GATech at AbjadGenEval Shared Task: Multilingual Embeddings for Arabic Machine-Generated Text Classification

Ahmed Khaled Khamis
Georgia Institute of Technology
akhamis6@gatech.edu

Abstract

We present our approach to the AbjadGenEval shared task on detecting AI-generated Arabic text. We fine-tuned the multilingual E5-large encoder for binary classification, and we explored several pooling strategies to pool token representations, including weighted layer pooling, multi-head attention pooling, and gated fusion. Interestingly, none of these outperformed simple mean pooling, which achieved an F1 of 0.75 on the test set. We believe this is because complex pooling methods introduce additional parameters that need more data to train properly, whereas mean pooling offers a stable baseline that generalizes well even with limited examples. We also observe a clear pattern in the data: human-written texts tend to be significantly longer than machine-generated ones.

1 Introduction

When ChatGPT (OpenAI et al., 2024) and similar models started producing fluent Arabic text, it became clear that detection tools would need to catch up. Unlike English, where several detectors already exist, Arabic has received far less attention, mostly due to its morphological complexity and the diversity of written styles across regions. AbjadGenEval (Abudalfa et al., 2025) (Ezzini et al., 2026) (Lamsiyah et al., 2025) addresses this gap with a shared task specifically for Arabic machine-generated text detection.

We approached this as a classification problem: take a pre-trained multilingual encoder (Vaswani et al., 2023) (E5-large) (Wang et al., 2024), add a classification head, and fine-tune on the provided data. The interesting part was figuring out how to pool the token representations. We implemented weighted layer pooling (learning which transformer layers matter most), attention-based pooling (learning which tokens to focus on), and gated fusion (learning how to combine multiple pooling outputs). After all that engineering, plain mean pooling gave us the best results.

This paper makes three contributions:

- A systematic comparison of pooling strategies for Arabic text classification, demonstrating that simple mean pooling outperforms complex learned aggregation methods on limited training data.
- Observations about the dataset: human-written texts average 632 words versus 303 for machine-generated
- A training recipe with layer-wise learning rate decay and multi-sample dropout regularization

Our final system scores 0.75 F1 on the shared task test set. [1]

2 Background

2.1 Task Setup

The AbjadEval task frames Arabic AI-generated text detection as a binary classification problem. Given an input text x, the system must predict a label $y \in \{\text{human}, \text{machine}\}$ indicating whether the text was written by a human or generated by an AI system.

2.2 Dataset

The competition dataset consists of 5,298 Arabic text samples with a balanced class distribution (50% human, 50% machine-generated). Table 1 summarizes the dataset statistics.

A notable characteristic of the dataset is the substantial length difference between classes: human-written texts average 632 words compared to only

[1] Code: https://github.com/KickItLikeShika/abjadgeneval

Statistic	Human	Machine
Samples	2,649	2,649
Avg. Words	632.0	303.0
Avg. Characters	3,806.7	1,865.5
Max Words	3,068	1,969

Table 1: Dataset statistics by class. Human-written texts are significantly longer than machine-generated texts.

303 words for machine-generated texts. This suggests that text length could be a discriminative feature, though our model learns to capture more nuanced patterns.

2.3 Related Work

Previous approaches to AI-generated text detection have employed statistical methods (Gehrmann et al., 2019), fine-tuned language models, and watermarking techniques (Kirchenbauer et al., 2024). For Arabic specifically, transformer-based models like AraBERT (Antoun et al., 2021) and CAMeL-BERT (Inoue et al., 2021) have shown strong performance on various NLP tasks. Recent work on multilingual text embeddings, particularly the E5 family (Wang et al., 2024), has demonstrated excellent cross-lingual transfer capabilities.

More recent work has shifted toward fine-tuning language models directly for detection. The intuition is simple: if a model like BERT (Vaswani et al., 2023) can learn what "natural" text looks like during pre-training, it should also be able to learn what generated text looks like with supervised fine-tuning. DetectGPT (Mitchell et al., 2023) took a different approach, using perturbation-based methods that don't require any training data at all—though these zero-shot methods typically lag behind supervised ones when labeled data is available.

3 System Overview

3.1 Model Architecture

Our system is built on the multilingual E5-large encoder (Wang et al., 2024), which consists of 24 transformer layers with a hidden size of 1,024. We add a classification head on top of the pooled representations.

The architecture follows a standard encoder-classifier setup. Input text is first tokenized and passed through the E5-large encoder, which produces a contextualized representation for each token. These token-level representations are then aggregated into a single fixed-size vector using a pooling operation. Finally, this pooled vector passes through a classification head that outputs probabilities for each class (human or machine).

3.2 Pooling Strategies

We experimented with several pooling strategies before settling on mean pooling: **Mean Pooling:** The simplest approach, where we average the hidden states across all non-padded tokens. Each token contributes equally to the final representation.

Weighted Layer Pooling: Instead of using only the final transformer layer, this method learns to combine outputs from multiple layers. The intuition is that different layers capture different types of information: lower layers tend to encode surface-level features while higher layers capture more semantic content. We assign a learnable weight to each layer, and take a weighted average, with weights normalized using softmax.

Multi-Head Attention Pooling: Rather than treating all tokens equally, this approach learns which tokens to focus on. We use 8 learnable query vectors, each attending to the token sequence independently. The resulting 8 context vectors are concatenated and projected back to the hidden dimension.

Gated Fusion: When combining multiple pooling methods, we use learned sigmoid gates to control how much each pooling output contributes. The gates are computed from the concatenation of all pooling outputs, allowing the model to dynamically weight different representations based on the input.

3.3 Classification Head

The pooled representation passes through a feed-forward layer with layer normalization, GELU activation, and dropout before the final classifier. We also use multi-sample dropout (Inoue, 2020): during training, we apply 5 different dropout masks (with rates 0.1, 0.15, 0.2, 0.25, and 0.3) and average the resulting logits. This acts like a small ensemble within a single forward pass, improving regularization without additional inference cost.

3.4 Loss Function

We use Focal Loss (Lin et al., 2018) instead of standard cross-entropy. Focal loss down-weights easy examples and focuses training on harder cases by scaling the loss based on prediction confidence.

4 Experimental Setup

4.1 Data Split

We trained on the full competition training set containing 5,298 samples. Due to the blind test evaluation setup.

4.2 Hyperparameters

Table 2 details our training configuration.

Parameter	Value
Model	multilingual-e5-large
Max Sequence Length	512 tokens
Batch Size	16
Gradient Accumulation	4 steps
Effective Batch Size	64
Learning Rate	2×10^{-5}
Weight Decay	0.01
LLRD Decay Factor	0.95
Epochs	2
Warmup Ratio	10%
Scheduler	Cosine with warmup

Table 2: Training hyperparameters.

4.3 Layer-wise Learning Rate Decay

We apply layer-wise learning rate decay (LLRD) to prevent catastrophic forgetting of pretrained knowledge. Lower transformer layers receive smaller learning rates.

4.4 Other Implementation Details

We utilized Dynamic Padding to ensure that sequences are padded to max length within each batch for efficiency.

5 Results

Our system achieved an **F1 score of 0.75** on the official test set using mean pooling with the E5-large encoder.

5.1 Pooling Strategy Results

Table 3 presents our comparison of pooling strategies during development.

Mean pooling demonstrated superior generalization on the held-out test set. Complex pooling strategies with more learnable parameters showed signs of overfitting.

Pooling Strategy	Test F1
Mean Pooling	**0.75**
Weighted Layer Pooling + Attention + Gated Fusion	0.70
Weighted Layer Pooling + Attention	0.71

Table 3: Pooling strategy comparison on development set (2 samples). All methods achieved perfect dev scores, but mean pooling performed best on test.

5.2 Analysis: Why Mean Pooling Works Best

We hypothesize that simple mean pooling outperformed complex aggregation strategies for several reasons:

1. Limited Training Data: With only 5,298 training samples, sophisticated pooling mechanisms like weighted layer pooling (which learns to weigh 20+ layer weights) and multi-head attention pooling (with learned query vectors and projection matrices) introduce many additional parameters that require substantial data to train effectively.

2. Pretrained Model Quality: The E5-large model already produces high-quality token representations. Mean pooling preserves these representations without introducing additional learned transformations that may degrade under limited supervision.

3. Regularization: Mean pooling acts as implicit regularization by not adding learnable parameters to the pooling stage. The classification signal must flow through the fixed aggregation, preventing the model from overfitting through complex pooling patterns.

4. Distributional Robustness: Mean pooling treats all tokens equally, which may be beneficial when the discriminative features are distributed throughout the text rather than concentrated in specific positions.

5.3 Error Analysis

Analysis of the dataset reveals that human-written texts are approximately twice as long as machine-generated texts (632 vs. 303 words on average). This length disparity could serve as a discriminative feature, but also poses challenges:

- **Truncation Effects:** With a maximum sequence length of 512 tokens, longer human texts are truncated, potentially losing discriminative information.

- **Length Bias:** The model may partially rely on length as a proxy feature, which could reduce robustness to length-controlled adversarial examples.

Figure 1 shows the word count distribution by class, illustrating the clear separation between human and machine-generated texts.

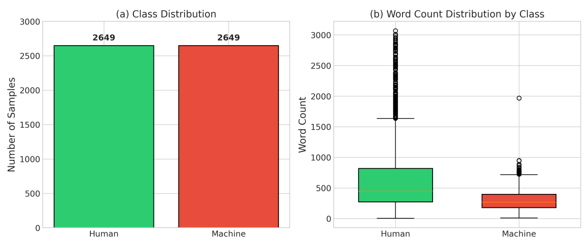

Figure 1: Dataset analysis: (a) Balanced class distribution with 2,649 samples per class. (b) Word count distribution showing human texts are significantly longer.

6 Conclusion

We presented the our system for the AbjadEval Arabic AI-generated text shared-task, achieving an F1 score of 0.75 using the multilingual E5-large encoder with mean pooling. Our key finding is that simple mean pooling outperforms sophisticated aggregation strategies like weighted layer pooling and multi-head attention pooling when training data is limited.

Limitations: Our system was trained only on the provided competition data without external datasets.

Future Work: Investigating: (1) adding more data for training, (2) longer context windows to capture full document content, (3) ensemble methods combining multiple pooling strategies, and (4) the relationship between training data size and optimal pooling complexity.

References

Shadi Abudalfa, Saad Ezzini, Ahmed Abdelali, Hamza Alami, Abdessamad Benlahbib, Salmane Chafik, Mo El-Haj, Abdelkader El Mahdaouy, Mustafa Jarrar, Salima Lamsiyah, and Hamzah Luqman. 2025. The AraGenEval shared task on Arabic authorship style transfer and AI generated text detection. In *Proceedings of The Third Arabic Natural Language Processing Conference at EMNLP 2025*, Suzhou, China. Association for Computational Linguistics.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2021. Arabert: Transformer-based model for arabic language understanding. *Preprint*, arXiv:2003.00104.

Saad Ezzini, Irfan Ahmed, Salmane Chafik, Shadi Abudalfa, Mo El-Haj, Ahmed Abdelali, Mustafa Jarrar, Nadir Durrani, Hassan Sajjad, and Farah Adeeba. 2026. Abjadgeneval: Abjad ai generated text detection shared task for languages using arabic script at abjadnlp 2026. In *Proceedings of the 2nd Workshop on NLP for Languages Using Arabic Script (AbjadNLP 2026), co-located with the 19th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2026)*, Rabat, Morocco.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M. Rush. 2019. Gltr: Statistical detection and visualization of generated text. *Preprint*, arXiv:1906.04043.

Go Inoue, Bashar Alhafni, Nurpeiis Baimukan, Houda Bouamor, and Nizar Habash. 2021. The interplay of variant, size, and task type in Arabic pre-trained language models. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, Kyiv, Ukraine (Online). Association for Computational Linguistics.

Hiroshi Inoue. 2020. Multi-sample dropout for accelerated training and better generalization. *Preprint*, arXiv:1905.09788.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2024. A watermark for large language models. *Preprint*, arXiv:2301.10226.

Salima Lamsiyah, Saad Ezzini, Abdelkader El Mahdaouy, Hamza Alami, Abdessamad Benlahbib, Samir El amrany, Salmane Chafik, and Hicham Hammouchi. 2025. M-DAIGT: A Shared Task on Multi-Domain Detection of AI-Generated Text. In *Proceedings of the Shared Task on Multi-Domain Detection of AI-Generated Text*, pages 1–9, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2018. Focal loss for dense object detection. *Preprint*, arXiv:1708.02002.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. *Preprint*, arXiv:2301.11305.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. *Preprint*, arXiv:1706.03762.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024. Text embeddings by weakly-supervised contrastive pre-training. *Preprint*, arXiv:2212.03533.