AyahVerse at MAHED Shared Task: Fine-Tuning ArabicBERT with Preprocessing for Hope and Hate Detection

Ibad-ur-Rehman Rashid, Muhammad Hashir Khalil
Government Post Graduate College, Mansehra, Pakistan
ibad@gcm.edu.pk, hashirkhalil3@gmail.com

Abstract

We participated in Subtask 1 of the MAHED Shared Task 2025, which focuses on detecting hope, hate, and not applicable labels in Arabic content. In this work, we tested a multiclass classifier for hope, hate, and not applicable label detection from a dataset provided by organizers as Subtask 1. We approach the task by two methods. The first one is a fine-tuned Arabic model, ArabicBERT, on a multiclass classification task. The second one is a two-step stacked architecture. Both of them include a dedicated pipeline for specific Arabic preprocessing with different techniques. Official results are 0.38 F1 score on the validation set and 0.47 on the test set with a single multiclass classifier. Post-submission improvements resulted in macro-F1 scores of 0.60(validation) and 0.63(test) for the single classifier, and 0.59(validation) and 0.91(test) for the stacked classifier.

1 Introduction

The MAHED Shared Task 2025 (Subtask 1)(Zaghouani et al., 2025) focuses on detecting hope and hate emotions in Arabic text from social media content and tweets about Middle East conflict. The dataset(Zaghouani and Biswas, 2025a) is annotated with the labels "hope" "hate" and "not applicable". Training on this task by participants contributes to new methodologies in Arabic NLP, especially in the classification of hate and hope speech and multimodal content detection for understanding online discourse and promoting positive engagement in social media communities. During the task we discover many challenges, including overfitting, underfitting, class imbalance, and label inconsistencies in a few cases. Our code is available at GitHub[1].

We discovered that proper Arabic preprocessing significantly improves performance. Undersampling and oversampling led to overfitting, and adjusted weights resulted in a performance increase in both validation and test sets. The difference between a single multiclass classifier and a stacked classifier is minimal, however removing emojis in stack classifier configuration resulted in a notable improvement, reaching 0.91 F1 score on the official test set.

The preprocessing pipeline consists of some general language preprocessing techniques like URL and hashtag removal, as well as some specific Arabic language preprocessing with different techniques like handling class imbalance, diacritization, and Arabic letter normalization.

We focused on two model architectures. The first one is a fine-tuned multiclass ArabicBERT(Safaya et al., 2020)[2] for predicting hate, hope, or not applicable labels in the text. The second model architecture consists of two binary fine-tuned 'ArabicBERT' classifiers for detecting hate and hope speech, and this layer is stacked with a final logistic regression meta-classifier.

On the official leaderboard, our system ranked 25th out of 25 teams, achieving a macro-F1 of 0.48 on the test set. Post-submission results increase F1 scores to around 0.60 in the validation set, and 0.91 in the test set in different configurations.

2 Background

2.1 Task setup

The shared task required participants to classify Arabic text into one of the classes:

- *hope* hopeful messages
- *hate* hateful or abusive messages

[1] https://github.com/Ebad-urRehman/MAHED_2025_subtask1_hate_and_hope/

[2] https://huggingface.co/asafaya/bert-base-arabic

Proceedings of The Third Arabic Natural Language Processing Conference, pages 670–676
November 8-9, 2025

- *not_applicable* neutral content

Example

- أَنَا مُتَحَمِّسٌ لِلشُّعُورِ بِالرُّوحِ الْخَاصَّةِ الَّتِي سَتَجْلِبُهَا إِلَى مَنْزِلِنَا
  → *hope*
- أنتم لا تستحقون الاحترام
  → *hate*
- تحب تقول ايه لتميم بن موزه يا جبان ما انت جنست ناس من السنغال للدفاع عنك
  → *not_applicable*

2.2 Dataset

We used the provided MAHED 2025 dataset (Zaghouani and Biswas, 2025a), consisting of Arabic social media posts gathered considering the linguistic diversity and dialect variations. The dataset is labeled for hope, hate, and not applicable categories, and it contains train, test, and validation splits D. The split used for the test set during training is 0.2. We use only training data for training of all models.

The full dataset size is 9843, with 6890 for training, 1476 for validation, and 1477 for testing.

Datasets for Subtask 2 are (Zaghouani and Biswas, 2025b) and (Zaghouani et al., 2024) which are not used in this work.

2.3 Track

We participated in Subtask 1 of the MAHED Shared Task 2025.

2.4 Related Works

Recent studies on Arabic hate speech, including (Althobaiti, 2022) provide a comparison between the BERT-based approach and two machine learning techniques, demonstrating that BERT-based models are more effective. They also experimented with incorporating sentiment information along with text into the BERT model and converting emojis to textual descriptions. While sentiment features slightly improved performance, the effect of emoji descriptions varied depending on class distribution.

(Almaliki et al., 2023) is a benchmark model for Arabic offensive language detection, which is classified into three classes: normal, abuse, and hate speech. Another study, (Aldjanabi et al., 2021) Using a Cross-Corpora Multi-Task Learning Model,' trained a model on a wide variety of datasets in multiple tasks; their model was fine-tuned on the MarBERT (Abdul-Mageed et al., 2021) Arabic model. Similarly, 'BERT-CNN for Offensive Speech Identification in Social Media' combines CNN with BERT and demonstrates the effectiveness of the ArabicBERT model when combined with CNN.

A multi-task learning strategy was more recently experimented with by (Abdelsamie et al., 2026) to address dialectal variations in Arabic hate speech detection. Their model captures the distinctive features of each of the five Arabic dialects (Egyptian, Levant, Saudi, Algerian, and Gulf) while leveraging shared knowledge across them. With remarkable F1 scores of 0.98, 0.84, 0.85, 0.76, and 0.80 for the corresponding dialects, it outperformed single-task models by about 14%.

In contrast to these studies, our system differs in the datasets used and the training approaches we employed. We experimented with different levels of preprocessing, including Arabic letter normalization, diacritics, and tatweel removal. The approaches we explored were two main strategies: a single multiclass classifier and a stacked binary ensemble of classifiers with two approaches. In the ensemble, one variant includes all 'not applicable' labels in both binary classifiers, while another variant splits 'not applicable' labels into two subsets to use separately with both binary classifiers.

3 System Overview

We choose ArabicBERT because it is one of the high-performing models of nlp arabic as per (Alammary, 2022). We aim to test it for multiclass classification with a single classifier as well as a stacked multilayer architecture. We trained and tested our model on provided datasets only.

3.1 Preprocessing

At first we implemented simple preprocessing techniques like URL, hashtag, handle, and stopword removal.

In later versions we included some specific Arabic preprocessing techniques including:

1. Mapped emojis to Arabic text equivalents using the defined 'emoji to text' dictionary.
2. Character normalization e.g., إ أ آ → ا; ى → ي.
3. Diacritics and tatweel removal for a unified formatted dataset and noise removal for better model understanding.

4. We also implemented some general preprocessing techniques like URL, hashtag, and handle removals and whitespace normalization, without stopword removal, as they carry context and meaning in Arabic.

5. For handling class imbalance, we used different techniques like undersampling, oversampling, and adjusting class weights.

3.2 Model

We tested different model configurations. Two of the main architectures are (i) a single multiclass classifier and stacked binary ensemble with a meta classifier.

3.2.1 For Single Multiclass Classifier:

Our system follows a preprocess → tokenize → classify → evaluate pipeline.

At the start, we experimented with a deeper classification head consisting of an additional fully connected layer of size 256 with a ReLU activation function on top of ArabicBERT. This 256-dimensional layer was connected to the final output layer, producing three logits, using the same general preprocessing pipeline. However, this design showed poor generalization. We then tried a simpler classifier where ArabicBERT was directly connected to a linear layer producing three logits, followed by dropout with improved preprocessing. This setup gave better performance and stability on both training and validation and was therefore chosen as our final classifier design.

After this we tested the selected model with different levels of preprocessing and class imbalance handling techniques like undersampling, oversampling, and adjusting class weights for loss calculation (see Appendix E). In addition to this multiclass approach, we also designed and tested a stacked binary ensemble architecture.

3.2.2 For stacked binary ensemble:

Our system follows a preprocess → tokenize → classify → ensemble → evaluate pipeline.

It is a two-step stacked architecture, where one ArabicBERT model was trained to classify hope vs. not applicable and another to classify hate vs. not applicable, with their probability outputs fed into a logistic regression meta-classifier for final prediction. The binary ArabicBERT model configurations are kept the same as the single multiclass classifier. Like the single multiclass classifier, we also tested this for different levels of preprocessing and class imbalancing handling techniques.

Figure 1: Meta Classifier Architecture

In another variant, as shown in the figure below, we applied an additional preprocessing step where the not applicable class was divided into two equal parts. One part was used alongside the hope examples, and the other part was paired with the hate examples for training. We tried this because binary classifiers (hope and hate) are underperforming on hate and hope classes due to more examples of not applicable in dataset. Binary class performance on this new architecture improves; however, performance of the meta-classifier in both stacked architectures yields close results.

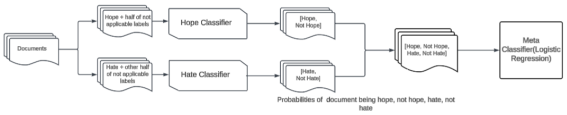

Figure 2: Meta Classifier with not applicable labels split

3.3 Challenges

The MAHED 2025 hope and hate text classification dataset is highly imbalanced. We explored several strategies, including oversampling, undersampling, and adjusting class weights. These approaches lead to more overfitting and underfitting and eventually a low F1 score for validation and test sets, except for adjusting class weights that gives an increase in F1 score.

However, the best results were achieved by performing specific Arabic preprocessing, without applying class imbalance techniques. Increasing the number of training epochs from 3 to 8 slightly improves the performance.

Earlier we tried to test without stopwords, but later we decided to retain them, and this is also a reason for improved performance in later experiments.

4 Experimental Setup

We fine-tuned ArabicBERT[3] using the Transformers library. Our model used the AdamW optimizer with the CrossEntropyLoss function, going through a training of 8 epochs. The max sequence length for sentences is 128, the single batch size is 16, and the learning rate is 2×10^{-5}.

We trained, validated, and tested our model using the official datasets. During training, 20% of the data was reserved for testing. Training of all models was performed only on `train.csv`.

Our implementation used Python 3.13, PyTorch, HuggingFace Transformers, scikit-learn, NLTK, pandas. Experiments are conducted on Google colab GPUs T4, L4 and A100. Consuming approximately 70 compute units on training, and testing.

5 Results

5.1 Official Results from scoring files

Official results from the scoring files show low scores, because model is underfitting due to exclusion of proper arabic preprocessing like diacritization, Arabic letter normalization, and converting emojis to arabic text. Another reasons of low F1 scores are custom layer on top of bert classifier, and less number of epochs. We also have not experimented with stacked classifier at that time. The official results of the scoring files are shown in Table 1.

Metrics (Macro)	F1	Accuracy	Precision	Recall
Validation File	0.376	0.649	0.376	0.377
Test File	0.465	0.624	0.458	0.474

Table 1: Official results.

5.2 Post Submission Results

F1 scores significantly improves in post submission experiments, because of specific arabic preprocessing pipeline and increased number of epochs.

Metrics (Macro)	F1	Accuracy	Precision	Recall
Validation File	0.603	0.621	0.596	0.612
Test File	0.608	0.632	0.628	0.594

Table 2: Post-submission performance (Macro metrics) of Single Multiclass Classifier with weight adjustments.

[3] https://huggingface.co/asafaya/bert-base-arabic

Metrics (Macro)	F1	Accuracy	Precision	Recall
Validation File	0.60	0.61	0.59	0.61
Test File	0.63	0.63	0.62	0.64

Table 3: Post-submission performance (Macro metrics) of Stacked Binary Ensemble Classifier.(with emojis replaced with arabic text)

5.3 Analysis

5.3.1 Analysis of Single Multiclass Classifier under different strategies

Our single multiclass classifier achieves an F1 score of 0.71 on training, 0.57 on validation, and 0.60 on test sets. With adjusted weights, our multiclass classifier achieves 0.98 on training, 0.60 on validation, and 0.63 on testing. With oversampling, we observe overfitting because duplicate examples may make the model memorize some examples instead of generalizing. With undersampling, too, we observe overfitting because of missing examples in training data, which makes the model perform poorly on test data.

Model	Train/Test	Validation	Test
No Strategy	0.717	0.578	0.608
With Oversampling	0.984	0.252	0.244
With Undersampling	0.809	0.236	0.231
With Adjusted weights	0.986	0.603	0.635

Table 4: Macro-F1 comparison across different training strategies for the Single Multiclass Classifier details in Appendix Table 6. A

With emojis removed instead of being replaced with Arabic text and no class imbalance technique applied, the model gives F1 scores of 0.60 and 0.62 for validation and test files, respectively. While removing emojis and adjusted weights gives an F1 score of 0.58 on validation and 0.64 on testing, details in Appendix Table 7. A

5.3.2 Analysis of Stacked Binary Ensemble Classifier under different strategies

Our two-layer stacked binary ensemble classifier achieves an F1 score of 0.60 on the validation set and 0.63 on the test set when emojis are replaced with Arabic words. When emojis are completely removed, the F1 score changes to 0.59 on validation and 0.91 on the test set.

In the second variant of the stacked binary ensemble, which includes an additional preprocessing step that splits the not applicable label into two subsets, the F1 score is 0.58 for validation and 0.63 for the test set. Excluding emojis in this configu-

ration results in an F1 score of 0.59 on validation and 0.65 on the test set.

Metrics (Macro)	F1	Accuracy	Precision	Recall
Training Performance	0.90	0.61	0.89	0.90
Validation File	0.60	0.61	0.59	0.61
Val (without emojis)	0.59	0.61	0.59	0.59
Test File	0.63	0.63	0.62	0.64
Test (without emojis)	0.91	0.91	0.92	0.91

Table 5: Post-submission performance of the Stacked Binary Ensemble classifier, details in Appendix Table 8. A

System Error Examples. The increase in F1-scores due to emoji removal in the test set might be due to sarcasm examples where a laughing emoji is used, but the overall text is hate. In such cases, removing emojis instead of converting them into Arabic equivalent words helps the models understanding.

For example, the translation of [laughing] emoji in the dictionary is ضحك (laughing), which gives a hopeful sentiment. However, in the dataset it appears frequently in hate and not-applicable examples as shown in Appendix. B

Some annotation mistakes also contributed to poor model understanding and thus lower performance as shown in Appendix. C

Our system sometimes overfits, especially with oversampling or deeper classification heads. This causes high training F1 scores but poor performance in validation / test sets. The performance of the system can be improved by better data quality, proper preprocessing, and the use of a suitable class imbalance handling technique.

6 Conclusion

In this work, we aimed to classify Arabic social media posts into hope, hate, and not applicable categories as part of MAHED Shared Task 2025 Subtask 1. We developed a multiclass classifier based on the Arabic model ArabicBERT, fine-tuned on the competition dataset, and achieved an F1 score of 0.60 and 0.63 on the given validation and test datasets. With our other approach, we tested stacked binary ensemble models and achieved F1 scores of 0.59 and 0.91 on validation and test sets. Specific Arabic preprocessing choices, like skipping stopword removal, normalizing Arabic letters, removing diacritics, and tatweel, resulted in improvement of the F1 score. The adjusted class weights technique for handling class imbalance performs better as compared to other techniques like oversampling and undersampling.

Acknowledgments

We would like to thank contributors and co-authors of this task, including Muhammad Hashir Khalil, Junaid Hussain, Syed Saqlain Gillani, and Shahid Khan, for providing assistance in writing this paper. We also thank the organizers of the MAHED Shared Task 2025 for providing the dataset and evaluation platform. We sincerely thank the anonymous reviewers for their insightful comments and constructive feedback, which greatly improved the quality of this work. Furthermore, we also acknowledge Hugging Face for open-sourcing ArabicBERT and Google Colab for providing the GPU resources used in training and evaluation.

References

Mahmoud Mohamed Abdelsamie, Shahira Shaaban Azab, and Hesham A. Hefny. 2026. The dialects gap: A multi-task learning approach for enhancing hate speech detection in arabic dialects. *Expert Systems with Applications*, 295:128584.

Muhammad Abdul-Mageed, AbdelRahim Elmadany, and El Moatez Billah Nagoudi. 2021. ARBERT & MARBERT: Deep bidirectional transformers for Arabic. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7088--7105, Online. Association for Computational Linguistics.

Ali Saleh Alammary. 2022. Bert models for arabic text classification: A systematic review. *Applied Sciences*, 12(11).

Wassen Aldjanabi, Abdelghani Dahou, Mohammed A. A. Al-qaness, Mohamed Abd Elaziz, Ahmed Mohamed Helmi, and Robertas Damaševičius. 2021. Arabic offensive and hate speech detection using a cross-corpora multi-task learning model. *Informatics*, 8(4).

Malik Almaliki, Abdulqader M. Almars, Ibrahim Gad, and El-Sayed Atlam. 2023. Abmm: Arabic bert-mini model for hate-speech detection on social media. *Electronics*, 12(4).

Maha Jarallah Althobaiti. 2022. Bert-based approach to arabic hate speech and offensive language detection in twitter: Exploiting emojis and sentiment analysis. *International Journal of Advanced Computer Science and Applications*, 13(5).

Ali Safaya, Moutasem Abdullatif, and Deniz Yuret. 2020. KUISAIL at SemEval-2020 task 12: BERT-CNN for offensive speech identification in social media. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2054--2059, Barcelona (online). International Committee for Computational Linguistics.

Wajdi Zaghouani and Md Rafiul Biswas. 2025a. An annotated corpus of arabic tweets for hate speech analysis. *arXiv preprint arXiv:2505.11969*.

Wajdi Zaghouani and Md Rafiul Biswas. 2025b. Emohopespeech: An annotated dataset of emotions and hope speech in english and arabic. *arXiv preprint arXiv:2505.11959*.

Wajdi Zaghouani, Md Rafiul Biswas, Mabrouka Bessghaier, Shimaa Ibrahim, Georgios Mikros, Abul Hasnat, and Firoj Alam. 2025. MAHED shared task: Multimodal detection of hope and hate emotions in arabic content. In *Proceedings of the Third Arabic Natural Language Processing Conference (ArabicNLP 2025)*, Suzhou, China. Association for Computational Linguistics.

Wajdi Zaghouani, Hamdy Mubarak, and Md Rafiul Biswas. 2024. So hateful! building a multi-label hate speech annotated arabic dataset. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 15044--15055.

A Tables

B Emoji Effects Example

[laughing emojis]

انا خايفه ادخل تويتر مو بس حسابك

"I'm afraid to log into Twitter, not just your account."
Label: Not Applicable

نعوضكم بأيش يعني قم انقلع بس يالعراكي [laughing emojis]
"How do we compensate you? [laughing emojis] Just get lost, you brat."
Label: Hate

C Incorrect Annotation Example

علشان انتى بومه وفقريه وهنفوز بأداء ونتيجه ان شاء الله الاهلى بيلعب كرة حلوة مع الفرق اللى بتلعب كرة وبعدين انتى تعرفي عن الكرة ايه غير انها مدورة [laughing emojis]
"Because you are an owl and a poor girl, and we will win with performance and results, God willing. Al-Ahly plays good football with teams that play football. And what do you know about football other than that it is round? [laughing emojis]"
Annotated Label: Hope
Correct Label: Hate
Model Predicted: Hope

السعوديه الثالثه في السمنه تباً لكم مرهلين وش قعدني معكم
"Saudi Arabia is third in obesity. Damn you, you idiots. What made me stay with you?"
Annotated Label: Hope
Correct Label: Hate
Model Predicted: Hate

D Dataset Train, Validation, and Test set Details

The full dataset size is 9843, with 6890 for training, 1476 for validation, and 1477 for testing.

E Adjusting Weights Logic

```
class_counts = np.bincount(labels_raw)
class_weights = 1. / class_counts
weights_tensor =
   torch.tensor(class_weights, dtype=torch.float)
     .to(device)
criterion =
     nn.CrossEntropyLoss(weight=weights_tensor)
```

Model	Metrics	Train/Test	Validation	Test
No Strategy	Macro-f1	0.717	0.578	0.608
	Macro-accuracy	0.757	0.621	0.632
	Macro-precision	0.733	0.602	0.628
	Macro-recall	0.731	0.563	0.594
With Oversampling	Macro-f1	0.984	0.252	0.244
	Macro-accuracy	0.985	0.411	0.392
	Macro-precision	0.983	0.243	0.231
	Macro-recall	0.986	0.265	0.260
With Undersampling	Macro-f1	0.809	0.236	0.231
	Macro-accuracy	0.806	0.329	0.319
	Macro-precision	0.789	0.259	0.251
	Macro-recall	0.856	0.219	0.215
With Adjusted Weights	Macro-f1	0.986	0.603	0.635
	Macro-accuracy	0.986	0.621	0.645
	Macro-precision	0.982	0.596	0.633
	Macro-recall	0.990	0.612	0.639

Table 6: Performance comparison of different training strategies for the Single Multiclass Classifier.

Metrics(Macro)	F1	Accuracy	Precision	Recall
Test(train split) set	0.717	0.757	0.733	0.731
Test(train split) set without emojis	0.984	0.985	0.984	0.985
(without emojis, with adjusted weights)	0.990	0.991	0.987	0.993
Validation File	0.578	0.621	0.602	0.563
Validation File without emojis	0.600	0.636	0.618	0.589
(without emojis, with adjusted weights)	0.589	0.608	0.585	0.594
Test File	0.608	0.632	0.628	0.594
Test File without emojis	0.622	0.640	0.635	0.613
(without emojis, with adjusted weights)	0.640	0.649	0.639	0.641

Table 7: Post Submission Performance metrics on evaluation and test datasets of Single Multiclass Classifier

Condition	Label	Macro-F1	Macro-Accuracy	Macro-Precision	Macro-Recall
With emojis replaced	Hope	0.73	0.77	0.72	0.74
	Hate	0.76	0.84	0.74	0.79
	Not Applicable	0.63	0.63	0.62	0.64
With no emojis	Hope	0.93	0.94	0.93	0.94
	Hate	0.95	0.97	0.96	0.94
	Not Applicable	0.92	0.91	0.91	0.91
Data split + emojis replaced	Hope	0.71	0.73	0.71	0.76
	Hate	0.78	0.84	0.76	0.82
	Not Applicable	0.63	0.62	0.62	0.67
Data split + no emojis	Hope	0.74	0.78	0.73	0.76
	Hate	0.78	0.86	0.78	0.77
	Not Applicable	0.65	0.66	0.65	0.65

Table 8: Performance of different setups for the Stacked Binary Ensemble Classifier on Test file.