TeamHateMate at BLP Task1: Divide and Conquer: A Two-Stage Cascaded Framework with K-Fold Ensembling for Multi-Label Bangla Hate Speech Classification

Mehedi Hasan, Mahbub Islam Mahim
Jahangirnagar University, Savar, Dhaka
These authors contributed equally to this work.

Abstract

Detecting hate speech on social media is essential for safeguarding online communities, yet it remains challenging for low-resource languages like Bangla due to class imbalance and subjective annotations. We introduce a two-stage cascaded framework with k-fold ensembling to address the BLP Workshop 2025 Shared Task's three subtasks: 1A (hate type classification), 1B (target identification), and 1C (joint classification of type, target, and severity). Our solution balances precision and recall, achieving micro-$F1$ scores of 0.7331 on 1A, 0.7356 on 1B, and 0.7392 on 1C, ranking 4th on 1A and **1st** on both 1B and 1C. It performs strongly on major classes, although underrepresented labels such as *sexism* and *mild severity* remain challenging. Our method makes the optimal use of limited data through k-fold ensembling and delivers overall balanced performance across majority and minority classes by mitigating class imbalance via cascaded layers.

1 Introduction

Social media's growth has accelerated the spread of hate speech, presenting major threats to online safety and public welfare (Vogels, 2021). Despite having a large speaker base, Bangla is still not well-studied, even though automatic detection systems have advanced in high-resource languages (Fortuna and Nunes, 2018; Das et al., 2021). Subjectivity in annotation, overlapping linguistic cues across categories, and a stark class imbalance make the task especially challenging (Vidgen and Derczynski, 2020).

To overcome these obstacles, we introduce a two-phase cascaded framework with k-fold ensembling in this paper (Tang and Dalzell, 2019). After distinguishing between hate and non-hate content, our method gradually improves predictions across multiple categories. Our approach reduces overfitting and stabilizes performance in imbalanced conditions by using ensembling and cross-validation (Mozafari et al., 2020). Our framework shows strong results across all subtasks of the BLP shared task 1 (Hasan et al., 2025b), demonstrating that our divide-and-conquer tactics can successfully maintain a balance between recall and precision for Bangla hate speech detection.

2 Related Work

Recent studies approach hate speech detection in low-resource languages using transformer models, hybrid architectures, and ensemble techniques. (Saha, 2023) combined IndicBERT with a Naive Bayes classifier and synthetic upsampling, achieving macro-F1 scores of 0.73 (Assamese), 0.68 (Bengali), and 0.84 (Bodo). Their approach of data augmentation using back translation tended to smooth out language-specific features, reducing multi-class separability in our study. (Ripoll et al., 2022) used multilingual transformer models trained across k-fold splits and ensembled via soft voting. Their system ranked first place in contextual hate speech, but lacks addressing class imbalance issues. (Das et al., 2023) proposed a Hierarchical-BERT for Bangla violence detection, where the first layer identifies major classes and the second layer refines them. However, misclassifications in the first layer often propagate errors, whereas our cascading design allows the second layer to revise initial mispredictions. (Veeramani et al., 2023) ensembled three BERT-based models with distinct optimization strategies such as extra classification heads and masked language model pretraining. Their system achieved F1 scores of 0.7347 for violence and 0.7173 for sentiment. (Ababu et al., 2025) applied BiLSTM with FastText embeddings for bilingual hate speech detection in Amharic and Afaan Oromo, reaching 78.05% accuracy. While FastText helped them deal with words that were out of their vocabulary, its static nature hindered contextual understanding—a crucial component of multi-class

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 453–460

hate classification.

3 Task Description

3.1 Task Overview

This shared task focuses on detecting hate speech in Bangla social media content across three subtasks: 1A - classify the type of hate (e.g., Abusive, Sexism, Religious, Political, Profane, or None); 1B - identify the target (Individuals, Organizations, Communities, or Society); 1C - jointly predict hate type, severity (little to none, mild, severe), and target in a multi-task setup. Subtasks 1A and 1B are evaluated using the Micro-F1 score to address class imbalance. Subtask 1C is evaluated using the average Micro-F1 across the hate type, severity, and target predictions.

3.2 Dataset Overview

We used the workshop's (Hasan et al., 2025a) YouTube comments dataset (35.5k rows), which mirrors real-world trends where non-hate content clearly dominates. Specifically, 56.28% of comments fall under "None" for `hate_type`, 66.24% are rated "Little to None" for `severity`, and 59.75% target no particular group. In contrast, the rarest labels are *Sexism* at just 0.35%, *Severe* at 14.48%, and *Society* at 6.17%.

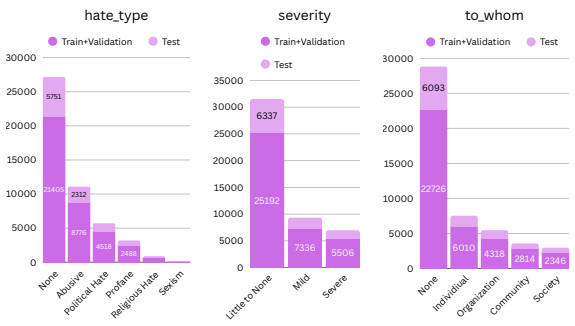

Figure 1: Distribution of classes

4 Methodology

We used a **cascading framework** to boost confidence and reduce error rates in major classes, raising the overall F1 score. Each subtask runs in two stages. Stage 1 separates hate from non-hate content with a binary classifier. Stage 2 applies multi-class classification to the samples labeled as hate in Stage 1. This stage can also reclassify content flagged as hate back into non-hate classes, giving any false positives a second chance. This setup helps the model generalize better across all categories. Because each level refines the previous one instead of forming a strict hierarchy, we call it cascading levels (Figure 2).

We selected **evaluation metrics** for each stage's needs. Stage 1's binary classifier uses the F_2-micro score, prioritizing recall to reduce false negatives in non-hate detection. Stage 2's multi-class classifier uses the F_1-micro score to balance precision and recall across all classes and ensure fair, accurate label distribution. These choices align with our cascading framework's goal of initial high-confidence filtering of major classes followed by comprehensive classification of all classes.

We used **k-fold cross-validation with an ensemble approach** (Figure 2) on each stage to improve generalization and make full use of our data. We merged the training and validation sets, then split them into folds. For Subtasks 1A and 1B, we set $k = 7$; for Subtask 1C's severity, $k = 9$. We applied `StratifiedKFold` to keep class ratios balanced across folds, producing k models at each stage. During inference, we ensembled those models by majority vote. In multi-class tasks, ties were broken by choosing the most frequent class in the training set.

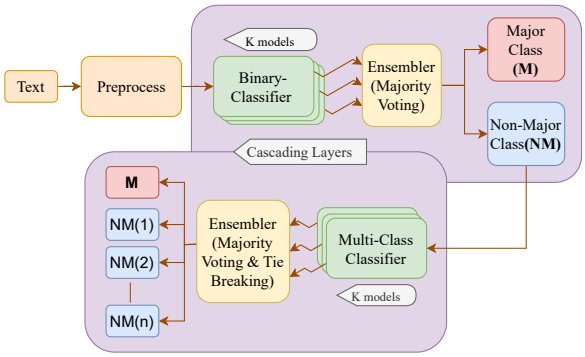

Figure 2: Cascading classification architecture with level-wise k-fold ensembling

To improve classification performance for Subtask 1B, we used an **enhanced `BanglaBERT` architecture** for the 2nd stage of the multi-class classifier (Figure 3). We processed each input by first encoding it into token IDs and attention masks, which were then passed through the pretrained BanglaBERT encoder to obtain contextual embeddings. To summarize token-level information, we applied an attention-pooling mechanism that dynamically weighted important tokens, producing a single pooled vector per instance. This pooled output was regularized using dropout and transformed

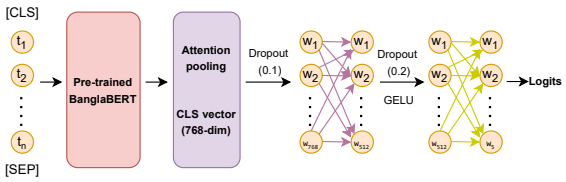

Figure 3: Enhanced BanglaBERT with Attention Pooling and Classification Head

via a dense layer followed by GELU activation to introduce non-linearity. A second dropout layer was applied to further reduce overfitting before final classification. The transformed features were then mapped to class logits using a linear layer. During training, we computed loss using CrossEntropy with optional label smoothing to improve stability on imbalanced data.

For Subtask 1C, we developed **three independent classification pipelines** shown in Figure 4, each trained separately; their outputs were then aggregated to improve the average micro-F1 score.

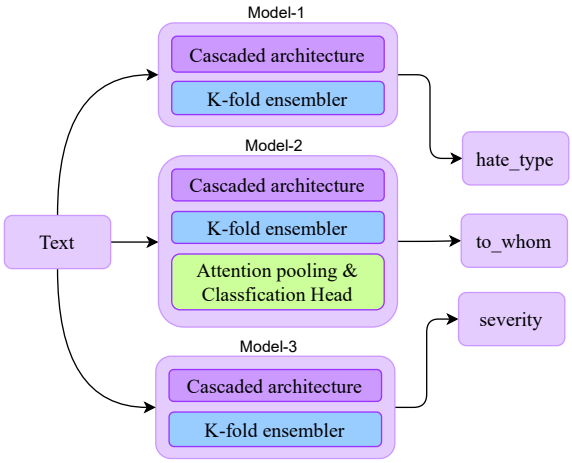

Figure 4: Aggregation of 3 models

5 Results and Findings

We evaluated all tasks using the final test dataset provided by the workshop organizers. Each subtask test set contains **10,200** instances.

The overall performance across the three subtasks is shown in Table 1. With F_1 scores of **0.7331**, **0.7356**, and **0.7392** for Subtasks 1A, 1B, and 1C, our cascaded framework with k-fold ensembling consistently delivers strong results. The reliability of our divide-and-conquer strategy is demonstrated by the improvements, which are not only evident in terms of F_1 but also balanced across precision and recall.

Subtask	Accuracy	Precision	Recall	F1 Score
1A	0.7331	0.7235	0.7331	0.7331
1B	0.7356	0.7276	0.7356	0.7356
1C	0.7392	0.7266	0.7392	0.7392

Table 1: Overall performance metrics across all subtasks.

For hate type classification, our system achieved an overall F_1 of **0.7331** with precision $= 0.7439$ and recall $= 0.7395$. As shown in Table 2, the *None* class achieved the highest performance ($F_1 = 0.8361$, precision $= 0.8221$, recall $= 0.8506$), indicating the effectiveness of the first-stage filtering in distinguishing non-hate content. Profane language also scored strongly ($F_1 = 0.7500$). However, performance was weaker for minority categories, particularly *Religious Hate* ($F_1 = 0.4531$) and *Sexism* (no correct predictions). The challenges with the *Sexism* category stem from its extreme underrepresentation, with only 9 examples in a training set of over 35k. In the test set, the first-stage classifier detected just 11 of 29 sexism instances (recall $= 0.38$), and none of these were correctly labeled as *Sexism* in the second stage, often falling under broader categories like *Abusive*.

Class	Precision	Recall	F1 Score
Abusive	0.5761	0.5272	0.5506
None	**0.8138**	**0.8597**	**0.8361**
Political Hate	0.6175	0.5730	0.5944
Profane	0.7309	0.7701	0.7500
Religious Hate	0.5385	0.3911	0.4531
Sexism	0.0000	0.0000	0.0000

Table 2: Per-class performance metrics for Subtask 1A.

In hate target prediction, our system obtained an overall F_1 of **0.7356**, with precision $= 0.7510$ and recall $= 0.7394$. Similar to Subtask 1A, the *None* category was detected with high performance ($F_1 = 0.8408$, recall $= 0.8611$), showing the effectiveness of cascading from the binary stage and the influence of the training dataset. Among hate-bearing categories, *Individual* ($F_1 = 0.6446$) and *Organization* ($F_1 = 0.6040$) were captured reasonably well. By contrast, *Community* ($F_1 = 0.4470$) and *Society* ($F_1 = 0.4499$) proved harder to detect, suggesting subtler linguistic markers and limited representation in the dataset.

The third subtask differs from the previous ones as it requires predicting hate type, target, and severity jointly. We reuse outputs from Subtasks 1A

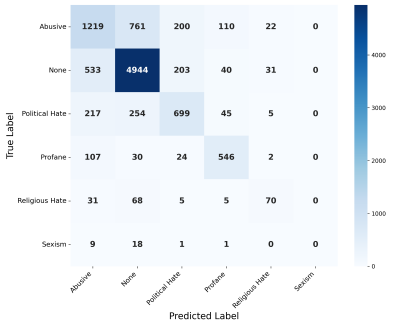

Figure 5: Confusion matrix for Subtask 1A.

Figure 6: Confusion matrix for Subtask 1B.

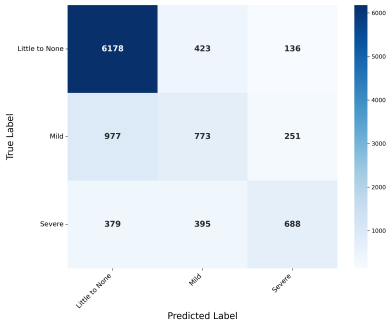

Figure 7: Confusion matrix for Subtask 1C.

Class	Precision	Recall	F1 Score
Community	0.4584	0.4361	0.4470
Individual	0.6699	0.6213	0.6446
None	**0.8190**	**0.8638**	**0.8408**
Organization	0.6146	0.5938	0.6040
Society	0.5166	0.3984	0.4499

Table 3: Per-class performance metrics for Subtask 1B.

and 1B, focus solely on severity here, then merge all three labels, simplifying the workflow and maintaining consistency. Our system achieved an overall F_1 of **0.7392** (precision $= 0.7471$, recall $= 0.7390$). It excelled on Little to None ($F_1 = 0.8658$, recall $= 0.9170$) but lagged on Severe ($F_1 = 0.5424$) and Mild ($F_1 = 0.4304$). This reflects the challenge of distinguishing between mild and severe hate, where subjectivity in annotation and overlapping cues complicate classification.

Class	Precision	Recall	F1 Score
Little to None	**0.8200**	**0.9170**	**0.8658**
Mild	0.4859	0.3863	0.4304
Severe	0.6400	0.4706	0.5424

Table 4: Per-class performance metrics for Subtask 1C - Hate Severity only.

The confusion matrices (Figures 5, 6, and 7) show systematic misclassification trends. In Subtask 1A, many instances of *Abusive*, *Profane*, and *Political Hate* are misclassified to the *None* class, with additional overlap between *Abusive* and both *Political Hate* and *Profane*. Subtask 1B shows a similar pattern, where most categories are misclassified as *None*, alongside confusions among *Community*, *Organization*, and *Society*. In Subtask 1C, errors largely reduce to *Little to None*, but we also observe overlap between *Mild* and *Severe*, with severe cases often predicted as mild. Together, these patterns underline both the dominance of neutral labels and the difficulty of separating closely related categories.

Key findings and overall insights:

- Subtask 1A and 1B results show that the system is highly effective at detecting the *None* category (non-hate content), which boosts global performance, but struggles with minority classes such as *Sexism*, *Religious Hate*, and *Community*; Subtask 1C reuses hate type and target labels, focuses on severity, and achieves an overall F_1 of 0.7392.
- Precision and recall remain high for dominant categories (e.g., *None*, *Little to None*), but recall drops significantly for underrepresented or ambiguous categories such as *Mild* severity, where annotation subjectivity is likely a factor.
- The cascaded framework achieves strong performance across all subtasks with balanced precision and recall; k-fold ensembling optimizes the use of limited low-resource language data to enhance overall results.

6 Conclusion

In this work, we proposed a two-stage cascaded framework for multi-label classification of hate speech in Bangla using k-fold ensembling. Our system ranked among the top submissions (4th, 1st, and 1st) in the shared task, consistently achieving balanced precision, recall, and F1-score across hate type, target, and severity. Although there are still issues with underrepresented and ambiguous labels like "Sexism" and "Mild severity," the framework was especially successful at handling dominant categories and filtering non-hate content. These results illustrate the potential of cascaded ensembling for low-resource hate speech detection as well as

the ongoing requirement for more balanced, richer datasets to capture subtle types of online abuse.

Limitations

The system lacks the implementation of contrastive learning. As the proposed problem to be solved has overlapping classes, contrastive learning could help. It pulls similar examples together and pushes dissimilar ones apart. This creates fine-grained distinctions between close classes like community and organizations.

The system lacks proper data augmentation. Minority classes suffer as a result. The model does well on dominant labels but fails to detect labels such as Sexism accurately. Per-class F1 scores drop despite strong global metrics.

Critical Analysis

We experimented with data augmentation through back-translation, but this strategy did not improve the performance of the models, most likely due to overlapping class annotations that limited the value of augmented samples.

We also explored an enhanced classification head for Subtasks 1A and 1C. Although initial experiments involved some hyperparameter tuning, the limited scope of training time and resources prevented us from fully optimizing the approach, and the gains remained inconsistent. Consequently, it was not included in our final system. In contrast, the same design showed clearer improvements in Subtask 1B, suggesting that with more extensive hyperparameter exploration, it could potentially benefit Subtasks 1A and 1C as well.

Lastly, model generalization was limited by dataset-specific issues like class imbalance and inconsistent annotations. Although methods such as weighted class and over-sampling were tried, they were unable to completely counteract the test dataset's bias toward majority classes.

References

Teshome Mulugeta Ababu, Michael Melese Woldeyohannis, and Emuye Bawoke Getaneh. 2025. Bilingual hate speech detection on social media: Amharic and afaan oromo. *Journal of Big Data*, 12(1):30.

Amit Kumar Das, Abdullah Al Asif, Anik Paul, and Md Nur Hossain. 2021. Bangla hate speech detection on social media using attention-based recurrent neural network. *Journal of Intelligent Systems*, 30(1):578–591.

Udoy Das, Karnis Fatema, Md Ayon Mia, Mahshar Yahan, Md Sajidul Mowla, Md Fayez Ullah, Arpita Sarker, and Hasan Murad. 2023. Emptymind at blp-2023 task 1: A transformer-based hierarchical-bert model for bangla violence-inciting text detection. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 174–178, Singapore. ACL.

Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *Acm Computing Surveys (Csur)*, 51(4):1–30.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025a. Llm-based multi-task bangla hate speech detection: Type, severity, and target. *arXiv preprint arXiv:2510.01995*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025b. Overview of blp 2025 task 1: Bangla hate speech identification. In *Proceedings of the Second International Workshop on Bangla Language Processing (BLP-2025)*, India. Association for Computational Linguistics.

Marzieh Mozafari, Reza Farahbakhsh, and Noël Crespi. 2020. Hate speech detection and racial bias mitigation in social media based on bert model. *PloS one*, 15(8):e0237861.

Maria Luisa Ripoll, Fadi Hassan, Joseph Attieh, Guillem Collell, and Abdessalam Bouchekif. 2022. Multi-lingual contextual hate speech detection using transformer-based ensembles. In *Proceedings of the Fourth Workshop on Hate Speech and Offensive Content Identification in Indo-European Languages (HASOC 2022)*, volume 3395 of *CEUR Workshop Proceedings*. CEUR-WS.org.

et al. Saha. 2023. Ensemble indicbert with naive bayes classifier and synthetic upsampling for hate-speech detection. In *Proceedings of CEUR Workshop*.

Yiwen Tang and Nicole Dalzell. 2019. Classifying hate speech using a two-layer model. *Statistics and Public Policy*, 6(1):80–86.

Hariram Veeramani, Surendrabikram Thapa, and Usman Naseem. 2023. Lowresourcenlu at blp-2023 task 1 & 2: Enhancing sentiment classification and violence incitement detection in bangla through aggregated language models. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 230–235, Singapore. ACL.

Bertie Vidgen and Leon Derczynski. 2020. Directions in abusive language training data, a systematic review: Garbage in, garbage out. *Plos one*, 15(12):e0243300.

Emily A Vogels. 2021. *The state of online harassment*, volume 13. Pew Research Center Washington, DC.

A Source Code

To foster reproducibility and future research, we release all implementation source code and resources at https://github.com/mahbubislammahim/Bangla-Hate-Speech-Identification

B Experimental Setup

B.1 Hardware and Runtime

All training experiments were performed using an NVIDIA GeForce RTX 4090 GPU with 24 GB memory. Additional experiments were occasionally run on free GPUs: Google Colab (T4, 16 GB) and Kaggle (P100 or T4, 15 GB). Training times were around an hour.

B.2 Text Normalization and Tokenization

All inputs are normalized using the BUET NLP normalizer to reduce script-level noise and standardize punctuation/spacing. Tokenization is performed with the `csebuetnlp/banglabert` WordPiece tokenizer, with sequences padded or truncated to a maximum of 256 tokens.

B.3 Cascaded Modeling

Each subtask uses a two-stage cascaded framework: stage 1 routes an input either to the "none" category or forwards it to Stage 2 for fine-grained classification. Stage 2 predicts the task-specific labels.

For Subtask 1A and 1C, both stages employ the BanglaBERT base model with standard configuration. For Subtask 1B, Stage 2 uses an enhanced classification head with attention pooling and a dense 512-GELU layer.

B.4 Training Protocol

We used the HuggingFace `Trainer` with AdamW, linear learning-rate scheduling with warmup, gradient accumulation, mixed precision (FP16), and weight decay. Stage 1 optimizes micro-F_2, while Stage 2 optimizes micro-F1. Cross-validation is stratified (`random_state=42`) and shuffling is enabled. Stage 1 models are selected by the best validation micro-F_2. Stage 2 models are selected by the best validation micro-F_1.

B.5 Inference and Ensemble Strategy

- **Stage 1:** average positive-class probability across folds.
- **Routing:** threshold τ decides between "none" vs. forwarding to Stage 2 ($\tau = 0.5$ for 1A/1B, $\tau = 0.3$ for 1C).
- **Stage 2:** per-fold argmax predictions are aggregated via majority vote; ties are broken deterministically using task-specific priority orders.

B.6 Hyperparameter Tuning

We adopt a pragmatic tuning strategy:

1. Base learning rate: 3×10^{-5} for Stage 1; 3–4$\times$ 10^{-5} for Stage 2.
2. Short training (2–3 epochs) with gradient accumulation.
3. Label smoothing of 0.1 for Subtasks 1A/1B to stabilize predictions and break ties.
4. Step-based validation for volatile multi-class heads.

B.7 Subtask Configurations

Subtask 1A (Hate Type; 6 classes) 7-fold CV (approx. 5074 rows/fold from 35522 total); Stage 1: {None, Hate}; Stage 2: {None, Religious Hate, Sexism, Political Hate, Profane, Abusive}.

Subtask 1B (Target; 5 classes) 7-fold CV (approx. 5074 rows/fold from 35522 total); Stage 1: {None, Hate}; Stage 2: {None, Society, Organization, Community, Individual}. Stage 2 uses enhanced head with attention pooling.

Subtask 1C (Severity; 3 classes) 9-fold CV (approx. 3946 rows/fold from 35522 total); Stage 1: {Little-to-None, Has-Severity}; Stage 2: {Little-to-None, Mild, Severe}. Stage 2 evaluates every 500 steps with $\tau = 0.3$ for improved recall. In this subtask, final outputs are merged into a single submission by joining 1A (hate type), 1B (target), and severity predictions on `id`.

B.8 Environment and Reproducibility

We use `transformers` ($\geq$ 4.21), `datasets` ($\geq$ 2.0), `accelerate` ($\geq$ 0.20), `PyTorch` ($\geq$ 1.12), `scikit-learn` (0.24), and `numpy` (1.20). Seeds are fixed at 42.

B.9 Training Analysis

To better illustrate the training process, we present example diagrams for Subtask 1B in Figure 8. We also include attention heatmap visualizations for 1B and 1C in Figure 9 and Figure 10, respectively. This demonstrates how the model attends to input tokens and provide insight into its decision-making process.

Hyperparameter	Stage 1 (Binary)	Stage 2 (6-way)
CV folds	7	7
Base model	csebuetnlp/banglabert	csebuetnlp/banglabert
Max seq. length	256	256
Optimizer / Scheduler	AdamW / Linear	AdamW / Linear
Learning rate	3×10^{-5}	3×10^{-5}
Epochs	2	2
Batch size (train / eval)	16 / 16	8 / 8
Grad. accumulation	2	2
Warmup ratio	0.1	0.1
Weight decay	0.01	0.01
Label smoothing	0.1	0.1
FP16	True	True
Ensemble	Prob. avg.	Majority vote + tie-break
Routing threshold τ	0.5	–

Table 5: Subtask 1A: Training setup.

Hyperparameter	Stage 1 (Binary)	Stage 2 (5-way, Enhanced)
CV folds	7	7
Base model	csebuetnlp/banglabert	csebuetnlp/banglabert + attention pooling
Head	Standard	Dense(512)+GELU + Dropout + Linear
Max seq. length	256	256
Optimizer / Scheduler	AdamW / Linear	AdamW / Linear
Learning rate	3×10^{-5}	4×10^{-5}
Epochs	2	3
Batch size (train / eval)	16 / 16	16 / 16
Grad. accumulation	2	2
Warmup ratio	0.1	0.1
Weight decay	0.01	0.01
Label smoothing	0.1	0.1
FP16	True	True
Ensemble	Prob. avg.	Majority vote + tie-break
Routing threshold τ	0.5	–

Table 6: Subtask 1B: Training setup.

Hyperparameter	Stage 1 (Binary)	Stage 2 (3-way)
CV folds	9	9
Base model	csebuetnlp/banglabert	csebuetnlp/banglabert
Max seq. length	256	256
Optimizer / Scheduler	AdamW / Linear	AdamW / Linear
Learning rate	3×10^{-5}	4×10^{-5}
Epochs	2	3
Batch size (train / eval)	16 / 16	16 / 16
Grad. accumulation	2	2
Warmup ratio	0.1	0.1
Weight decay	0.01	0.01
Label smoothing	–	–
FP16	True	True
Ensemble	Prob. avg.	Majority vote + tie-break
Routing threshold τ	0.3	–

Table 7: Subtask 1C: Training setup.

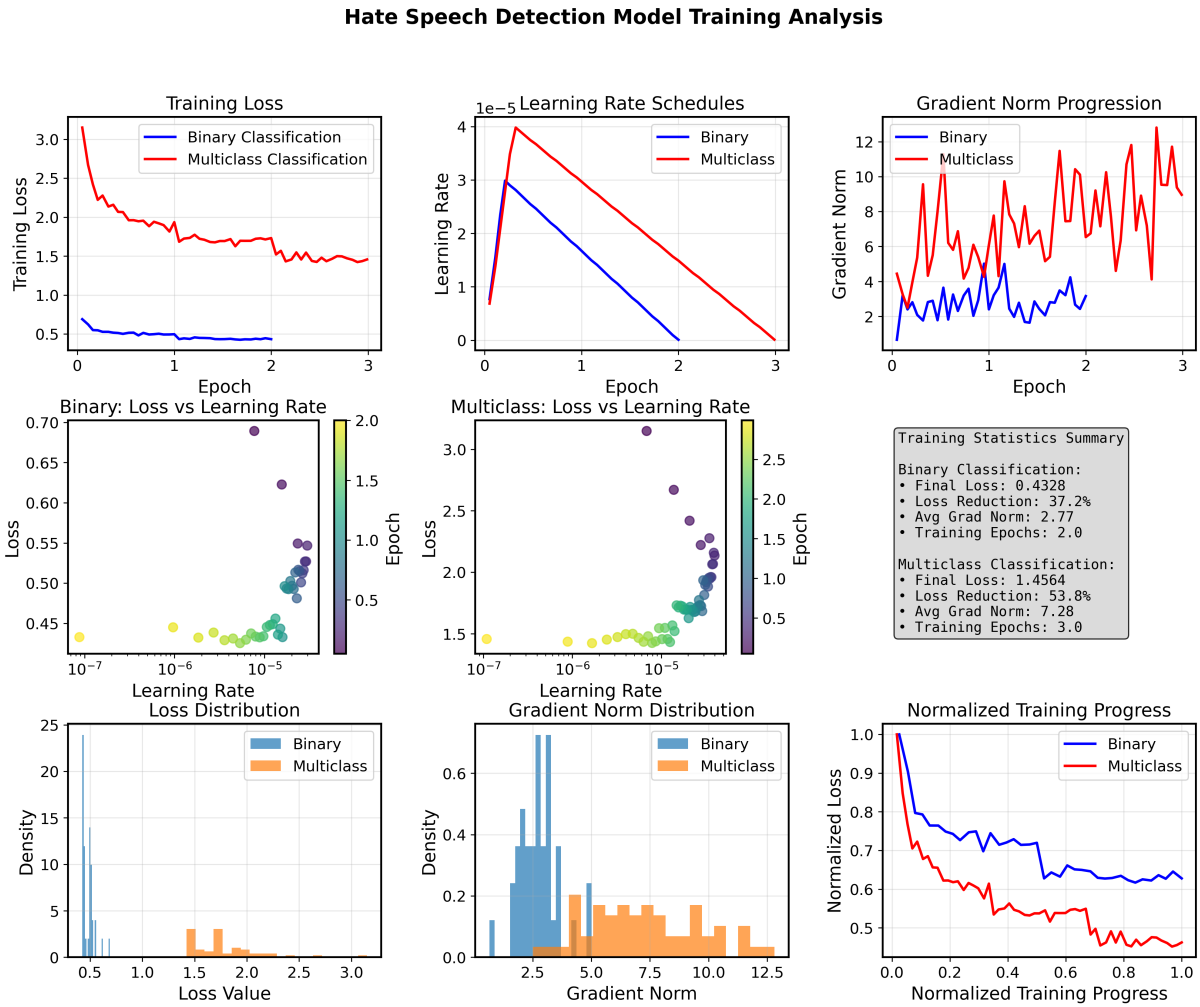

Figure 8: Training analysis.

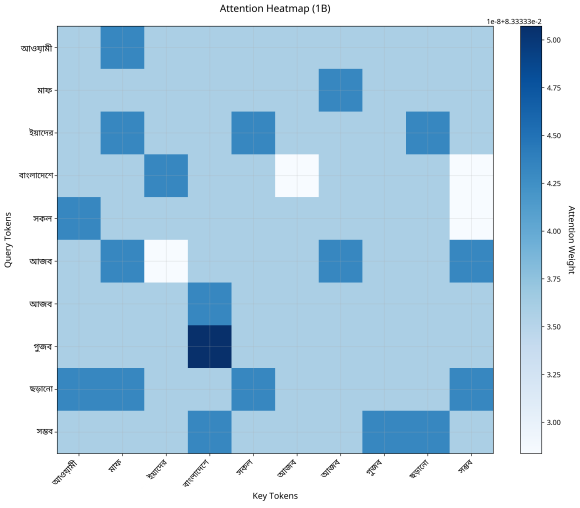

Figure 9: Attention heatmap for Subtask 1B.

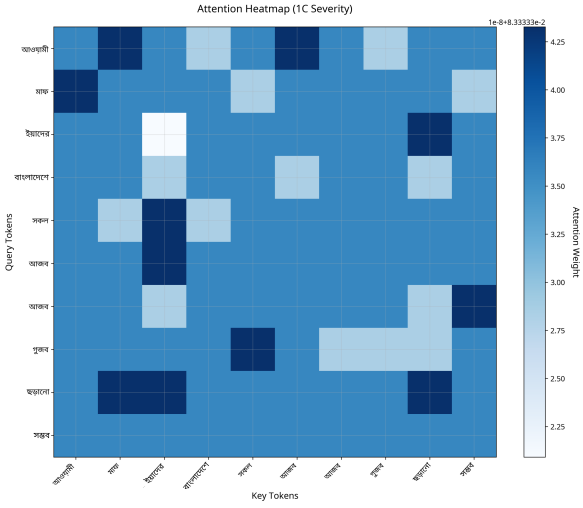

Figure 10: Attention heatmap for Subtask 1C.