reproducibility

Fixed random seeds are set for:

- NumPy
- TensorFlow
- Python's `random` module

This supports reproducible experiments in the same environment.

9. Limitations and Future Work

- Romanized Bangla is highly variable and noisy; specialized normalization layers could improve performance.
- Even with class weights, rare sentiment classes (Highly Positive/Negative) and some less frequent emotions remain challenging.
- Deep learning models could be improved through:
    - Pre-trained multilingual embeddings

    - Transformer-based architectures (e.g., multilingual BERT, XLM-R)
    - More extensive hyperparameter tuning and regularization strategies.
- Additional evaluation metrics such as macro-averaged F1 per class can be used to more clearly analyze performance on underrepresented labels.

10. Conclusion

This SRD has documented the system for multilabel sentiment and emotion detection on Bangla YouTube comments, focusing strictly on the modeling code implemented in the sentiment and emotion notebooks.

Across all three tasks, BiLSTM consistently provides the best test accuracy:

- 3-label sentiment: **0.6340**
- 5-label sentiment: **0.5000**
- Emotion detection: **0.4758 (approx.)**

Classical models (especially Logistic Regression, Naive Bayes, and SVM) remain strong baselines with competitive performance and simpler architectures.

The system successfully handles noisy, code-mixed, real-world comments and provides a clear comparative evaluation framework for future improvements.