# CUET-NLP_Team_SS306 at AraGenEval Shared Task: A Transformer-based Framework for Detecting AI-Generated Arabic Text

**Sowrav Nath[*], Shadman Saleh[*], Kawsar Ahmed and Mohammed Moshiul Hoque**
Department of Computer Science and Engineering
Chittagong University of Engineering and Technology
{u2004006, u2004030, u1804017}@student.cuet.ac.bd
moshiul_240@cuet.ac.bd

## Abstract

With the rapid emergence of large language models (LLMs), AI-generated content has increased, presenting new opportunities and significant risks. Detecting such content is crucial, yet while research in high-resource languages like English has advanced, work in low-resource languages, such as Arabic, remains limited. To help fill this gap, the AraGenEval 2025 workshop organized a shared task on AI-generated Text Detection in Arabic. We participated in Task 3, where we evaluated several transformer-based models, including AraBERT, RoBERTa, AraRoBERTa, mBERT, and marBERT, both with and without chunking of input sequences during training. The experimental results show that applying chunking prior to training improves the performance of transformers. Among the evaluated models by the system testset, AraBERT with chunking achieved the highest F1 score (0.67), outperforming the others. Based on these results, our team ranked 12th in Shared Task 3.

## 1 Introduction

The rise of large language models (LLMs) has transformed text production, enabling rapid generation of coherent, human-like content. This evolution presents opportunities in creative writing, software engineering, and customer support, but also introduces risks to the integrity of educational assessment. Additionally, LLMs can enhance the sophistication and accessibility of social engineering attacks in online communication, leading to more convincing scams and the dissemination of misinformation. Reliable detection of AI-generated text is essential for maintaining trust and authenticity. While advances have occurred for languages such as English, Arabic remains challenging due to its characteristics, including root-and-pattern word construction, inflectional complexity, diverse dialects, and diacritics. Consequently, systems developed for high-resource languages frequently underperform when handling Arabic.

This work addresses these critical gaps, motivated by the need for reliable AI-generated text detection tools tailored to Arabic. We evaluate various transformer-based models for this purpose as part of the **AraGenEval 2025** shared task (Abudalfa et al., 2025). We investigate transformer-based models, including AraBERT(Antoun et al., 2020), RoBERTa (Liu et al., 2019), mBERT (Devlin et al., 2019) etc, both with and without chunking of input sequences. This work aims to provide insights into the strengths of current techniques and highlight the specific challenges of detecting AI-generated text in Arabic. The key contributions in this work are as follows:

- Evaluated multiple transformer-based models for detecting AI-generated text in Arabic.
- Introduced a chunking and confidence base aggregation approach with transformers to enhance detection performance.

## 2 Background

While most work in detecting machine-generated text has been conducted in high-resource languages (HRLs), such as English, some efforts have begun in low-resource languages (LRLs), including Arabic. Prova, 2024 made significant efforts to detect AI-generated text using BERT (Devlin et al., 2019), XGB (Chen and Guestrin, 2016), and SVM techniques. BERT models performed the best in the task, achieving an F1 score of 0.93. However, the research focused on English. Recent work by (Zhang et al., 2024) proposes a novel approach to distinguish between human and AI text. They integrated traditional TF-IDF (Takenobu, 1994) strategies with machine learning algorithms like Bayesian classifiers, Stochastic Gradient Descent

[*]Authors contributed equally to this work.

(SGD), and Categorical Gradient Boosting (CatBoost) (Prokhorenkova et al., 2019). Their methods reached an impressive ROC-AUC score of 0.975 on English text. In another study (Sadasivan et al., 2025), several types of detectors were assessed, including watermarking, neural network-based detectors, zero-shot detectors, and retrieval-based detectors. They found that AI detectors can be fooled by recursive paraphrasing, meaning the text is repeatedly reworded to evade detection. One major issue with Arabic language detection is handling diacritics, which are marks used in Arabic script to indicate pronunciation. Recent work by (Alshammari and Elleithy, 2024) focused on this challenge, comparing transformer-based models such as AraELECTRA (Antoun et al., 2021), AraBERT (Antoun et al., 2020), XLM-R (Conneau et al., 2020), and mBERT (Devlin et al., 2019). They showed that AI-detection systems struggle with Arabic text that includes diacritics and often misclassify human-written text as AI-generated.

Similar challenges exist for other LRLs. For example, a study on AI-generated review classification in Malayalam (Hasan et al., 2025) used LLMs to identify AI-generated reviews. The Gemma-2B model achieved an F1-score of 0.89. This demonstrates the potential of LLMs in detecting AI-generated content in underrepresented languages. With these findings in mind, this work employed preprocessing steps in which diacritics were removed and variants of Arabic letters were normalized. Subsequently, transformer-based techniques were applied to detect AI-generated text. In contrast to previous studies that primarily focused on HRLs or the role of diacritics in Arabic, this work utilizes chunking of input sequence before training and confidence based aggregation in output with transformer-based models to enhance long-context representation in Arabic AI-text detection.

## 3 Dataset and Task Description

The shared task[1], ARATECT: Arabic AI-Generated Text Detection, was part of the AraGenEval (Abudalfa et al., 2025) challenge. It focuses on distinguishing between human-written and AI-generated Arabic text. The ARATECT dataset comprises two primary sources. First, human-written texts were collected from reputable Arabic news sites and verified literary sources. Second, AI-generated texts were produced using Arabic-compatible large language models (e.g., GPT-4, Mistral, LLaMA) through diverse prompting strategies. Participants received a labeled training set of Arabic text samples with binary labels (human or machine). They also received an unlabeled test set for evaluation. The training set contains 4,798 samples (2,399 per class), and the test set includes 500 unlabeled samples, as shown in Table 1. The task was hosted on Codabench [2]. It aimed to advance Arabic AI-generated content detection.

| Set | Class | $S_C$ | $A_W$ | Min | Max | $T_S$ |
|---|---|---|---|---|---|---|
| Train | Human | 2399 | 657 | 1 | 3068 | 54839 |
| | Machine | 2399 | 314 | 9 | 1969 | 37768 |
| Test | All | 500 | 230 | 12 | 1589 | 7772 |

Table 1: ARATECT dataset statistics. $S_C$: sample count, $A_W$: average words per sample, Min/Max: minimum and maximum words per sample, and $T_S$: total sentences.

## 4 System Overview

Several transformer models are implemented with and without the chunking of input sequence before training and investigated to address the tasks. Figure 1 outlines the methodology.

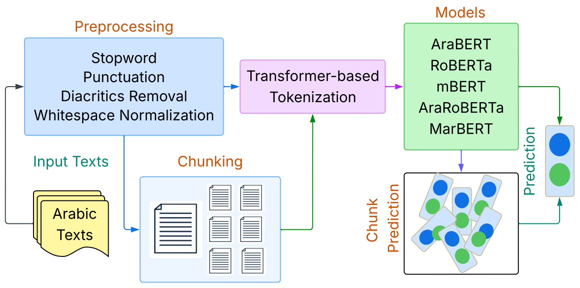

Figure 1: Schematic process of Arabic AI-generated content detection.

### 4.1 Data Preprocessing

Several preprocessing steps were applied to prepare the dataset for model training. For the training data, each sample was made using only the content field. For the test data, the title and content fields were concatenated. Subsequent preprocessing involved removing diacritics and normalizing variant Arabic letters. Repeated characters were eliminated using regular expressions. In addition, non-essential punctuation and special characters were removed. Excessive whitespace was normalized. Finally, labels were mapped to binary values in the training dataset.

[1]https://ezzini.github.io/AraGenEval/

[2]https://www.codabench.org/competitions/9120/

## 4.2 Transformer-based Models

Transformer-based models were used for this task because they efficiently process large-scale contextual information, making them well-suited for multilingual text classification. Several pre-trained transformer models from Hugging Face, including RoBERTa (Liu et al., 2019) and AraBERT (Antoun et al., 2020), mBERT (Devlin et al., 2019), ara-RoBERTa (Liu et al., 2019), MarBERT (Abdul-Mageed et al., 2020) were evaluated. Before passing data through the transformers, preprocessing and tokenization were done using each model's respective tokenizer. Inputs were truncated or padded to a maximum sequence length of 512. Since many input texts exceeded this maximum length, we applied a chunking strategy. Specifically, long texts were split into overlapping chunks of 400 with an overlap of 50 to preserve contextual continuity across chunks. Each chunk was independently processed through the model to obtain a confidence score. To aggregate predictions, we grouped chunks based on their original document IDs and computed the mean confidence score across all chunks. The final classification label was then derived from this aggregated score. This averaging approach ensures that information from all parts of the input sequence is considered, rather than being biased toward the first 512 tokens, thereby making the model more robust to long and information-dense texts. A formal description of the chunking and aggregation method is provided in Appendix A while Appendix A.5 reports rationale behind the choice of chunk size of 400 with overlap of 50.

| Parameter | Value |
|---|---|
| Batch Size | 16 |
| Epochs | 5 |
| Weight Decay | 0.001 |
| Learning Rate | 2e-5 |

Table 2: Hyperparameter configuration for the transformer-based approach.

Each model was fine-tuned for the binary classification task, with hyperparameters optimized to enhance performance (Table 2). This chunking and aggregation mechanism was particularly effective in improving performance, as it allowed the models to capture richer semantic information from long documents while mitigating the loss of important context.

## 5 Results

Transformer-based models were evaluated to assess their effectiveness in detecting Arabic AI-generated content, both on the system test set (as submitted to CodaBench [3]) and on a custom test set derived from the training data. Table 3 presents each transformer model's performance with and without chunking, reporting Precision (P), Recall (R), F1-score, and performance across short, medium, and long texts. The first two rows correspond to the system test set, while the last five rows show results on the custom test set, providing a more comprehensive analysis of model behavior.

The AraBERT achieved an F1-score of 0.62 without chunking, improving to 0.67 with chunking (+0.05). RoBERTa also benefited slightly, increasing from 0.58 to 0.61. These results indicate that chunking enhances model performance even on general sequences by better handling longer inputs.

| **Transformer** | **Approach** | Precision | Recall | F1-score | Short | Mid | Long |
|---|---|---|---|---|---|---|---|
| AraBERT (System Testset) | w/o Chunk | 0.47 | 0.89 | 0.62 | - | - | - |
| | + Chunk | 0.51 | 0.97 | 0.67 | - | - | - |
| | $\Delta$ | +0.04 | +0.08 | +0.05 | - | - | - |
| RoBERTa (System Testset) | w/o Chunk | 0.53 | 0.64 | 0.58 | - | - | - |
| | + Chunk | 0.47 | 0.87 | 0.61 | - | - | - |
| | $\Delta$ | +0.06 | +0.23 | +0.03 | - | - | - |
| AraBERT | w/o Chunk | 0.82 | 0.76 | 0.79 | 0.74 | 0.80 | 0.73 |
| | + Chunk | 0.88 | 0.87 | 0.87 | 0.89 | 0.90 | 0.83 |
| | $\Delta$ | +0.06 | +0.11 | +0.08 | +0.15 | +0.10 | +0.10 |
| RoBERTa | w/o Chunk | 0.62 | 0.54 | 0.58 | 0.79 | 0.78 | 0.42 |
| | + Chunk | 0.78 | 0.70 | 0.73 | 0.76 | 0.80 | 0.84 |
| | $\Delta$ | +0.16 | +0.16 | +0.15 | -0.03 | +0.02 | +0.42 |
| mBERT | w/o Chunk | 0.84 | 0.80 | 0.81 | 0.95 | 0.87 | 0.64 |
| | + Chunk | 0.77 | 0.50 | 0.60 | 0.37 | 0.46 | 0.76 |
| | $\Delta$ | -0.07 | -0.30 | -0.21 | -0.58 | -0.41 | +0.12 |
| Ara-RoBERTa | w/o Chunk | 0.23 | 0.50 | 0.31 | 0.64 | 0.46 | 0.12 |
| | + Chunk | 0.27 | 0.52 | 0.35 | 0.44 | 0.53 | 0.78 |
| | $\Delta$ | +0.04 | +0.02 | +0.04 | -0.20 | +0.07 | +0.66 |
| MARBERT | w/o Chunk | 0.83 | 0.78 | 0.80 | 0.87 | 0.79 | 0.41 |
| | + Chunk | 0.88 | 0.86 | 0.87 | 0.92 | 0.86 | 0.69 |
| | $\Delta$ | +0.05 | +0.08 | +0.07 | +0.05 | +0.07 | +0.28 |

Table 3: Comparison of transformer models with and without chunking on system and custom test set. $\Delta$ indicates the performance gain from chunking. Short, Mid, and Long are the performance on texts less than 512, 512 to 1024, and greater than 1024, respectively.

Since gold labels for the system test set were not disclosed, models were further evaluated on the custom test set to analyze behavior in detail, including performance by input length. Chunking produced more substantial improvements on this set: AraBERT's F1 increased from 0.79 to 0.87, with gains across short (+0.15), medium (+0.10), and long texts (+0.10), showing better context capture in sequences of varying lengths. RoBERTa gained +0.15 overall, with the largest improvement on long texts (+0.42), while MARBERT improved

[3]`https://www.codabench.org/competitions/9120/#/results-tab`

across all lengths (+0.07 overall, +0.28 on long texts), reflecting strong Arabic-specific pretraining. In contrast, mBERT decreased on short (-0.58) and medium (-0.41) texts but improved slightly on long sequences (+0.12), suggesting multilingual pretraining is less effective on shorter Arabic texts in chunked form. Ara-RoBERTa, though generally weaker, benefited notably on long texts (+0.66 F1), highlighting chunking's advantage for extended sequences.

Overall, chunking consistently improves AraBERT, RoBERTa, and MARBERT, with AraBERT (Chunk) achieving the highest F1 of 0.87. Gains are particularly pronounced for long texts (Appendix B.1), emphasizing that chunking effectively preserves full context in extended Arabic input. Models with language-specific pretraining, such as AraBERT and MARBERT, provide the most robust and balanced performance across all sequence lengths.

## 6 Error Analysis

Figure 2 shows the quantitative error analysis of the AraBERT model with chunking.

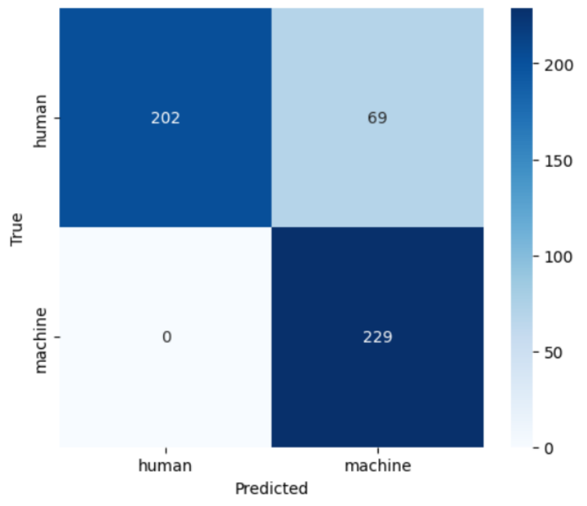

Figure 2: Confusion matrix of AraBERT with chunking

Since gold labels for the final test set were not disclosed, we evaluated our models on a custom test set alongside the system test set. The confusion matrices (Fig. 2) show that the chunked approach correctly classified 431 out of 500 texts, improving human text predictions by 60 compared to the non-chunked approach, though 69 human texts were still misclassified as machine-generated. This demonstrates how chunking helps the model capture clearer context within shorter segments (Appendix B.1). These gains are also reflected in Table 3, where most models show positive Δ values. Errors persist in long texts, where relations across distant chunks are harder to preserve. Additionally, human-written texts can be subtly altered using paraphrasing or grammar correction tools, making them resemble AI-generated outputs and further challenging detection. Appendix B provides qualitative error analysis for AraBERT, while Appendix B.1 reports performance by text length.

## 7 Conclusion

This work explored various transformer-based models for detecting AI-generated text in Arabic. Evaluation results showed that Arabic-specific BERT models with chunking, such as AraBERT and MARBERT, consistently outperformed other models. Chunking proved particularly effective for longer sequences, improving performance across short, medium, and long texts by better capturing contextual information. Future work could explore hierarchical modeling, memory-augmented transformers, and improved chunking with overlap or retrieval-based aggregation for transformer based approach, as well as integrating modern LLMs with contextualized embeddings or multilingual and Arabic-dialect-aware pretraining to further enhance detection robustness and adaptability across diverse text varieties.

## Limitations

The current study on AI-generated text detection has several limitations. A few critical issues are: (i) The dataset used was relatively small, and it is unclear whether paraphrasing techniques were applied to obscure AI-generated content or if adversarial modifications were present, which may limit the model's ability to generalize and affect its reliability. (ii) We did not explore the use of advanced large language models (LLMs) or transformer architectures like Longformer that are designed for longer contexts, leaving potential performance gains from state-of-the-art techniques unexplored. (iii) While our chunking strategy was motivated by the need to fit longer texts into the 512-token context window and did improve model performance, more sophisticated chunking and aggregation methods could be investigated to better capture context and further enhance model effectiveness.

## References

Muhammad Abdul-Mageed, AbdelRahim Elmadany, and El Moatez Billah Nagoudi. 2020. Arbert & marbert: Deep bidirectional transformers for arabic. *arXiv preprint arXiv:2101.01785*.

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 (ArabicNLP 2025)*, Suzhou, China. Association for Computational Linguistics.

Hamed Alshammari and Khaled Elleithy. 2024. Toward robust arabic ai-generated text detection: Tackling diacritics challenges. *Information*, 15(7).

Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. Arabert: Transformer-based model for arabic language understanding. *arXiv preprint arXiv:2003.00104*.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2021. Araelectra: Pre-training text discriminators for arabic language understanding. *Preprint*, arXiv:2012.15516.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794. ACM.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. *Preprint*, arXiv:1911.02116.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Md Zahid Hasan, Safiul Alam Sarker, MD Musa Kalimullah Ratul, Kawsar Ahmed, and Mohammed Moshiul Hoque. 2025. Cuet_nlp_finiteinfinity@ dravidianlangtech 2025: Exploring large language models for ai-generated product review classification in malayalam. In *Proceedings of the Fifth Workshop on Speech, Vision, and Language Technologies for Dravidian Languages*, pages 599–604.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Preprint*, arXiv:1907.11692.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2019. Catboost: unbiased boosting with categorical features. *Preprint*, arXiv:1706.09516.

Nuzhat Prova. 2024. Detecting ai generated text based on nlp and machine learning approaches. *Preprint*, arXiv:2404.10032.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2025. Can ai-generated text be reliably detected? *Preprint*, arXiv:2303.11156.

Tokunaga Takenobu. 1994. Text categorization based on weighted inverse document frequency. *Information Processing Society of Japan, SIGNL*, 94(100):33–40.

Ye Zhang, Qian Leng, Mengran Zhu, Rui Ding, Yue Wu, Jintong Song, and Yulu Gong. 2024. Enhancing text authenticity: A novel hybrid approach for ai-generated text detection. *Preprint*, arXiv:2406.06558.

## A Mathematical Intuition of Chunking and Aggregation

Let the input sequence be denoted as

$$X = (x_1, x_2, \ldots, x_L),$$

where $L$ is the sequence length and may exceed the maximum input size (512 tokens) allowed by transformer models.

### A.1 Chunking Formulation

We split $X$ into overlapping chunks of length $k = 400$ tokens with an overlap of $o = 50$ tokens. The $j$-th chunk is defined as:

$$C_j = (x_{s_j}, x_{s_j+1}, \ldots, x_{s_j+k-1}), \quad j = 1, \ldots, N,$$

where the starting index is

$$s_j = (j-1) \times (k-o) + 1,$$

and the total number of chunks is

$$N = \left\lceil \frac{L-o}{k-o} \right\rceil.$$

### A.2 Model Predictions

Each chunk $C_j$ is passed through the fine-tuned transformer model $f_\theta$, which outputs a confidence score:

$$p_j = f_\theta(C_j) \in [0, 1],$$

representing the probability that the text is AI-generated.

### A.3 Aggregation Mechanism

Since a document is split into multiple chunks, we aggregate chunk-level predictions into a document-level score. We compute the mean confidence score:

$$\hat{p} = \frac{1}{N} \sum_{j=1}^{N} p_j.$$

The final label is then derived using a threshold $\tau$ (typically $\tau = 0.5$):

$$\hat{y} = \begin{cases} 1, & \text{if } \hat{p} \geq \tau, \\ 0, & \text{otherwise.} \end{cases}$$

### A.4 Intuition

- **Chunking:** Ensures that the model processes inputs within the 512-token limit while retaining context through overlap.
- **Overlap:** The overlap $o = 50$ provides contextual continuity between adjacent chunks, mitigating boundary information loss.
- **Aggregation:** Mean aggregation smooths noisy predictions and approximates a document-level probability by considering evidence from all chunks, making the model more robust on long texts.

### A.5 Choice of Chunk Size

We chose a chunk size of 400 tokens with a 50-token overlap to stay within the model's limits while keeping context intact. Since most transformer models cap at 512 tokens, using 400 leaves enough buffer for [CLS], [SEP], and extra subword splits that Arabic tokenization often produces. Going right up to 512 is risky because any expansion can cause truncation. The overlap of about 50 tokens ( 12%) helps avoid cutting sentences in half at chunk boundaries, so important context isn't lost between chunks. This setup gave us a good trade-off: reliable coverage of long documents, preserved continuity, and faster processing compared to always maxing out at 512.

## B Qualitative Analysis

Table B1 presents representative examples of model predictions. In some cases, the model misclassified the text, which can be attributed to several factors. First, certain human-written texts exhibit stylistic or structural patterns that closely resemble AI-generated content, making them difficult to distinguish. Second, the training dataset may lack sufficient diversity across topics, writing styles, and dialects, limiting the model's ability to generalize to unseen text variations. Third, while chunking helps manage long sequences, it can lead to partial context loss across chunks, causing the model to miss subtle cues indicative of human or AI authorship. These factors collectively contribute to the observed misclassifications and highlight the challenges of detecting AI-generated Arabic text in realistic, heterogeneous datasets.

### B.1 Performance by Text Length

Figure B1 shows the performance of different transformer models across three text lengths: Short (top), Mid (middle), and Long (bottom), comparing models with and without chunking. Solid lines indicate performance with chunking, while dashed lines indicate performance without chunking. For

| Text Sample | Actual | Predicted |
|---|---|---|
| في عصرنا الرقمي الحالي، أصبحت الصور جزءًا لا يتجزأ من حياتنا اليومية، سواء كا... In our current digital age, images have become an integral part of our daily lives, whether as… | Machine | Machine |
| في تاريخ العراق الحديث، هناك لحظات فارقة شكلت وجدان الشعب وأعادت رسم ملامح ال... In modern Iraqi history, there are pivotal moments that shaped the consciousness of the people and redrew the contours of… | Machine | Machine |
| عبد العزيز أبو بكر- كيب تاون عادة ما يكون ظهر يوم الخميس في منطقة سكوتسدين على... Abdulaziz Abu Bakr – Cape Town usually appeared on Thursday in the Scottsdene area on… | Human | Machine |
| ثمَّة زوايا عديدة لتقييم نتائج الانتخابات المحلية التركية التي هُزم فيها حزب ... There are many angles from which to evaluate the results of the Turkish local elections in which the party was defeated… | Human | Human |

Table B1: Sample text predictions from the evaluated models.

short and mid-length texts, most transformers perform well even without chunking, with slight improvements observed for AraBERT, RoBERTa, and MarBERT, and a noticeable improvement of Ara-RoBERTa in short texts. For long texts, chunking provides substantial improvements, especially for AraBERT, RoBERTa, and ara-RoBERTa, while mBERT without chunking performs poorly. Overall, the figure illustrates that chunking consistently enhances transformer performance, particularly for longer sequences.

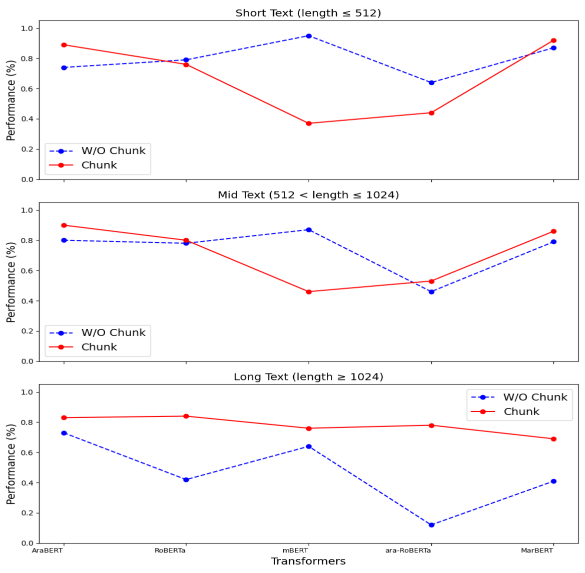

Figure B1: Transformer Performance Across Text Lengths (Chunk vs W/O Chunk)).