big data and cognitive computing

Article

AI-Generated Text Detector for Arabic Language Using Encoder-Based Transformer Architecture

Hamed Alshammari [1], Ahmed El-Sayed [2] and Khaled Elleithy [1,*]

[1] Department of Computer Science and Engineering, University of Bridgeport, Bridgeport, CT 06604, USA; halsh@my.bridgeport.edu
[2] Department of Electrical and Computer Engineering, University of Bridgeport, Bridgeport, CT 06604, USA; aelsayed@bridgeport.edu
* Correspondence: elleithy@bridgeport.edu

Abstract: The effectiveness of existing AI detectors is notably hampered when processing Arabic texts. This study introduces a novel AI text classifier designed specifically for Arabic, tackling the distinct challenges inherent in processing this language. A particular focus is placed on accurately recognizing human-written texts (HWTs), an area where existing AI detectors have demonstrated significant limitations. To achieve this goal, this paper utilized and fine-tuned two Transformer-based models, AraELECTRA and XLM-R, by training them on two distinct datasets: a large dataset comprising 43,958 examples and a custom dataset with 3078 examples that contain HWT and AI-generated texts (AIGTs) from various sources, including ChatGPT 3.5, ChatGPT-4, and BARD. The proposed architecture is adaptable to any language, but this work evaluates these models' efficiency in recognizing HWTs versus AIGTs in Arabic as an example of Semitic languages. The performance of the proposed models has been compared against the two prominent existing AI detectors, GPTZero and OpenAI Text Classifier, particularly on the AIRABIC benchmark dataset. The results reveal that the proposed classifiers outperform both GPTZero and OpenAI Text Classifier with 81% accuracy compared to 63% and 50% for GPTZero and OpenAI Text Classifier, respectively. Furthermore, integrating a Dediacritization Layer prior to the classification model demonstrated a significant enhancement in the detection accuracy of both HWTs and AIGTs. This Dediacritization step markedly improved the classification accuracy, elevating it from 81% to as high as 99% and, in some instances, even achieving 100%.

Keywords: AI detector; transformer; BARD; AI-generated; AI detection; AI content; synthetic texts; ChatGPT; NLP; AIRABIC

check for updates

Citation: Alshammari, H.; El-Sayed, A.; Elleithy, K. AI-Generated Text Detector for Arabic Language Using Encoder-Based Transformer Architecture. *Big Data Cogn. Comput.* **2024**, *8*, 32. https://doi.org/10.3390/bdcc8030032

Academic Editor: Min Chen

Received: 4 February 2024
Revised: 11 March 2024
Accepted: 13 March 2024
Published: 18 March 2024

1. Introduction

1.1. Arabic Language Challenges

The Arabic language, a linguistic tapestry rich in history and cultural significance, is one of the most widely spoken languages globally, boasting over 420 million speakers with it as their first language [1,2]. It is the official language in more than 22 countries, underscoring its geopolitical importance [3]. Additionally, Arabic is one of the six official languages of the United Nations (UN), highlighting its essential role in international diplomacy and communication [4]. The Arabic language is one of the Semitic languages that face many Natural Language Processing (NLP) challenges, including the language's morphological complexity, a variety of dialects, a writing style from right to left, the existence or absence of diacritics, and orthographic inconsistencies. Many studies have discussed these challenges, such as [5–8]. Our focus in this paper is the challenge of the existence of diacritics. In the following section, we will address this challenge in detail.

1.2. Arabic Diacritization Marks' Background

Arabic diacritization marks, also known as Tashkeel or Harakat, play an essential role in the Arabic language. These marks are small signs or symbols (ـ ـِإ ـ ـُأ ـ ـْ ـْ ـْ) written above or below the letters to indicate the correct pronunciation, with a particular emphasis on the articulation of short vowels. The importance of these diacritization marks becomes particularly evident in the context of semantic differentiation. For example, a single Arabic word with three letters, such as 'علم', can yield many meanings, contingent upon the specific diacritics employed. Therefore, applying varying diacritics to 'علم' elucidates its semantic versatility. For instance, 'عُلِمْ' (pronounced 'ulim'), translates to 'understood (adj)'; 'عَلَمْ' (pronounced 'alam'), means a 'flag (n)'; 'عَلَّمَ' (pronounced 'allama'), means 'taught (v)'; 'عِلْمْ' (pronounced 'ilm'), means 'knowledge'; and so on. Thus, the essential role of diacritization in Arabic lies in its ability to clarify meanings and ensure clear communication, making it easier for speakers and readers to understand each other.

1.3. Challenges for AI Detectors in Processing Arabic Texts with Diacritics

This study [5] underscores that one of the primary sources of ambiguity in Arabic NLP (ANLP) systems is attributed to the absence of diacritic marks. However, their existence in human-written text (HWT) makes noise in the AI detector and the text is classified as AI-generated text (AIGT). This was analyzed in depth when GPTZero [9] and OpenAI Text Classifier [10] were evaluated against AIRABIC (https://github.com/Hamed1Hamed/AIRABIC) (accessed on 18 November 2023) [11], which is an Arabic benchmark dataset for assessing the AI detectors. The study results provide insights into the performance of GPTZero and OpenAI Text Classifier in classifying a set of 500 HWT examples. The findings indicate a significant decrease in GPTZero's classification accuracy, particularly when handling texts containing diacritics. In that context, it could correctly classify only 31 examples with diacritics, in contrast to a slightly higher accuracy with 119 examples that lacked diacritics. On the other hand, the OpenAI Text Classifier demonstrated a noticeable bias when processing Arabic text, categorizing all such texts uniformly as AIGT. This outcome suggests a potential improvement in the classifier's ability to distinguish between HWTs and AIGTs, especially in Arabic texts with their unique linguistic features.

1.4. Large Language Models

Large Language Models (LLMs) have demonstrated impressive skills in producing smooth, well-structured, and persuasive text in recent years. Earlier models, including GPT-3, introduced in 2020 [12], and PaLM, in 2022 [13], showcased the capabilities of LLMs in various NLP tasks. By the end of November 2022, ChatGPT [14], a variant of the GPT-3 model [12], emerged as a pivotal and often debated milestone in LLM evolution [15]. However, ChatGPT is not only an expansion of existing models. It has been distinctly refined and enhanced through a process involving supervised and reinforcement learning methods informed by human feedback [16,17]. Representing a groundbreaking advancement in NLP, it has become the first model to be embraced beyond NLP research, and it is adept at producing text closely resembling human writing [18]. ChatGPT's ability to mimic human writing, further enhanced in its latest version based on the GPT-4 Model, which arrived in March 2023, has been a distinguishing feature. Following ChatGPT, subsequent generative models like BARD [19] have also attracted attention for their ability to produce content that mimics human-like expressions with their ability to access the internet utilizing Google searches.

Before ChatGPT emerged, extensive academic research was conducted on the ethical risks associated with LLMs, particularly those involved in natural language generation (NLG). These investigations delved into potential societal impacts, highlighting concerns ranging from the confident distribution of inaccurate information to the creation of widespread false news and information [20,21]. With the advent of ChatGPT, concerns have increased, as this study highlighted its risks [22].

These technological advancements in generative language models (GLMs) have unintentionally opened doors to various unethical applications across multiple domains. In the education field, one of the most critical concerns is plagiarism, a concern extensively examined in the study [23]. Similarly, in the field of scientific research, there has been a notable trend of utilizing ChatGPT for crafting abstracts, a practice critically analyzed in the study [24]. The implications also extend to the medical sector, where the misuse of these technologies is explored in depth in the study [25]. Considering these instances, there is an increasing focus on identifying AIGT. This focus on detection aligns with a broader movement toward the ethical and suitable application of ChatGPT and other GLMs [26–28].

An increasing number of businesses are attempting to tackle this issue in response to the potential misuse of ChatGPT and other GLMs. The latest research study by [16] summarizes various commercial and accessible online tools available for this purpose.

However, it is noteworthy that most evaluations conducted thus far have predominantly focused on languages rooted in English or Latin-driven scripts. To the best of our knowledge, no AI detector has been trained on HWT and AIGT from ChatGPT in Arabic to the date of writing this paper. Therefore, the AIRABIC benchmark dataset reveals the lack of existing AI detectors to process and recognize Arabic HWT.

Motivated by the need for an AI detector that works well with the Arabic language, this paper introduces a novel Arabic AI classifier. While our architecture is versatile and can be adapted for datasets in various languages, we specifically targeted the Arabic language in this study. The primary objective of this research is to pioneer the development of a ChatGPT detector tailored explicitly to the Arabic language and to evaluate its effectiveness against the AIRABIC benchmark dataset rigorously. To achieve our objective, our methodology encompasses building classifiers upon two datasets. The first dataset, referred to as the large dataset, comprises 43,958 examples. The second, known as the custom dataset, contains 3078 examples. Therefore, our contribution to this paper can be summarized as follows:

- Development of Two Arabic AI Classifier Models: We trained these models on a dataset comprising 35,166 examples, validated them on 4369 examples, and tested them on another set of 4369 examples. These examples include HWTs and AIGTs derived from the ChatGPT 3.5 Turbo model.
- Evaluation Using a Custom-Crafted Dataset: The models were further evaluated on our handcrafted dataset, which includes 2466 training examples, 306 validation examples, and 306 testing examples. This dataset encompasses a diverse range of HWTs from various sources and AIGTs generated by ChatGPT 3.5, ChatGPT 4, and BARD.
- Benchmarking Against AIRABIC Dataset: We employed the AIRABIC benchmark dataset to assess our models' robustness and efficacy. This allowed us to measure our models' performance and compare it with two prominent AI detectors, GPTZero and the OpenAI Text Classifier.
- Incorporation of the Dediacritization Layer: We incorporated a Dediacritization Layer in our architecture, aiming to enhance the classification performance of the proposed models, particularly for texts with diacritics.
- We conducted a detailed comparative analysis of the used pre-trained models emphasizing their performance across various datasets and under different hyperparameter configurations, thereby offering critical insights for enhancing language-specific AI text classifiers.

The remainder of this paper is structured as follows:

➢ Section 2, Related Works, delves into the evolution of AIGT detection, focusing on developments before and after the emergence of ChatGPT. This section sets the stage for our research by highlighting the progress and remaining gaps in the field.

➢ Section 3, Methodology, details the comprehensive approach taken in our study, including data collection and the architecture of our detector. Special attention is paid to the fine-tuning process, the innovative Dediacritization Layer, and the design of our evaluation pipeline. Additionally, this section outlines the experimental protocol and hyperparameter settings, emphasizing scheduler and learning rate optimizations to underscore the rigor of our research design.

➢ Section 4, Results, offers an in-depth comparative analysis of our models against established benchmarks such as GPTZero and OpenAI Text Classifier. Through a series of tests on the AIRABIC benchmark dataset, we demonstrate the effectiveness of our fine-tuned AraELECTRA and XLM-R models. This section highlights the advancements our research contributes to the field and presents evidence of our models' superior performance and efficiency.

➢ Section 5, Discussion, provides insights gleaned from our findings, with an emphasis on the impact of dataset size and content variation. It also provides a comparative performance analysis of the XLM-R versus AraELECTRA models. The significant role of the Dediacritization Layer in enhancing classifier performance is examined in detail, showcasing the methodological innovations at the heart of our study.

➢ Section 6, Conclusion and Future Work, summarizes the contributions of our research, outlines the potential for future work, and suggests directions for further research to build on the strong foundation laid by our study, aiming to inspire continued advancements in the field of Arabic AI detection.

2. Related Works

2.1. Detection of AIGT Prior to ChatGPT

In mid-2019, a significant development in this field was the introduction of the GROVER model [29], capable of generating and detecting fake news. GROVER had access to 5000 of its own generated articles and unlimited real news content, boasting a 92% accuracy rate in detection, surpassing other deep pre-trained models such as [30,31]. In June 2019, the GLTR (Giant Language Model Test Room) tool [32] was released. As an open-source resource, GLTR uses various baseline statistical methods to detect texts generated by models like GPT-2, identifying generation artifacts from different sampling methods used in language models. Later in the same year, OpenAI rolled out a specialized GPT-2 detector [33], fine-tuning the Roberta model [34].

In 2020, a novel model was developed to generate and detect fake online reviews [35]. This innovative approach combined GPT-2's review generation capabilities with a fine-tuned BERT model [31] as a classifier for the detection phase. In the same year, the authors of [36] explored the challenge of distinguishing between texts written by humans and various neural network-based language models. They focused on three specific authorship attribution problems: determining if two texts were generated by the same method, identifying whether a human or a machine wrote a text, and identifying the specific neural method behind a given text. Their study involved empirical experiments with texts written by humans and those generated by eight different models, including CTRL [37], GPT [30], GPT-2, GROVER, XLM [38], XLNET [39], PPLM [40], and FAIR [41]. The study revealed that while most text generators still produce content distinguishable from HWT, some models, like GPT2, GROVER, and FAIR, create higher-quality outputs that more frequently confuse machine classifiers.

In 2021, a detector was developed to identify deepfake tweets as part of the 'TweepFake' project [42,43]. This work involved creating the first real dataset of deepfake tweets posted on Twitter. The research team collected and analyzed tweets from 23 bots that replicated 17 real user accounts employing a range of content creation methods, including GPT-2, Markov Chains, RNN, and LSTM. The study used and fine-tuned several pre-trained language models for detection, such as BERT, DistilBERT [30], RoBERTa, and XLNet [26]. They encompassed an evaluation of 13 advanced deepfake text detection methods to highlight the specific challenges presented by TweepFake and to establish a reliable foundation of techniques for detection. Another study in 2021 [44] proposed a new detection method using text similarity with round-trip translation (TSRT). This approach involves translating text into another language and returning it to the original language. They found that TSRT achieved an 86.9% accuracy rate in detecting texts translated by unfamiliar translators, surpassing both traditional detectors (with a 77.9% accuracy) and human discernment capabilities (53.3%). Similar work was conducted using BERT for detecting Arabic GPT2 tweets [43]

In the same year, 2021, AraGPT-2 [45], an advanced language generation model for Arabic, demonstrated a dual capability in generating and detecting synthetic text. It excels in generating high-quality synthetic text, such as news articles, that are challenging to differentiate from humans. This capability is particularly evident in its largest variant, AraGPT2-mega, which has 1.46 billion parameters. Concurrently, the authors of this paper also developed an automatic discriminator model to address the potential misuse of such synthetic text generation. This model can detect text generated by AraGPT-2 with a high accuracy rate of 98%, making it a valuable tool for distinguishing between human-written and machine-generated content in Arabic. Subsequently, the same author of AraGPT2 released the AraELECTRA model [32], which is based on an ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately) model [46]. AraELECTRA was trained on a vast Arabic corpus to identify content created by their AraGPT2 model. It is worth noting that AraELECTRA was trained on the same dataset used for training the AraGPT2 model. The effectiveness of the AraELECTRA model can be attributed to its pre-training goal of detecting token replacements (RTD), an advanced approach that represents a significant shift from the conventional Masked Language Modeling (MLM) strategy employed in models like BERT. In contrast to MLM, where specific tokens in the input text are masked and subsequently predicted, RTD innovatively replaces tokens in the sequence with alternatives generated by a secondary model. The principal task for the discriminator within the ELECTRA framework is to accurately differentiate between the original tokens and those that have been intentionally substituted. This method trains the model to identify 'real' tokens from the original text and 'fake' ones introduced by the generator. By focusing on discerning real from altered tokens rather than simply predicting masked ones, this refined approach substantially enhances the model's learning efficacy, allowing it to extract more comprehensive insights from the input data.

In 2022, the development of a neural network-based detector that combines textual information with explicit factual knowledge was introduced [47]. This is achieved through entity–relation graphs, encoded by a graph convolutional neural network, which capture interactions between various entities and relationships within the text. The model aims to detect manipulated news articles by reasoning about the facts mentioned, distinguishing them from detectors that only utilize stylometric signals. This work contributes to the broader field of fake news detection, emphasizing the importance of understanding the content of news articles and verifying their factual accuracy. Employing factual knowledge in conjunction with textual analysis represents a notable advancement in identifying texts that have been altered, which could be mistakenly considered human-written due to their preserved writing style. However, the authors concluded their experiment by stating that detecting manipulated text continues to be difficult and challenging.

2.2. Detection of AIGT after ChatGPT

Since the release of ChatGPT in November 2022, there has been a marked surge in research interest surrounding detection models, focusing predominantly on identifying text generated by this advanced multi-model chatbot. In the subsequent section, we summarize some of the latest research and models in this area until we wrote this paper. Our review mainly strictly covers academic studies, and we intentionally omit web-based non-academic tools due to the lack of clarity regarding their training methods and internal mechanisms.

One of the first released studies in 2023 that brought both datasets generated by ChatGPT and HWT and detectors trained on the same dataset in both the English and Chinese languages was conducted by the study in [48], which examines the capabilities of ChatGPT in comparison to human experts. The researchers collected a large dataset and called their dataset Human ChatGPT Comparison Corpus (HC3). It consists of nearly 40,000 questions and answers by utilizing datasets such as OpenQA [49], Reddit ELI5 [50], and ChatGPT to generate the responses for these questions, covering various domains such as finance, medicine, law, and psychology. They conducted detailed human evaluations and linguistic analyses to study the differences and similarities between human-written content and ChatGPT responses. The study led to the development of several models to detect whether content is generated by ChatGPT or written by humans, demonstrating decent performance in different scenarios. They used other methods, starting from machine learning and deep learning algorithms. The approach involved employing logistic regression, trained using GLTR features, along with a deep learning classifier based on the pre-trained Transformer model, specifically the RoBERTA model [34]. The RoBERTA model was fine-tuned to detect both single-text formats and question-and-answer-type texts. Additionally, all collected data were made open source to aid further academic research, meaning some of the studies that came after leveraged their dataset, either by adapting it to different languages or utilizing its English examples.

One of the earliest studies released in 2023 that introduced a novel watermarking framework for LLMs was conducted in the study in [51]. Their approach leverages the output log probability of LLMs at each generation step to embed a watermark, primarily through the use of a green token list. In this framework, the LLM is strategically inclined to select tokens from the green list more frequently (e.g., nine green tokens for humans), and more than that, it has a high probability of being AIGT. This method is designed to embed detectable signals in the generated text that are invisible to human readers, offering a way to responsibly manage and identify the outputs of these powerful language models. Similarly, other work conducted on watermarking [52] introduces three significant enhancements to existing watermarking methods. It enhances watermarking in LLMs by introducing statistically sound tests for false positive rate accuracy, assessing text quality through NLP benchmarks, and advancing toward scalable multi-bit watermarking systems. Another significant contribution in the field of watermarking, conducted in the study in [53], adopts a distinct approach by leveraging cryptographic principles. This innovative method, unique in its reliance on cryptography, ensures that watermarks embedded within LLMs remain undetectable unless a specific secret key is used. This cryptographically inspired technique marks a paradigm shift in watermarking, focusing on the imperceptibility of watermarks to enhance the security and authenticity verification of AIGTs.

A recent study also aimed to develop a classifier capable of identifying restaurant reviews generated by ChatGPT [54]. They utilized the Kaggle restaurant reviews dataset as a foundation and prompted ChatGPT to create multiple types of reviews, such as those for poor-quality restaurants. To further challenge the classifier, they also had ChatGPT reword existing human-written reviews, generating an adversarial dataset to test the classifier's effectiveness. They fine-tuned the uncased version of the DitlBeERT model on a dataset for training and testing only. In their efforts, they also created a classifier that relies on the perplexity score. This involved calculating the perplexity score of each entry in the training set using the GPT-2 model. This measure helps gauge the complexity or uncertainty of the text as interpreted by the model. In this manner, they attempted to establish a critical perplexity score threshold for classifying text. Texts exceeding this threshold were labeled as human (0), while those below were tagged as ChatGPT (1). The threshold was optimized to enhance the F1-score in training. The methodology was then applied to the test set for effective classification.

Another paper was released on detecting ChatGPT text, targeting both English and French [55]. The English model built upon the HC3 dataset by translating its English content into French using Google Cloud Translate API. They also included an additional tiny French out-of-domain dataset comprising 113 examples of direct French responses from ChatGPT and 116 examples from BingGPT to some of the translated HC3 questions from the testing set. Then, they subsidized their dataset with French question–answer pairs (4454 examples) from the Multilingual FAQ (MFAQ) dataset [56] and sentences from the French Treebank dataset (1235 sentences). Additionally, the dataset incorporates a small set (61 examples) of adversarial examples crafted by humans in a style similar to ChatGPT's outputs. They fine-tuned two pre-trained models, the CamemBERT [57] and CamemBERTa [58] models, for the French language and fine-tuned the RoBERTa and ELECTRA models for the English language, then used XLM-R [59] for the combined datasets that have the two languages. The study indicated that the French models effectively identified AIGT within the same domain. However, the English models showed superior performance. The XLM-RoBERTa-base model demonstrated the most robust and consistent results in detecting adversarial attack datasets for English and French texts. Nonetheless, the ability of these models to accurately detect AI-generated content dropped when applied to out-of-domain text.

3. Methodology

3.1. Data Collection

This study employed two comprehensive datasets for the training, validation, and testing phases, as delineated in Figure 1. The first one, which we called a large dataset, comprises 43,958 examples, of which 21,979 are HWTs that were obtained and preprocessed from three sources: the Arabic-SQuAD [60] dataset, the TyDiQA (TyDiQA-GoldP) dataset [61], and the MultiLingual Question Answering (MLQA) dataset [62]. Our focus in these datasets was explicitly on the textual content, excluding any aspect related to question-answering. To ensure a balanced dataset, we translated the English content of ChatGPT-generated text in the HC3 dataset [48] using the Google Translate built-in in Sheet to inspect the translation quality during the process, resulting in 20,982 ChatGPT-generated examples. This step was vital as our examination of the HC3 dataset revealed empty instances labeled 'false' (which represent ChatGPT-generated text), and some of them were redundant content, which were subsequently removed to maintain data quality and relevance. This process was integral in aligning the volume of ChatGPT texts with the HWT examples. To further equalize the datasets, additional translations from a dataset (https://github.com/amartyahatua/AI_Generated_Text_Classfier/tree/master/data) (access on 12 October 2023) came with this study [63], leading to 35,196 training examples, with 4396 each for validation and testing, encompassing both HWT and ChatGPT text.

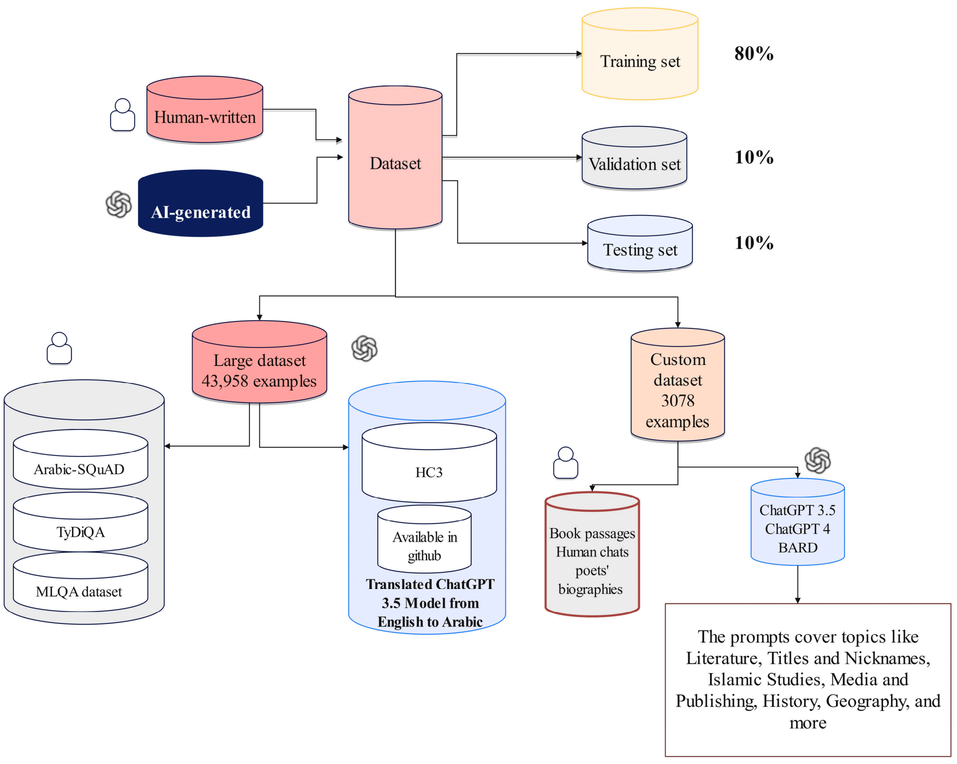


Figure 1. Dataset configuration and segregation.

The custom dataset in our study was deliberately crafted manually to exhibit a range of writing styles from various human writing styles and different leading chatbot models. For the AIGT, this dataset includes an equal number of instances (513 each) from ChatGPT 3.5, ChatGPT 4.0, and BARD. This composition results in a total of 1539 AIGT samples, offering a comprehensive representation of different LLM output styles. We ensured each prompt was sent only once and selected a single response for inclusion, especially from BARD, which often provides three drafts per response. The prompts cover topics like Literature, Titles and Nicknames, Islamic Studies, Media and Publishing, History, and Geography. We meticulously chose questions from each section for input into ChatGPT versions and BARD as detailed in Table 1. As for the HWT dataset, it also has 1539 instances, including written lectures and human text chats collected from the Tashkila dataset [64], after being preprocessed by removing diacritics. It also has passages from different books from various sources, including the Shamela Library [65] and Tashkila. We also included poets' biographies written on the Aldiwan website [66]. The specific contributions from each source are detailed in Table 2. The passages that were chosen have different typing formats that include, but are not limited to, in-text citations, bullet points, long paragraphs, and more.

Table 1. Dataset composition of AIGT showcasing prompt distribution from ChatGPT Models 3.5, 4.0, and BARD.

Custom Dataset AIGT					
ChatGPT				BARD	
Model 3.5 turpo		Model 4		-	
513 Prompts		513 Prompts		513 Prompts	
Topics					
Literature	Titles and Nicknames	Islamic Studies	Media and Publishing	History	Geography
71 prompts	29 prompts	100 prompts	75 prompts	106 prompts	132 prompts

Table 2. Dataset composition of HWT showcasing dataset structure detailing number of instances from every source.

Custom Dataset HWT		
Shamela Library	Aldiwan website	Tashkila dataset (diacritics-free)
249 samples obtained from book passages	240 samples obtained from poets' biographies	1050 samples, including book passages, human text chats, and lecture notes

We allocated 80% of the dataset to training, with the remaining 20% divided equally between validation and testing across both datasets. This distribution was deliberately chosen, with a predominant portion dedicated to training, to achieve more than mere performance evaluation on the same datasets. The primary objective was to enhance the model's robustness, ensuring its capability to generalize well across diverse datasets. By prioritizing a substantial training set, we aimed to equip the model with a deep and nuanced understanding of the underlying patterns, which is critical for its adaptability and effectiveness when faced with different data.

3.2. Detector Architecture

In developing our detector architecture, our approach was grounded in the strategic adoption of deep learning paradigms, specifically focusing on the fine-tuning of pre-trained encoder-based Transformer models, which are quintessential for their superior proficiency in language understanding. To accommodate the linguistic diversity of the large dataset, which intentionally incorporates non-Arabic scripts without translation, we selected the

XLM-R model [59], lauded for its exceptional multilingual capabilities as evidenced by its performance in related studies [48,55] and its ability to adapt over 100 languages. In tandem with the AraELECTRA model [67], one of the state-of-the-art models in Arabic built upon the ELECTRA model [46] was chosen for its state-of-the-art efficacy in Arabic language understanding, building upon the advancements of the ELECTRA architecture.

3.2.1. Fine-Tuning Process

The process begins with the raw input text, which is tokenized into a sequence of tokens. These tokens serve as the input to a pre-trained model of language understanding designed to extract contextual representations from the text (as illustrated in Figure 2). Each token is transformed into a dense vector representation, encapsulating the semantic and syntactic information learned during the pre-training phase. Upon processing the input text, the pre-trained model generates a sequence of representation vectors (Rep 1 to Rep 512). These representations are then fed into a dense (fully connected) linear layer that projects the high-dimensional features into an output space corresponding to the number of classes required for our classification task. This dense layer is important as it adapts the general language representations to the specificities of distinguishing HWT from AIGT. Following the dense layer, a SoftMax function is applied to the output logits, converting them into normalized probabilities, as shown in Equation (1). During training, these probabilities are evaluated using a cross-entropy loss function for binary classification, as shown in Equation (2), which measures the discrepancy between the predicted probability distribution and the true class labels. This loss is then used to update all network layers through backpropagation, effectively fine-tuning the entire model to improve its ability to classify texts as HWT or AIGT. During inference, the SoftMax-normalized probabilities inform the prediction, where the class with the highest probability is selected as the predicted class. This fine-tuned model, therefore, not only leverages the sophisticated linguistic understanding of the pre-trained model but also applies it specifically to distinguish between HWT and AIGT with a high degree of accuracy, as indicated by our classification results. All network layers were updated throughout the training process to ensure that the pre-trained language representation models and the additional dense layer were optimized for our specific classification problem. This comprehensive fine-tuning approach ensures that the model is well-adapted to the nuances of the task, leading to improved performance and robustness in classifying the text.

For a binary classification task such as ours, the SoftMax function can be expressed for two classes, HWT (label 1) and AIGT (label 0):

$$\sigma(z)_i = \frac{e^{z_i}}{e^{z_0} + e^{z_1}} \tag{1}$$

where $\sigma(z)_i$ represents the predicted probability for the $i-$th class given the input vector z of logits, with i corresponding to either HWT or AIGT.

The cross-entropy loss function for binary classification is defined as

$$L(y, \hat{y}) = -[y\log(\hat{y_1}) + (1-y)\log(1-\hat{y_1})] \tag{2}$$

where L is the loss and y represents the true label (1 for HW and 0 for AIG). The predicted probability $\hat{y_1}$ indicates the model's confidence that the input text is HW, and $1-\hat{y_1}$ represents the model's confidence that the input text is AIG. This function penalizes the predictions that diverge from the actual labels. For a true label of 1 (HW), the loss is driven by the term $y\log(\hat{y_1})$, encouraging the model to increase the predicted probability $\hat{y_1}$ for HWT. Conversely, for a true label of 0 (AIG), the loss is influenced by the term $(1-y)\log(1-\hat{y_1})$, which encourages the model to increase $1-\hat{y_1}$, the predicted probability for the text being AIG. By penalizing the divergence in both cases, the model is trained to make more accurate predictions throughout training.

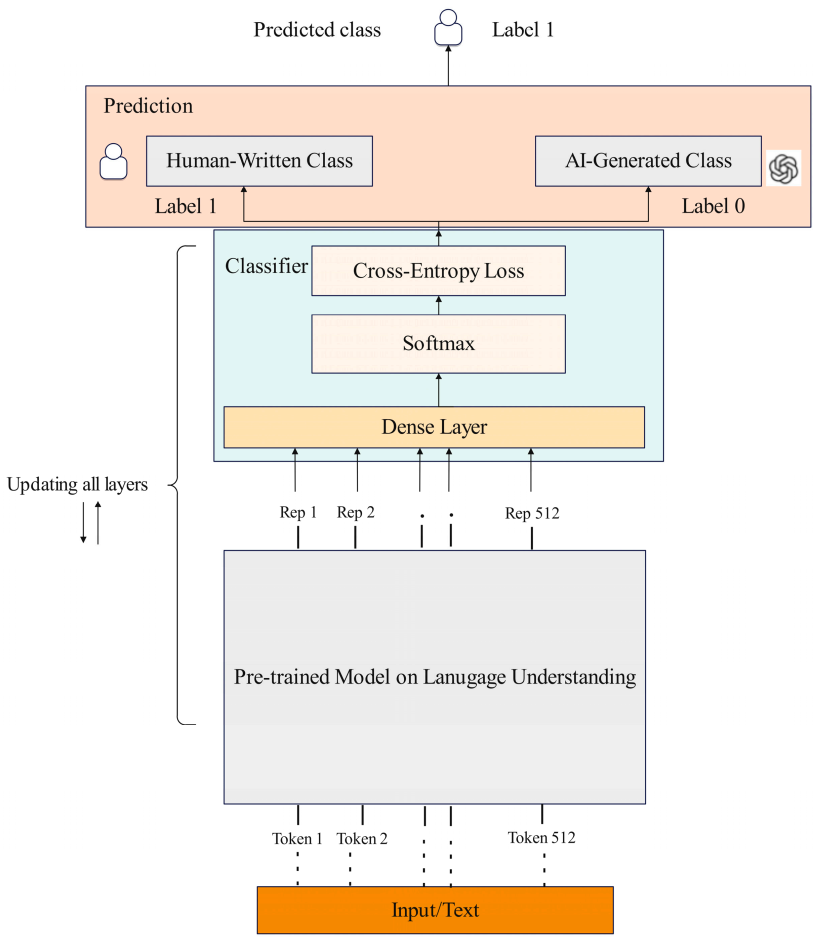


Figure 2. Architectural diagram of the fine-tuning process. This figure illustrates the end-to-end workflow for the binary classification task.

3.2.2. Dediacritization Layer

Previous investigations within this research [11] have identified the presence of diacritics in the Arabic text as a potential source of noise that may impair the efficacy of classification algorithms. To address this challenge, we proposed the integration of a Dediacritization Layer prior to the classification stage. In this study, we operationalized this by employing the Dediacritization Layer function, utilizing the preprocessing capabilities of the CAMeL ANLP tools [5] to remove diacritics from the text (with no changes to the text itself), as shown in Figure 3. This preprocessing was exclusively applied during the evaluation phase of the AIRABIC benchmark dataset to substantiate our hypothesized impact of diacritics on classifier performance. However, this dediacritization process was not utilized in the training, validation, or testing phases for models using both the custom and large datasets, as these datasets are not laden with diacritics. Figure 4 shows the code for this step.

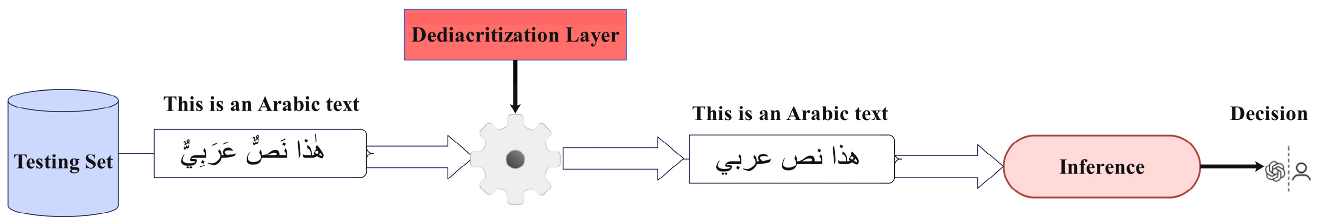
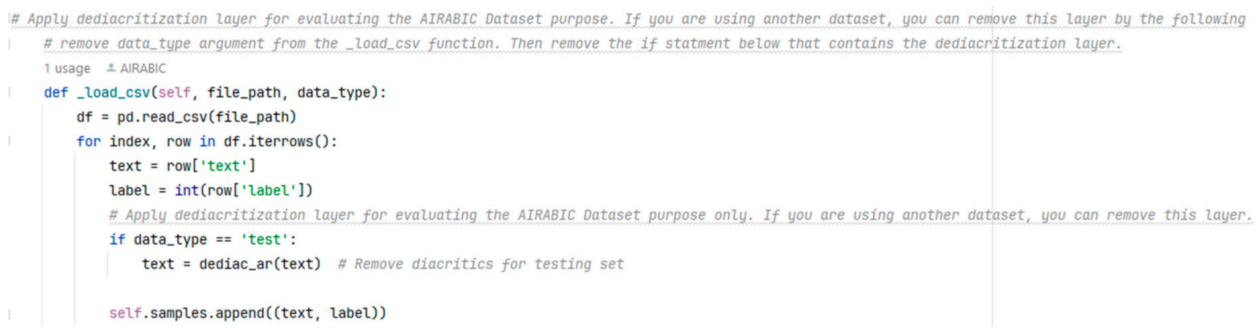


Figure 3. Procedural framework for dediacritization layer.



```
# Apply dediacritization layer for evaluating the AIRABIC Dataset purpose. If you are using another dataset, you can remove this layer by the following
    # remove data_type argument from the _load_csv function. Then remove the if statment below that contains the dediacritization layer.
    1 usage  ≗ AIRABIC
    def _load_csv(self, file_path, data_type):
        df = pd.read_csv(file_path)
        for index, row in df.iterrows():
            text = row['text']
            label = int(row['label'])
            # Apply dediacritization layer for evaluating the AIRABIC Dataset purpose only. If you are using another dataset, you can remove this layer.
            if data_type == 'test':
                text = dediac_ar(text)  # Remove diacritics for testing set

            self.samples.append((text, label))
```

Figure 4. Dediacritization layer code.

3.3. Pipeline Design

We designed our architecture with the intent of versatility and reusability across various design scenarios. Figure 5 shows that the processing architecture encompasses several phases. Depending on the dataset being assessed, the model's workflow bifurcates for the inference phase. Aside from the Dediacritization Layer, which works specifically for Arabic, our design is adaptable to any language across various datasets. Thus, our design is universally applicable and ready to be utilized with datasets in any language.

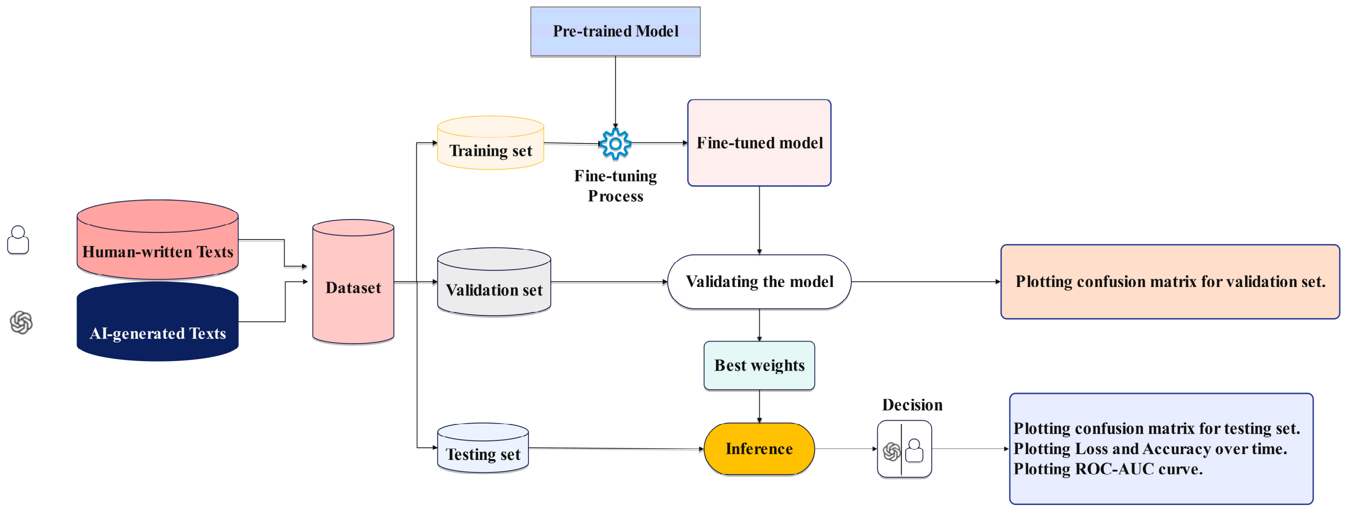


Figure 5. High-level workflow of the text classification architecture.

3.4. Experimental Evaluation Protocol

A range of methodologies were explored to ascertain the most favorable outcomes. This exploration was followed by the execution of 12 meticulously designed experiments to assess the robustness of the two models across disparate datasets, as illustrated in Figure 6.

In our case, we had two datasets for training and three distinct datasets for testing, one of which, the AIRABIC dataset, fell outside the domain of the training data. To assess the robustness and generalization capability of the two models, we conducted multiple evaluations that were carefully designed to prevent any overlap between the training datasets and to avoid transfer learning biases during testing. Thus, all models started training, validation, and testing on a specific dataset, and then the models' best weights were evaluated against the other datasets. Then, the model was finally tested against the benchmark out-of-domain dataset, AIRABIC. Since AIRABIC has two types of writing, one with diacritics and one without (after incorporating the Dediacritization Layer in the inference), every model was tested on both to be compared against the existing leading AI detectors, such as GPTZero and OpenAI Classifier.

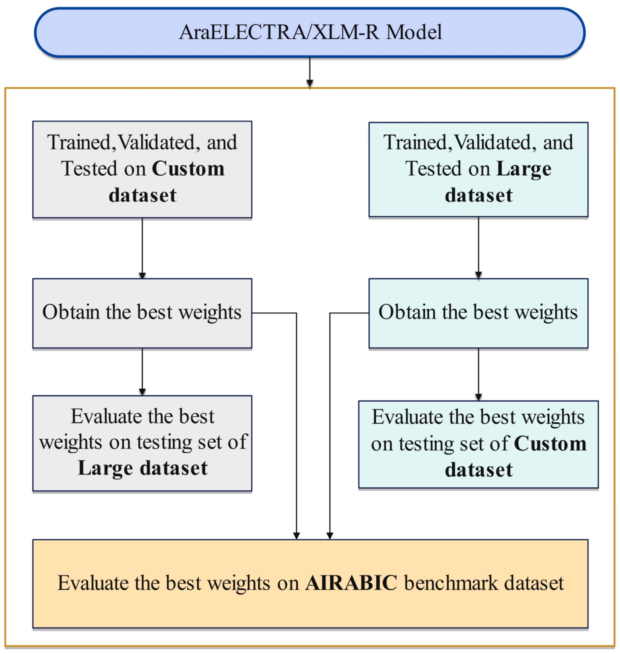

AraELECTRA/XLM-R Model
Trained, Validated, and Tested on **Custom dataset**
Trained, Validated, and Tested on **Large dataset**
Obtain the best weights
Obtain the best weights
Evaluate the best weights on testing set of **Large dataset**
Evaluate the best weights on testing set of **Custom dataset**
Evaluate the best weights on **AIRABIC** benchmark dataset

Figure 6. Cross-evaluation methodology for optimized model parameters across custom, large, and AIRABIC benchmark datasets with AraELECTRA and XLM-R models.

3.5. Hyperparameters

A comprehensive series of experiments was undertaken to optimize a suite of hyperparameters, the objective being to ascertain the most efficacious configuration for our models. However, due to the inherent differences between the two models used and the distinct characteristics of the datasets, with the large dataset contrasting significantly from the custom dataset, no single approach proved universally effective. Consequently, hyperparameter values were meticulously calibrated to align with the distinct requirements of each dataset, with a concerted effort to maintain consistency in the experimental process by standardizing the random seed for all iterations (refer to Table 3 for hyperparameter details). For the large dataset, a batch size of 64 was employed throughout the training, validation, and testing stages. In contrast, the batch size was experimentally adjusted between 32 and 64 for the custom dataset to fine-tune the model's performance. Our trials spanned a range of epochs from 30 down to 8 or 10 epochs, but our results focus on the most successful outcomes achieved with 10 epochs. Moreover, an early stopping protocol was implemented focusing on global minima loss to mitigate the potential for overfitting, applying a patience parameter of three to four for training up to 10 epochs and extending this to five to seven for sessions exceeding this epoch threshold. In the initial phases of inference, a batch size of 8 was set for testing, which, upon elevation to a batch size of 32, yielded a nominal yet notable decrease in loss.

Table 3. Hyperparameter specifications for model training and evaluation.

Parameter	Value	Description
Seed	1	Used consistent random seed throughout all experiments.
Batch Size (Large Dataset)	64	Used for training, validation, and testing phases.
Batch Size (Custom Dataset)	32 to 64	Varied to find the optimal setting for model performance.
Epochs	10	Range considered: 30 to 8; results reported for 10 epochs.
Early Stopping Patience (<10 Epochs)	3 to 4	To prevent overfitting, applied for models trained for less than 10 epochs.
Early Stopping Patience (>10 Epochs)	5 to 7	Applied when training for more than 10 epochs to mitigate overfitting.
Initial Testing Batch Size	8	Initial size used during the inference phase.
Adjusted Testing Batch Size	32	Incremented to enhance performance and decrease loss.

Scheduler and Learning Rate

The experiments showed that both linear and cosine annealing schedulers can perform well in some cases. Thus, we designed our custom Combined Learning Rate Scheduler algorithm, which utilizes both linear and cosine annealing schedulers, to effectively facilitate the convergence of neural network training. Our algorithm operates in a warm-up linear phase and a cosine annealing phase, as shown in Figure 7.

```
Combined Learning Rate Scheduler

Require: $epoch, initial\_learning\_rate, main\_lr, warmup\_epochs, epochs$
Ensure: $learning\_rate$
1: if $epoch < warmup\_epochs$ then
2:     $warmup\_lr \leftarrow initial\_learning\_rate + (main\_lr - initial\_learning\_rate) \times \frac{epoch}{warmup\_epochs}$
3:     $learning\_rate \leftarrow warmup\_lr$
4: else
5:     $adjusted\_epoch \leftarrow epoch - warmup\_epochs$
6:     $cosine\_annealing\_lr \leftarrow 0.5 \times (1 + \cos(\pi \times \frac{adjusted\_epoch}{epochs}))$
7:     $learning\_rate \leftarrow cosine\_annealing\_lr$
8: end if
9: return $learning\_rate$
```

Figure 7. Combined Learning Rate Scheduler algorithm.

In the initial warm-up phase, our algorithm tackles the cold start problem encountered during training by progressively increasing the learning rate from a minimal initial value (e.g., 1×10^{-8}) to a predefined main learning rate (up to 3.5×10^{-5}). The increase is linear, contingent on the current epoch, which ensures a smooth scaling of the learning rate across the warmup epochs. This method enables the network to gradually adjust from minimal updates, preventing abrupt shifts in the weights that could destabilize the model.

Upon completing the warm-up phase, which in most cases lasts for half the total epochs, the scheduler advances to the cosine annealing phase. Here, the learning rate descends following a half-cosine cycle from the main learning rate down to a lower limit, modulated by the main learning rate and the cosine of the adjusted epoch, with the latter being the epoch count after the conclusion of the warm-up period. This reduction is designed to be non-linear, allowing for smooth and gradual fine-tuning of the weights, thus facilitating the model's convergence to a more advantageous local minimum. The shift from the warm-up to the cosine annealing phase occurs automatically after exceeding the predetermined warmup_epochs, merging the stabilizing effect of the warm-up with the refinement of cosine annealing to enhance the training's end-stage convergence. Our application of this combined learning rate scheduler demonstrates improvements over linear or cosine annealing schedules alone.

4. Results

The best-performing model weights were identified and applied to evaluate the model on the other testing dataset, including the AIRABIC benchmark, both with and without incorporating a Dediacritization Layer. Table 4 comprehensively shows the performance metrics of the AraELECTRA and XLM-R models.

Table 4. Comparative performance metrics of AraELECTRA and XLM-R models trained on custom and large datasets.

Model	Trained on	Evaluated on	Precision	Recall	F1 Score	AUC-ROC	Loss
AraELECTRA	Custom Dataset	Validation set	1.0	1.0	1.0	1.0	0.0002
		Testing set	1.0	1.0	1.0	1.0	0.0006
		Large dataset	0.9437	0.7333	0.8253	0.9525	0.8197
		AIRABIC without Dediacritization Layer	0.7953	0.824	0.8094	0.9139	0.4267
		AIRABIC with Dediacritization Layer	0.9900	1.0	0.9950	0.9998	0.0302
	Large Dataset	Validation set	0.9968	0.9955	0.9961	0.9999	0.0138
		Testing set	0.9995	0.9927	0.9961	0.9999	0.0180
		Custom dataset	0.9114	0.9411	0.9260	0.9698	0.4422
		AIRABIC without Dediacritization Layer	0.6435	0.928	0.7600	0.7745	2.42
		AIRABIC with Dediacritization Layer	0.8169	1.0	0.8992	0.9719	0.6587

Table 4. *Cont.*

Model	Trained on	Evaluated on	Precision	Recall	F1 Score	AUC-ROC	Loss
XLM-R	Custom Dataset	Validation set	1.0	1.0	1.0	1.0	0.0003
		Testing set	1.0	0.9608	0.98	0.9804	0.1426
		Large dataset	0.9223	0.8753	0.8982	0.9476	0.6820
		AIRABIC without Dediacritization Layer	1.0	0.634	0.7760	0.9652	1.372
		AIRABIC with Dediacritization Layer	1.0	1.0	1.0	1.0	0.0001
	Large Dataset	Validation set	0.9977	0.9977	0.9977	0.9997	0.0153
		Testing set	0.9977	0.9977	0.9977	0.9999	0.0151
		Custom dataset	0.5	0.9738	0.6607	0.8223	4.4082
		AIRABIC without Dediacritization Layer	0.5263	1.0	0.6896	0.7182	3.9560
		AIRABIC with Dediacritization Layer	0.5470	1.0	0.7072	0.9042	3.5100

4.1. Our Best Models vs. GPTZero and OpenAI Text Classifier

We benchmarked the performance of our leading models against two well-known AI detectors, GPTZero and the OpenAI Classifier, using the AIRABIC dataset. Notably, our models demonstrated superior performance even without implementing a Dediacritization Layer. This achievement is significant, especially in recognizing HWTs, a task where GPTZero and the OpenAI Classifier showed limitations. Furthermore, our models performed well in processing text with diacritics, outperforming GPTZero and the OpenAI Classifier in this aspect. It is important to mention that these results were achieved without specifically training our models on diacritic-laden texts as they existed on the AIRABIC dataset.

The following results show that the Receiver Operating Characteristic (ROC) curves for the GPTZero and OpenAI Text Classifier models are based on binary classification outcomes from the confusion matrix results below. It is important to note that the curve was generated from discrete classification results rather than continuous probability estimates, which are typically used in ROC analysis. This approach may not fully represent the model's performance across all possible thresholds, as it does not account for the varying degrees of certainty in the predictions. Therefore, the presented data points and curves should be viewed as an approximate measure of the model's capacity to discriminate between HWT and AIGT. However, it should be noted that the ROC curves for our models, which will be presented following the discussion of the GPTZero and OpenAI Text Classifier results, are derived from probability estimates and provide a comprehensive overview of the model's performance across a wide range of decision thresholds.

4.1.1. GPTZero against AIRABIC Benchmark Dataset

Table 5 details GPTZero's performance evaluation as applied to the AIRAIBC Benchmark Dataset. Figure 8 depicts an approximate ROC curve that was generated from discrete classification results rather than continuous probability estimates, which are typically used in ROC analysis.

Table 5. Evaluation of GPTZero detector on AIRABIC benchmark dataset.

GPTZero	Predicted: Human-Written	Predicted: AI-Generated	Performance Metrics	Value
Actual: Human-written	150 (TP)	350 (FN)	Sensitivity	30%
			Specificity	95%
			Precision	86.7%
Actual: AI-generated	23 (FP)	477 (TN)	Accuracy	62.7%
			F1-Score	44.5%

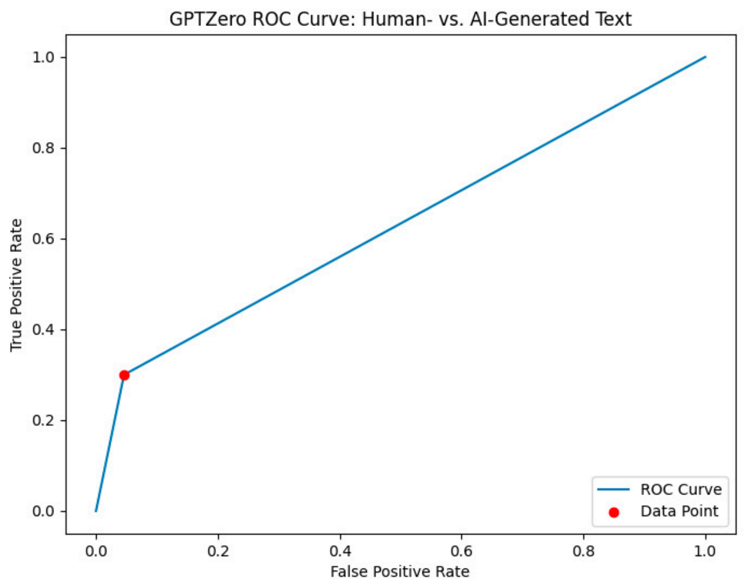


Figure 8. ROC curve of GPTZero detector.

4.1.2. OpenAI Text Classifier against AIRABIC Benchmark Dataset

Table 6 shows OpenAI Text Classifier's performance evaluation as applied to the AIRAIBC Benchmark Dataset. Figure 9 depicts an approximate ROC curve that was generated from discrete classification results rather than continuous probability estimates, which are typically used in ROC analysis.

Table 6. Evaluation of OpenAI Text Classifier on AIRABIC benchmark dataset.

OpenAI Text Classifier	Predicted: Human-Written	Predicted: AI-Generated	Performance Metrics	Value
Actual: Human-written	0 (TP)	500 (FN)	Sensitivity Specificity	0% 100%
			Precision	0%
Actual: AI-generated	0 (FP)	500 (TN)	Accuracy F1-Score	50% 0%

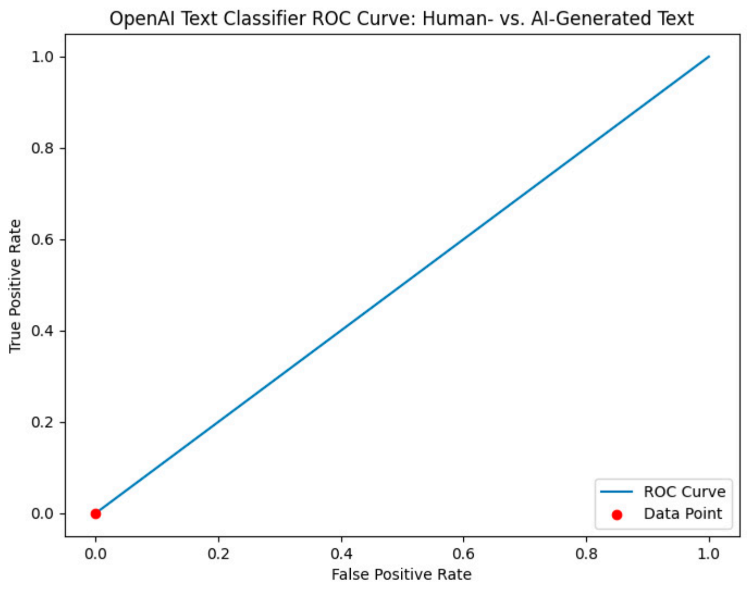


Figure 9. ROC curve of OpenAI Text Classifier.

4.1.3. Fine-Tuned AraELECTRA Model against AIRABIC Benchmark Dataset

Table 7 details the fine-tuned AraELECTRA model's performance evaluation as applied to the AIRAIBC Benchmark Dataset without applying the Dediacritization Layer. Figure 10 depicts the ROC curve derived from probability estimates and provides a comprehensive overview of the model's performance across a wide range of decision thresholds.

Table 7. Evaluation of the fine-tuned AraELECTRA model without using a Dediacritization Layer on the AIRABIC benchmark dataset.

AraELECTRA Model	Predicted: Human-Written	Predicted: AI-Generated	Performance Metrics	Value
Actual: Human-written	412 (TP)	88 (FN)	Sensitivity Specificity	82% 79%
Actual: AI-generated	106 (FP)	394 (TN)	Precision Accuracy F1-Score	79% 81% 82%

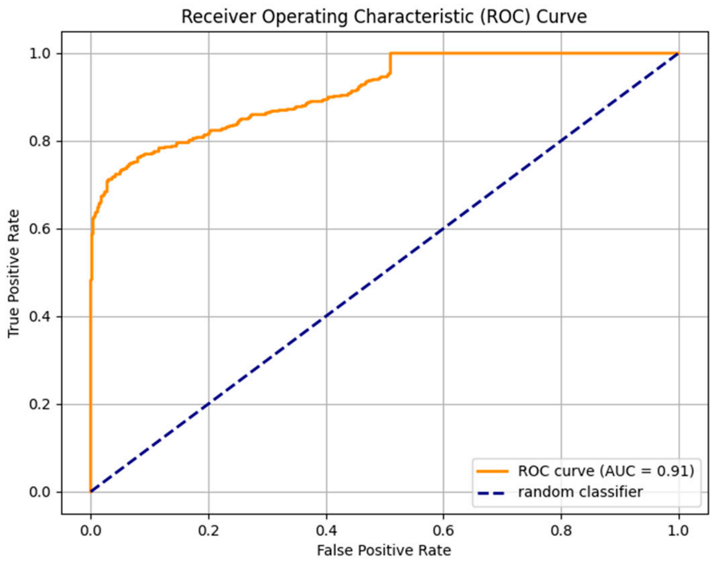


Figure 10. ROC curve of the fine-tuned AraELECTRA model without using a Dediacritization Layer.

Table 8 details the fine-tuned AraELECTRA model's performance evaluation as applied to the AIRAIBC Benchmark Dataset after using the Dediacritization Layer. Figure 11 depicts the ROC curve derived from probability estimates and provides a comprehensive overview of the model's performance across a wide range of decision thresholds.

Table 8. Evaluation of the fine-tuned AraELECTRA model using a Dediacritization Layer on the AIRABIC benchmark dataset.

AraELECTRA Model	Predicted: Human-Written	Predicted: AI-Generated	Performance Metrics	Value
Actual: Human-written	500 (TP)	0 (FN)	Sensitivity Specificity	100% 99%
Actual: AI-generated	5 (FP)	495 (TN)	Precision Accuracy F1-Score	99% 99% 99%

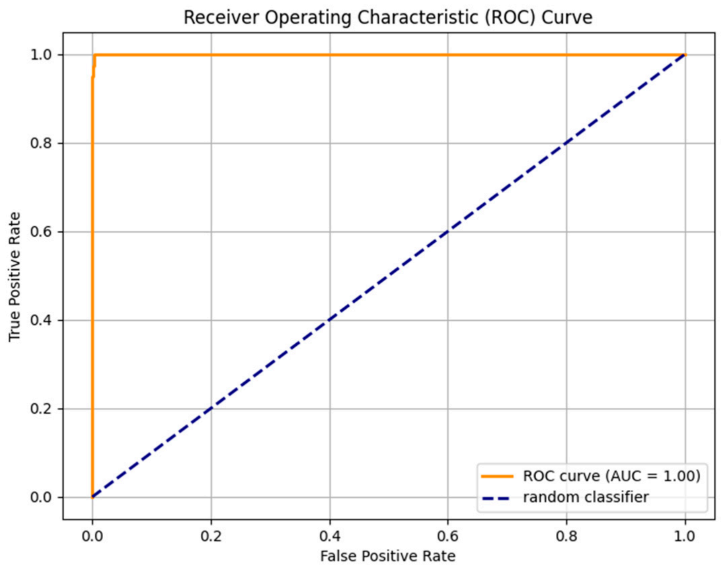


Figure 11. ROC curve of the fine-tuned AraELECTRA model using a Dediacritization Layer.

4.1.4. Fine-Tuned XLM-R Model against AIRABIC Benchmark Dataset

Table 9 details the fine-tuned XLM-R model's performance evaluation as applied to the AIRAIBC Benchmark Dataset without applying the Dediacritization Layer. Figure 12 depicts the ROC curve derived from probability estimates and provides a comprehensive overview of the model's performance across a wide range of decision thresholds.

Table 9. Evaluation of fine-tuned XLM-R model without using Dediacritization Layer on AIRABIC benchmark dataset.

XLM-R Model	Predicted: Human-Written	Predicted: AI-Generated	Performance Metrics	Value
Actual: Human-written	317 (TP)	183 (FN)	Sensitivity Specificity	63% 100%
Actual: AI-generated	0 (FP)	500 (TN)	Precision Accuracy F1-Score	100% 81% 77%

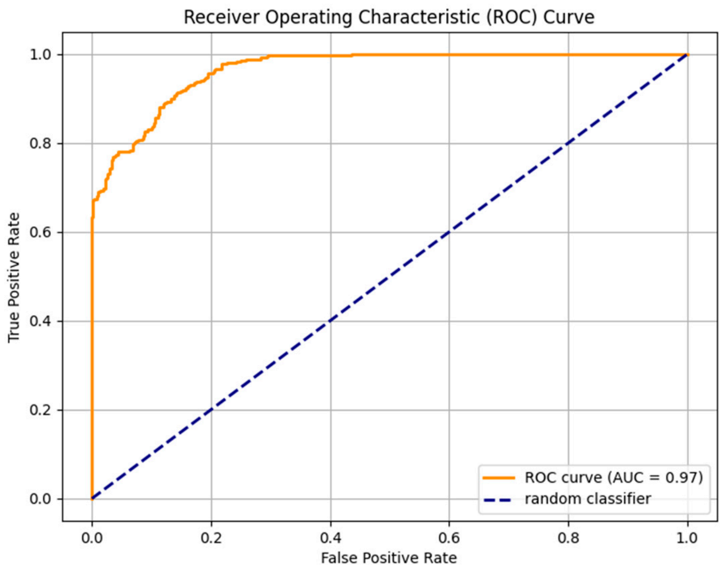


Figure 12. ROC curve of the fine-tuned XLM-R model without using a Dediacritization Layer.

Table 10 details the fine-tuned XLM-R model's performance evaluation as applied to the AIRAIBC Benchmark Dataset after using the Dediacritization Layer. Figure 13 depicts the ROC curve derived from probability estimates and provides a comprehensive overview of the model's performance across a wide range of decision thresholds.

Table 10. Evaluation of the fine-tuned XLM-R model using a Dediacritization Layer on the AIRABIC benchmark dataset.

XLM-R Model	Predicted: Human-Written	Predicted: AI-Generated	Performance Metrics	Value
Actual: Human-written	500 (TP)	0 (FN)	Sensitivity	100%
			Specificity	100%
			Precision	100%
Actual: AI-generated	0 (FP)	500 (TN)	Accuracy	100%
			F1-Score	100%

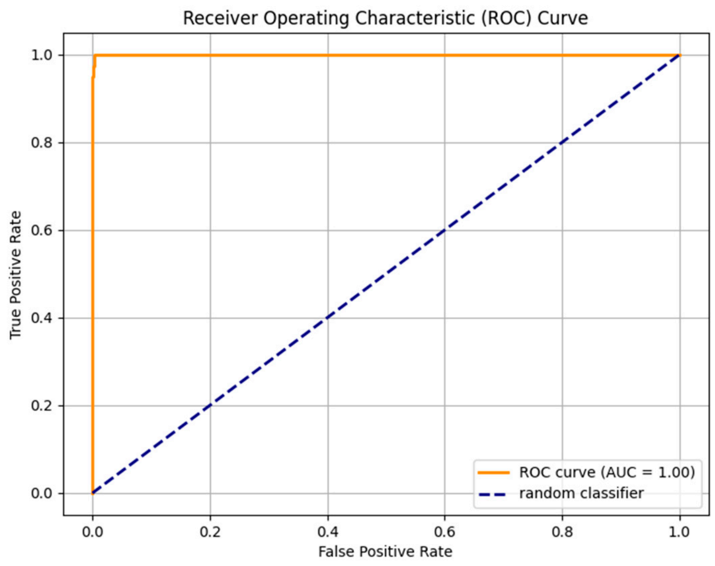


Figure 13. Fine-tuned XLM-R model using Dediacritization Layer ROC curve on AIRABIC benchmark dataset.

5. Discussion

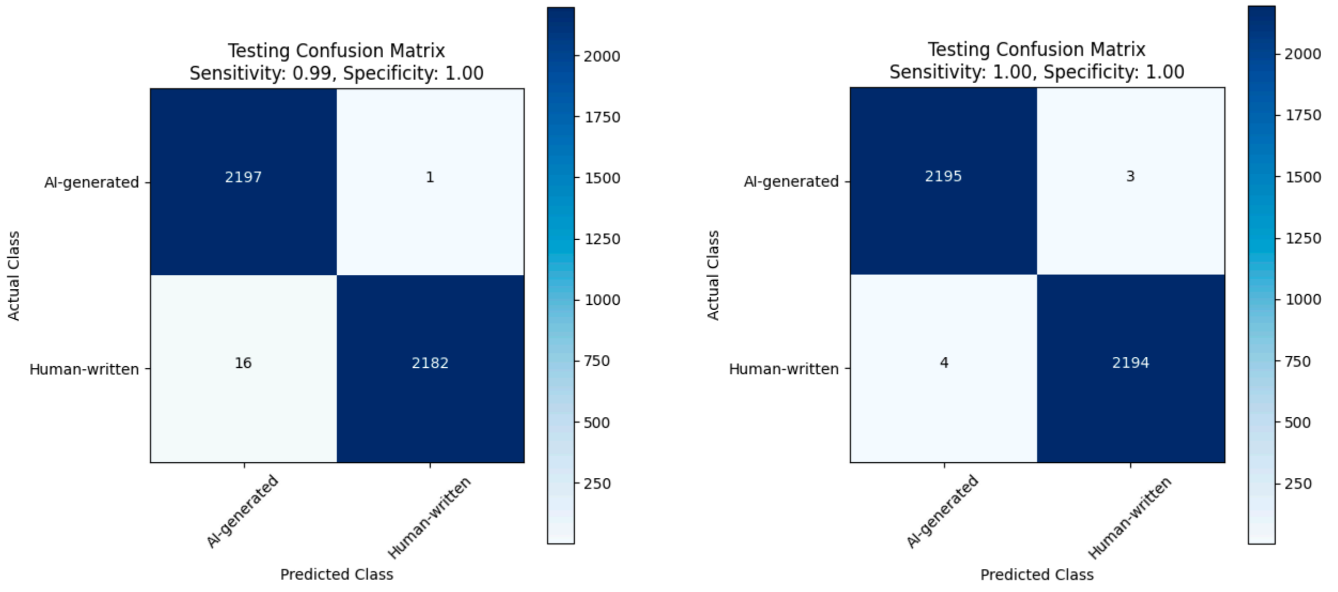
5.1. Large Dataset vs. Custom Dataset Content Variation

When the models were applied to the two datasets, the performance disparities between the two models can be attributed to various factors. First, the large dataset content of AIGT has some non-Arabic scripts due to the translation. Moreover, the narrative structure within this dataset is generally limited to one or two paragraphs per instance of HWT, resulting in a narrower range of writing styles than the custom dataset. In contrast, the custom dataset was developed meticulously by handcrafting it to encompass a diverse range of HWT patterns. Additionally, the large dataset's scope of AIGT is confined to translated versions of ChatGPT 3.5 outputs. This presents a limitation, as it fails to encompass more advanced and varied writing patterns characteristic of GPT-4 and BARD outputs, which are critical for a robust evaluation of AIGT.

Furthermore, it is important to highlight that the HWT was obtained from Arabic-SQuAD [60] in a large dataset comprising content translated from the English version of SQuAD [49]. This translation aspect is crucial as the quality of these translations inherently affects the dataset's robustness. Due to its reliance on translated content, the large dataset may not fully capture the nuances and complexity of native Arabic text, which could affect the effectiveness of models trained on this dataset in understanding and processing authentic Arabic language constructs in comparison with the native dataset, such as the custom dataset.

5.2. XLM-R vs. AraELECTRA Performance

The comparative analysis reveals that fine-tuning both the XLM-R and AraELECTRA models demonstrates exceptional performance in classifying HWT and AIGT on large and custom datasets. Specifically, in the larger dataset containing a mix of Arabic and non-Arabic vocabulary, the XLM-R model was expected to perform well due to its capacity to recognize over 100 languages [51]. It accurately classified 4362 out of 4369 examples, a slight improvement over the AraELECTRA model, which correctly identified 4352 and missed classifying 17 examples, as shown in Figure 14.

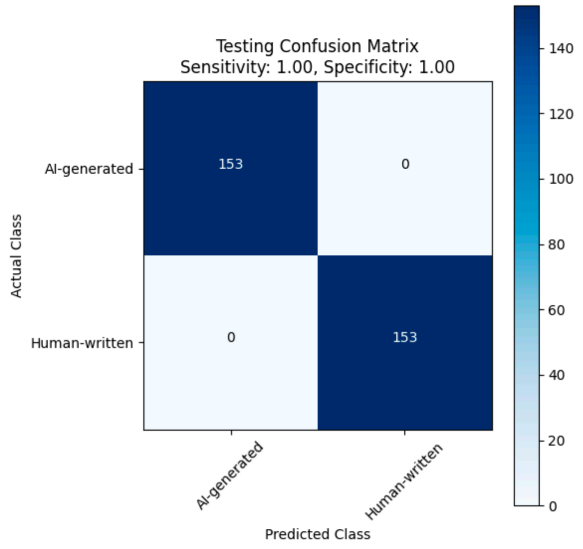


(**a**) (**b**)

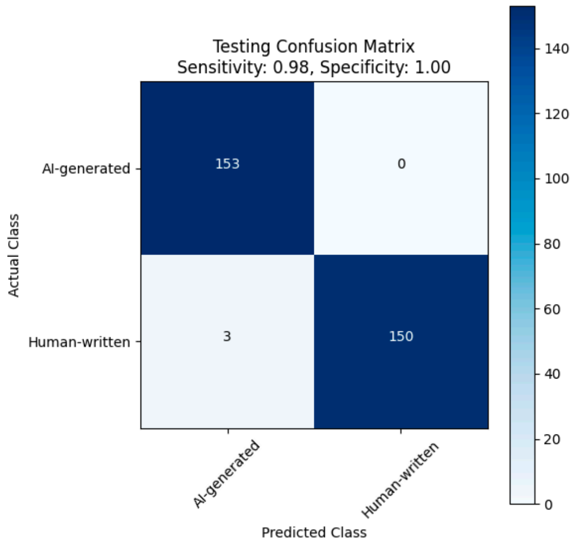
Figure 14. Comparison of the two models' performance on the large dataset testing set: (**a**) AraELECTRA; (**b**) XLM-R.

The high accuracy of the AraELECTRA model, even in a dataset mixed with Arabic and non-Arabic vocabulary, can be attributed to different factors. First, the RTD technique distinguishes between real and fake words within a text, is a core component of the ELECTRA training method, and plays an important role in enhancing the model's performance. In the RTD setup, a small generator network proposes words to replace tokens in a text, and the discriminator (the main model) has to determine if a word is original or replaced by the generator. This approach is more efficient than traditional language modeling, encouraging the model to learn finer distinctions in word choice and context. That method is particularly beneficial for distinguishing between AIGT and HWT. Secondly, being trained explicitly on Arabic datasets, it is likely to have a more nuanced understanding of Arabic linguistic features. This can contribute to its strong performance in datasets with prominent Arabic text, even if mixed with other languages.

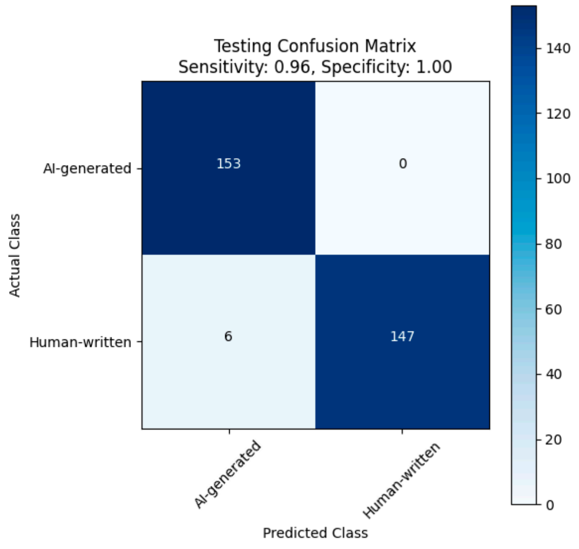
This distinction becomes more evident when examining the custom dataset. While both models performed admirably, the AraELECTRA model achieved perfect accuracy, correctly classifying all instances, as shown in Figure 15a. The XLM-R model was tested under the same hyperparameters as the AraELECTRA model in an experiment focused on the custom dataset. The XLM-R model initially misclassified three instances in this setup, as shown in Figure 15b. Further adjustments were made by lowering both the initial and foremost learning rates. After these modifications, the model incorrectly classified six instances, as shown in Figure 15c. However, this tuning enhanced the XLM-R model's performance on other datasets during the inference phase, particularly with the custom and AIRABIC datasets, when operating with the adjusted lower initial and default learning rates.



(**a**)



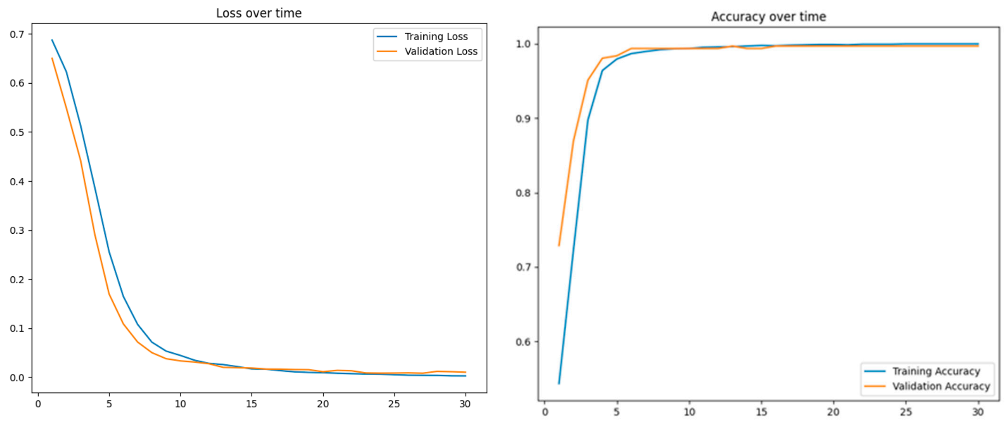
(**b**)



(**c**)

Figure 15. Performance comparison of two models on the custom dataset's testing set: (**a**) AraELECTRA model; (**b**) XLM-R model using the same hyperparameters as the AraELECTRA model; (**c**) XLM-R model with adjusted hyperparameters ('learning_rate': 3.2×10^{-5}, 'initial_learning_rate': 1×10^{-8}).

The big challenge in fine-tuning pre-trained models is preventing overfitting. Our study addressed this by carefully adjusting the learning rates for the AraELECTRA model during its application to the custom dataset. Specifically, the initial learning rate was reduced to 1×10^{-8}, and the main learning rate was reduced to approximately 1.5×10^{-6}. These adjustments enabled the model to be effectively trained for up to 30 epochs. The optimal performance was observed at epoch 27, where the model achieved its lowest loss value of 0.008, as illustrated in Figure 16.



(**a**) (**b**)

Figure 16. Performance of the AraELECTRA model on a custom dataset. (**a**) The loss during the training and validation phases, where the *x*-axis represents epochs (e.g., 5 epochs, 10 epochs, 15 epochs) indicating the training process's progress over time. (**b**) The training and validation accuracy over the same epochs. This representation provides insights into the model's learning dynamics and overall effectiveness throughout the training period.

In contrast, the XLM-R model exhibited a different pattern. The validation loss started relatively low, at 0.1, even with the previously mentioned hyperparameters. Subsequent efforts to address this issue have been taken for tackling the loss, such as increasing the dropout ratio to 0.2, 0.3, and subsequently to 0.5, but were unsuccessful in significantly improving the results or mitigating the overfitting problem. Consequently, it was observed that the XLM-R model is particularly susceptible to overfitting. The most favorable results obtained from running the XLM-R model on the custom dataset were by running the model for 10 epochs using the following hyperparameters: batch size 32 with enhancements on the learning rate by applying a warmup phase, learning_rate: 3.2×10^{-5}, initial_learning_rate: 1×10^{-8}, warmup_epochs: 2. The model's best loss was obtained at epoch 6, with a validation loss of 0.0002. Beyond this point, the model's loss escalated, indicative of increasing overfitting.

In the experiments conducted on the large dataset, both the XLM-R and AraELECTRA models initially demonstrated low validation losses, starting at 0.03. Despite this promising outset, the XLM-R model exhibited notable overfitting, more so than its counterpart. Extensive experimentation was conducted to optimize hyperparameters specifically for this dataset. Despite the variability in results, both models demonstrated proficiency in detecting AIGT and HWT, though validation losses fluctuated and eventually increased. Optimal results on this dataset were achieved using the XLM-R model for ten epochs, with a batch size of 64, an initial learning rate of 5×10^{-8}, a learning rate of 3.2×10^{-6}, and warmup epochs set at five. These hyperparameters yielded the most favorable outcomes during the inference phase on both the custom and AIRABIC datasets despite being less effective than runs on the custom dataset alone.

In contrast, AraELECTRA displayed superior performance during the inference phase on both the custom and AIRABIC datasets compared to XLM-R. This suggests greater robustness in AraELECTRA, although its results were less optimal than when trained solely on the custom dataset. This disparity underscores the crucial role of dataset quality and composition in model training. Native datasets, with their inherent linguistic authenticity, often provide a more conducive environment for effective training than datasets composed predominantly of translated material.

The robustness of the model is evident during the inference phase when tested against various datasets. Notably, XLM-R demonstrates poor performance following training on large datasets. In contrast, AraELECTRA exhibits satisfactory results under similar conditions. However, when both models are trained on a custom dataset, they each show improved performance, particularly against the benchmark dataset, which was the measurement criteria of our study. More details are in the following section.

5.3. Improvement Contribution of Dediacritization Layer toward the Classifier

The findings elucidated that the dataset's robustness on which the classifier is trained emerges as the pivotal factor influencing its efficacy. When trained on the custom dataset, the classifier demonstrated enhanced performance, outperforming the results obtained from a significantly larger, translated dataset (large dataset). Nonetheless, the inherent robustness of the dataset proved insufficient for the classifier to effectively process texts laden with diacritics, as in 500 examples of the AIRABIC dataset. Therefore, integrating a Dediacritization Layer enhances the classifier's ability to receive diacritics-free texts, thereby markedly improving its classification accuracy. Table 11 compares the detection of the two fine-tuned models before and after the use of the Dediacritization Layer.

Table 11. Dediacritization Layer contribution toward classifier efficacy: comparative performance metrics of the two models against the AIRABIC benchmark dataset with and without Dediacritization Layer intervention.

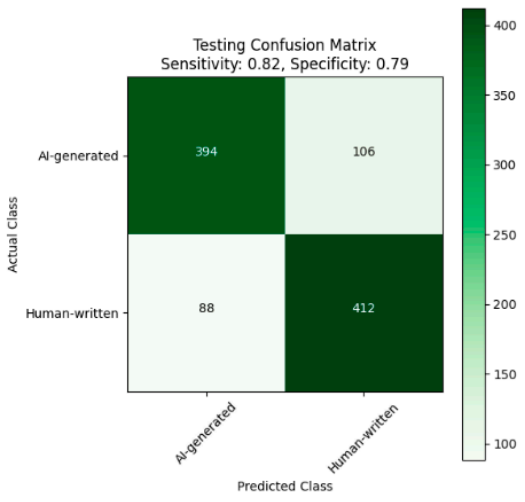
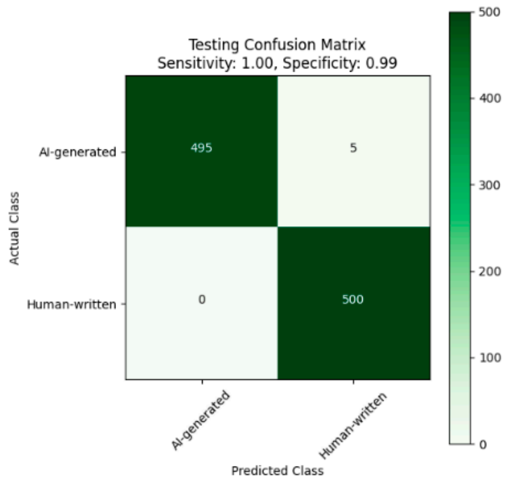
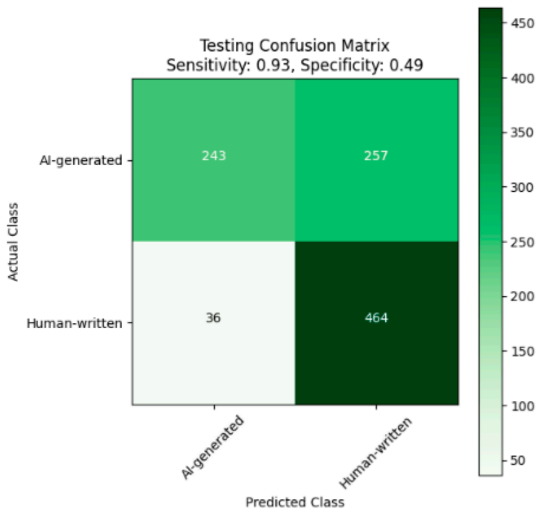
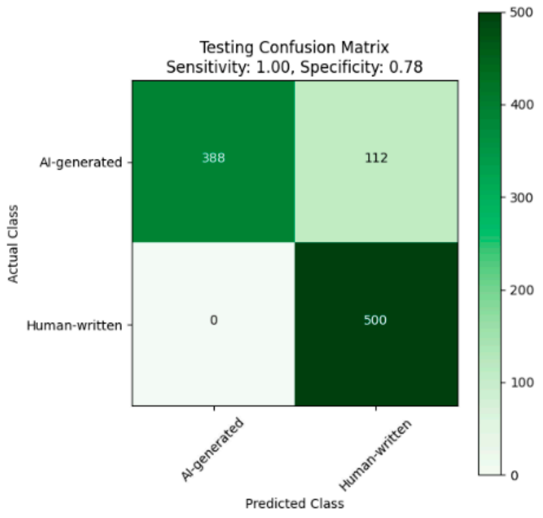
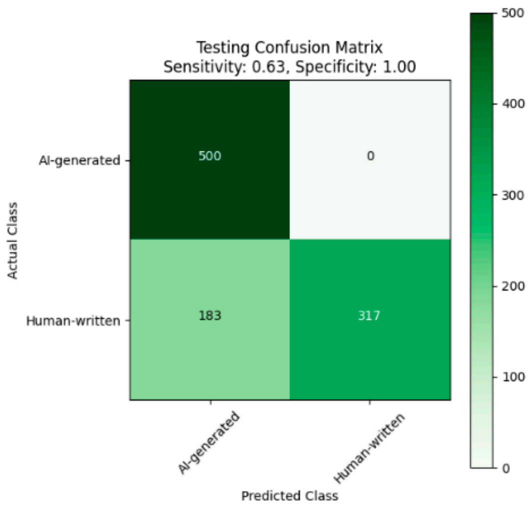
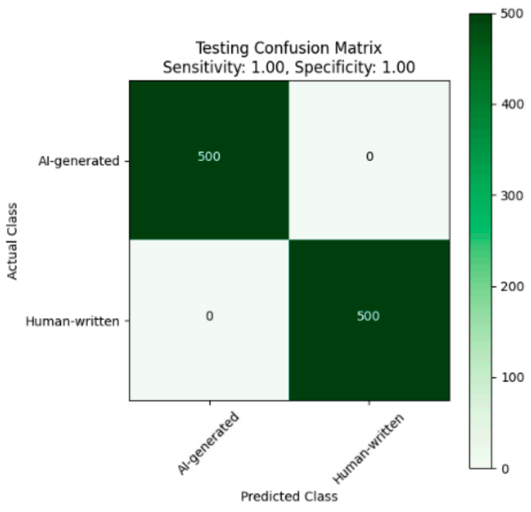
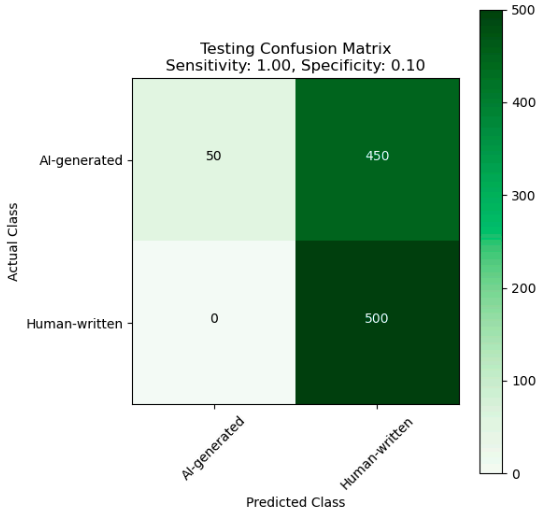
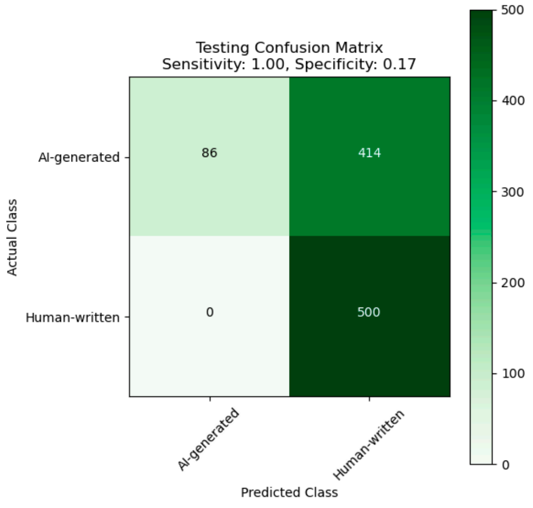
Without Dediacritization Layer	**With Dediacritization Layer**
 (**a**)	 (**b**)
Fine-tuned AraELECTRA model trained on the custom dataset. (**a**) Without Dediacritization Layer. (**b**) With Dediacritization Layer.	
 (**a**)	 (**b**)
Fine-tuned AraELECTRA model trained on the large dataset. (**a**) Without Dediacritization Layer. (**b**) With Dediacritization Layer.	

Table 11. *Cont.*

Without Dediacritization Layer	With Dediacritization Layer
 (**a**)	 (**b**)
Fine-tuned XLM-R model trained on the custom dataset. (**a**) Without Dediacritization Layer. (**b**) With Dediacritization Layer.	
 (**a**)	 (**b**)
Fine-tuned XLM-R model trained on the large dataset. (**a**) Without Dediacritization Layer. (**b**) With Dediacritization Layer.	

6. Conclusions and Future Work

In this paper, we developed the first Arabic AI detector after ChatGPT emerged. It was designed to distinguish between Arabic synthetic ChatGPT- and BARD-generated texts and HWTs, a domain in which current AI detection systems have shown considerable shortcomings. Two encoder-based Transformer architecture models, AraELECTRA and XLM-R, were fine-tuned on datasets intentionally devoid of diacritics-laden texts. This was conducted to assess how well our classifiers recognize diacritical content, particularly using the AIRABIC benchmark dataset, which includes texts both with and without diacritics. These models outperformed two well-known AI detectors, GPTZero and the OpenAI Text Classifier. Incorporating a Dediacritization Layer further enhanced the detection accuracy, increasing it to 99% and, in some instances, 100%. This marks a significant advancement in developing an AI text classifier for Arabic texts that distinguishes between HWT and AIGT from ChatGPT and BARD. Building on this success, in our future research, we intend to delve deeper into this domain by utilizing a variety of pre-trained models. These models will be trained on texts containing both forms of Arabic script, with and without diacritics. A pivotal aspect of future research will be to compare the effectiveness of two methodologies: training models on diacritics-laden texts versus employing a Dediacritization Layer, with the latter having already demonstrated significant efficacy. This comparison is crucial in identifying the superior approach, particularly in developing a classifier that supports multiple languages.

Author Contributions: Conceptualization, H.A. and K.E.; methodology, H.A.; software, H.A.; validation, H.A., A.E.-S. and K.E.; formal analysis, H.A.; investigation, H.A.; resources, H.A.; data curation, H.A.; writing—original draft preparation, H.A.; writing—review and editing, H.A., A.E.-S. and K.E.; visualization, H.A.; supervision, A.E.-S. and K.E.; project administration, K.E. All authors have read and agreed to the published version of the manuscript.

Funding: The authors received no financial support for this research.

Data Availability Statement: All materials related to our study, including the trained models, detailed results reports, source code, and datasets, are publicly accessible via our dedicated GitHub repository: https://github.com/Hamed1Hamed/Arabic_AI_Detector (accessed on 1 February 2024).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ahmed, A.; Ali, N.; Alzubaidi, M.; Zaghouani, W.; Abd-alrazaq, A.A.; Househ, M. Freely available Arabic corpora: A scoping review. *Comput. Methods Programs Biomed. Update* **2022**, *2*, 100049. [CrossRef]
2. UNESCO. World Arabic Language Day. Available online: https://www.unesco.org/en/world-arabic-language-day (accessed on 19 December 2023).
3. Chemnad, K.; Othman, A. Advancements in Arabic Text-to-Speech Systems: A 22-Year Literature Review. *IEEE Access* **2023**, *11*, 30929–30954. [CrossRef]
4. United Nations. Official Languages. Available online: https://www.un.org/en/our-work/official-languages (accessed on 25 December 2023).
5. Obeid, O.; Zalmout, N.; Khalifa, S.; Taji, D.; Oudah, M.; Alhafni, B.; Inoue, G.; Eryani, F.; Erdmann, A.; Habash, N. CAMeL tools: An open source python toolkit for Arabic natural language processing. In Proceedings of the Twelfth Language Resources and Evaluation Conference, Marseille, France, 11–16 May 2020; pp. 7022–7032.
6. Farghaly, A.; Shaalan, K. Arabic natural language processing: Challenges and solutions. *ACM Trans. Asian Lang. Inf. Process. (TALIP)* **2009**, *8*, 1–22. [CrossRef]
7. Darwish, K.; Habash, N.; Abbas, M.; Al-Khalifa, H.; Al-Natsheh, H.T.; Bouamor, H.; Bouzoubaa, K.; Cavalli-Sforza, V.; El-Beltagy, S.R.; El-Hajj, W. A panoramic survey of natural language processing in the Arab world. *Commun. ACM* **2021**, *64*, 72–81. [CrossRef]
8. Habash, N.Y. Introduction to Arabic natural language processing. *Synth. Lect. Hum. Lang. Technol.* **2010**, *3*, 1–187.
9. GPTZero. Available online: https://gptzero.me/ (accessed on 1 June 2023).
10. OpenAI. Available online: https://beta.openai.com/ai-text-classifier (accessed on 1 June 2023).
11. Alshammari, H.; El-Sayed, A. AIRABIC: Arabic Dataset for Performance Evaluation of AI Detectors. In Proceedings of the 2023 International Conference on Machine Learning and Applications (ICMLA), Jacksonville Riverfront, FL, USA, 15–17 December 2023; pp. 1–5.
12. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A. Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1877–1901.
13. Chowdhery, A.; Narang, S.; Devlin, J.; Bosma, M.; Mishra, G.; Roberts, A.; Barham, P.; Chung, H.W.; Sutton, C.; Gehrmann, S. Palm: Scaling language modeling with pathways. *arXiv* **2022**, arXiv:2204.02311.
14. OpenAI. ChatGPT (Mar 14 Version) [Large Language Model]. Available online: https://chat.openai.com/chat (accessed on 14 March 2023).
15. Trichopoulos, G.; Konstantakis, M.; Caridakis, G.; Katifori, A.; Koukouli, M. Crafting a Museum Guide Using ChatGPT4. *Big Data Cogn. Comput.* **2023**, *7*, 148. [CrossRef]
16. Pegoraro, A.; Kumari, K.; Fereidooni, H.; Sadeghi, A.-R. To ChatGPT, or not to ChatGPT: That is the question! *arXiv* **2023**, arXiv:2304.01487.
17. Wölfel, M.; Shirzad, M.B.; Reich, A.; Anderer, K. Knowledge-Based and Generative-AI-Driven Pedagogical Conversational Agents: A Comparative Study of Grice's Cooperative Principles and Trust. *Big Data Cogn. Comput.* **2023**, *8*, 2. [CrossRef]
18. Hassani, H.; Silva, E.S. The role of ChatGPT in data science: How ai-assisted conversational interfaces are revolutionizing the field. *Big Data Cogn. Comput.* **2023**, *7*, 62. [CrossRef]
19. Bard. Available online: https://bard.google.com/ (accessed on 30 January 2023).
20. Sheng, E.; Chang, K.-W.; Natarajan, P.; Peng, N. Societal biases in language generation: Progress and challenges. *arXiv* **2021**, arXiv:2105.04054.

21. Weidinger, L.; Uesato, J.; Rauh, M.; Griffin, C.; Huang, P.-S.; Mellor, J.; Glaese, A.; Cheng, M.; Balle, B.; Kasirzadeh, A. Taxonomy of risks posed by language models. In Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency, Seoul, Republic of Korea, 21–24 June 2022; pp. 214–229.
22. Zhuo, T.Y.; Huang, Y.; Chen, C.; Xing, Z. Exploring ai ethics of chatgpt: A diagnostic analysis. *arXiv* **2023**, arXiv:2301.12867.
23. Cotton, D.R.; Cotton, P.A.; Shipway, J.R. Chatting and cheating: Ensuring academic integrity in the era of ChatGPT. *Innov. Educ. Teach. Int.* **2023**, *61*, 228–239. [CrossRef]
24. Gao, C.A.; Howard, F.M.; Markov, N.S.; Dyer, E.C.; Ramesh, S.; Luo, Y.; Pearson, A.T. Comparing scientific abstracts generated by ChatGPT to original abstracts using an artificial intelligence output detector, plagiarism detector, and blinded human reviewers. *BioRxiv* **2022**. [CrossRef]
25. Anderson, N.; Belavy, D.L.; Perle, S.M.; Hendricks, S.; Hespanhol, L.; Verhagen, E.; Memon, A.R. AI did not write this manuscript, or did it? Can we trick the AI text detector into generated texts? The potential future of ChatGPT and AI in Sports & Exercise Medicine manuscript generation. *BMJ Open Sport Exerc. Med.* **2023**, *9*, e001568. [PubMed]
26. Kumar, S.; Balachandran, V.; Njoo, L.; Anastasopoulos, A.; Tsvetkov, Y. Language generation models can cause harm: So what can we do about it? An actionable survey. *arXiv* **2022**, arXiv:2210.07700.
27. Abramski, K.; Citraro, S.; Lombardi, L.; Rossetti, G.; Stella, M. Cognitive network science reveals bias in GPT-3, GPT-3.5 Turbo, and GPT-4 mirroring math anxiety in high-school students. *Big Data Cogn. Comput.* **2023**, *7*, 124.
28. Taecharungroj, V. "What Can ChatGPT Do?" Analyzing Early Reactions to the Innovative AI Chatbot on Twitter. *Big Data Cogn. Comput.* **2023**, *7*, 35. [CrossRef]
29. Zellers, R.; Holtzman, A.; Rashkin, H.; Bisk, Y.; Farhadi, A.; Roesner, F.; Choi, Y. Defending against neural fake news. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019.
30. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. Improving Language Understanding by Generative Pre-Training. 2018. *work in progress*. Available online: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf (accessed on 1 December 2023).
31. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
32. Gehrmann, S.; Strobelt, H.; Rush, A.M. Gltr: Statistical detection and visualization of generated text. *arXiv* **2019**, arXiv:1906.04043.
33. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog* **2019**, *1*, 9.
34. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv* **2019**, arXiv:1907.11692.
35. Adelani, D.I.; Mai, H.; Fang, F.; Nguyen, H.H.; Yamagishi, J.; Echizen, I. Generating sentiment-preserving fake online reviews using neural language models and their human-and machine-based detection. In *Advanced Information Networking and Applications, Proceedings of the 34th International Conference on Advanced Information Networking and Applications (AINA-2020), Caserta, Italy, 15–17 April 2020*; pp. 1341–1354.
36. Uchendu, A.; Le, T.; Shu, K.; Lee, D. Authorship attribution for neural text generation. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Online, 16–20 November 2020; pp. 8384–8395.
37. Keskar, N.S.; McCann, B.; Varshney, L.R.; Xiong, C.; Socher, R. Ctrl: A conditional transformer language model for controllable generation. *arXiv* **2019**, arXiv:1909.05858.
38. Lample, G.; Conneau, A. Cross-lingual language model pretraining. *arXiv* **2019**, arXiv:1901.07291.
39. Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.R.; Le, Q.V. Xlnet: Generalized autoregressive pretraining for language understanding. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019.
40. Dathathri, S.; Madotto, A.; Lan, J.; Hung, J.; Frank, E.; Molino, P.; Yosinski, J.; Liu, R. Plug and play language models: A simple approach to controlled text generation. *arXiv* **2019**, arXiv:1912.02164.
41. Ng, N.; Yee, K.; Baevski, A.; Ott, M.; Auli, M.; Edunov, S. Facebook FAIR's WMT19 news translation task submission. *arXiv* **2019**, arXiv:1907.06616.
42. Fagni, T.; Falchi, F.; Gambini, M.; Martella, A.; Tesconi, M. TweepFake: About detecting deepfake tweets. *PLoS ONE* **2021**, *16*, e0251415. [CrossRef] [PubMed]
43. Harrag, F.; Debbah, M.; Darwish, K.; Abdelali, A. Bert transformer model for detecting Arabic GPT2 auto-generated tweets. *arXiv* **2021**, arXiv:2101.09345.
44. Nguyen-Son, H.-Q.; Thao, T.; Hidano, S.; Gupta, I.; Kiyomoto, S. Machine translated text detection through text similarity with round-trip translation. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online, 6–11 June 2021; pp. 5792–5797.
45. Antoun, W.; Baly, F.; Hajj, H. AraGPT2: Pre-trained transformer for Arabic language generation. *arXiv* **2020**, arXiv:2012.15520.
46. Clark, K.; Luong, M.-T.; Le, Q.V.; Manning, C.D. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv* **2020**, arXiv:2003.10555.
47. Jawahar, G.; Abdul-Mageed, M.; Lakshmanan, L.V. Automatic Detection of Entity-Manipulated Text using Factual Knowledge. *arXiv* **2022**, arXiv:2203.10343.

48. Guo, B.; Zhang, X.; Wang, Z.; Jiang, M.; Nie, J.; Ding, Y.; Yue, J.; Wu, Y. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *arXiv* **2023**, arXiv:2301.07597.
49. Rajpurkar, P.; Zhang, J.; Lopyrev, K.; Liang, P. Squad: 100,000+ questions for machine comprehension of text. *arXiv* **2016**, arXiv:1606.05250.
50. Fan, A.; Jernite, Y.; Perez, E.; Grangier, D.; Weston, J.; Auli, M. ELI5: Long form question answering. *arXiv* **2019**, arXiv:1907.09190.
51. Kirchenbauer, J.; Geiping, J.; Wen, Y.; Katz, J.; Miers, I.; Goldstein, T. A watermark for large language models. *arXiv* **2023**, arXiv:2301.10226.
52. Fernandez, P.; Chaffin, A.; Tit, K.; Chappelier, V.; Furon, T. Three bricks to consolidate watermarks for large language models. *arXiv* **2023**, arXiv:2308.00113.
53. Christ, M.; Gunn, S.; Zamir, O. Undetectable Watermarks for Language Models. *arXiv* **2023**, arXiv:2306.09194.
54. Mitrović, S.; Andreoletti, D.; Ayoub, O. Chatgpt or human? detect and explain. explaining decisions of machine learning model for detecting short chatgpt-generated text. *arXiv* **2023**, arXiv:2301.13852.
55. Antoun, W.; Mouilleron, V.; Sagot, B.; Seddah, D. Towards a Robust Detection of Language Model Generated Text: Is ChatGPT that Easy to Detect? *arXiv* **2023**, arXiv:2306.05871.
56. De Bruyn, M.; Lotfi, E.; Buhmann, J.; Daelemans, W. MFAQ: A multilingual FAQ dataset. *arXiv* **2021**, arXiv:2109.12870.
57. Martin, L.; Muller, B.; Suárez, P.J.; Dupont, Y.; Romary, L.; de La Clergerie, É.V.; Seddah, D.; Sagot, B. CamemBERT: A tasty French language model. *arXiv* **2019**, arXiv:1911.03894.
58. Antoun, W.; Sagot, B.; Seddah, D. Data-Efficient French Language Modeling with CamemBERTa. *arXiv* **2023**, arXiv:2306.01497.
59. Conneau, A.; Khandelwal, K.; Goyal, N.; Chaudhary, V.; Wenzek, G.; Guzmán, F.; Grave, E.; Ott, M.; Zettlemoyer, L.; Stoyanov, V. Unsupervised cross-lingual representation learning at scale. *arXiv* **2019**, arXiv:1911.02116.
60. Mozannar, H.; Hajal, K.E.; Maamary, E.; Hajj, H. Neural Arabic question answering. *arXiv* **2019**, arXiv:1906.05394.
61. Clark, J.H.; Choi, E.; Collins, M.; Garrette, D.; Kwiatkowski, T.; Nikolaev, V.; Palomaki, J. Tydi qa: A benchmark for information-seeking question answering in ty pologically di verse languages. *Trans. Assoc. Comput. Linguist.* **2020**, *8*, 454–470. [CrossRef]
62. Lewis, P.; Oğuz, B.; Rinott, R.; Riedel, S.; Schwenk, H. MLQA: Evaluating cross-lingual extractive question answering. *arXiv* **2019**, arXiv:1910.07475.
63. Nguyen, T.T.; Hatua, A.; Sung, A.H. How to Detect AI-Generated Texts? In *2023 IEEE 14th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*; IEEE: Piscataway, NJ, USA, 2023.
64. Zerrouki, T.; Balla, A. Tashkeela: Novel corpus of Arabic vocalized texts, data for auto-diacritization systems. *Data Brief.* **2017**, *11*, 147–151. [CrossRef]
65. Shamela. Available online: https://shamela.ws/ (accessed on 3 August 2023).
66. Aldiwan: Encyclopedia of Arabic Poetry. Available online: https://www.aldiwan.net/ (accessed on 1 October 2023).
67. Antoun, W.; Baly, F.; Hajj, H. AraELECTRA: Pre-training text discriminators for Arabic language understanding. *arXiv* **2020**, arXiv:2012.15516.