Project: Sentiment Analysis of Bangla Social Media Comments: BERT vs Random Forest

Introduction: This project focuses on two approach for classifying Bangla social media content and comparing these two methods.

Why Two Approaches:

- Transformer(Bangla-BERT): Chosen for it's ability to understand of nuanced and difficult text using contextual embeddings. Ideal for production-grade applications.
- Random Forest: Selected for it's simplicity, speed and for faster prototype.

Training Pipeline Comparison:

Random Forest Model Pipeline:

1. Data Loading & Splitting
2. Intensive Text Preprocessing
3. Class Balancing with RandomOverSampler
4. Text Vectorization using TF-IDF
5. Hyperparameter Tuning with Optuna
6. Model Training
7. Evaluation

Transformer Based (Bangla-BERT Classification) Model Pipeline:

1. Data Loading & Basic Preprocessing
2. Tokenization & Dataset Wrapping
3. DataLoader Creation with Imbalance Handling
4. Model Initialization
5. Training Utilities
6. Hyperparameter Tuning with Optuna
7. Threshold Optimization
8. Final Training & Evaluation
9. ONNX Export for Deployment

How I Handle Bangla Text Classification Differently: BERT vs. Random Forest

1. Preparing the Data:

For Random Forest (the classic approach):

- I first split our data into train/test sets, then manually balance the training data by oversampling rare classes (like "threat" comments). This happens before I convert text to numbers.

- The text cleaning is intensive - I remove noise, filter stopwords, and normalize everything to make word frequencies meaningful.

For BERT (the modern approach):

- I keep the original imbalanced data but teach the model to pay more attention to rare examples during training. Every batch it sees gets automatically rebalanced.

- I do minimal text cleaning because BERT understands messy social media text well.

2. Converting Text to Numbers

Random Forest's Way:

- I use TF-IDF (a fancy word counting method) that looks at:
    - Individual words
    - Pairs of consecutive words
- Creates a sparse 10,000-column "bag of words".

BERT's Way:

- Uses its built-in understanding of Bangla to break text into meaningful pieces (even parts of words)
- Creates rich 768-number vectors that capture word meanings and context.

3. Dealing with Unbalanced Classes

Random Forest Solution:

1. First, I artificially create more copies of rare comments.
2. Then I tell the model to penalize mistakes on rare classes more heavily

BERT Solution:

1. Every time the model looks at a small batch of data, I make sure rare examples appear more often.
2. I also adjust the grading system to care more about mistakes on rare cases

Result comparison:

Category	#Examples	BERT Score	RF Score	Difference
Not Bully	15,340	85%	64%	+21%
Troll	10,462	77%	49%	+28%
Sexual	8,928	84%	43%	+41%
Religious	7,577	92%	48%	+44%
Threat	1,694	79%	20%	+59%

Overall Accuracy:

- BERT: 84%
- Random Forest: 51%

Limitations:

- Rare labels like “threat” still have few examples
- Fine-tuning BERT requires GPU time and memory

Future Improvements:

- Gather more “threat” and other low-count samples
- Distill BERT into a smaller model for faster, cheaper inference
- Add simple explainers (e.g. attention heatmaps) so users see why a comment is flagged
- Build a feedback loop to retrain on mistakes

How to run the API: Start FastAPI server,

uvicorn app.main:app --reload --host 0.0.0.0 --port 8000

Example requests and responses: