JISTech (Journal of Islamic Science and Technology)

JISTech, 8(2), 169-191, Juli-Desember 2023

ISSN: 2528-5718

http://jurnal.uinsu.ac.id/index.php/jistech

DIACRITIC-AWARE ALIGNMENT AND CLASSIFICATION IN ARABIC SPEECH: A FUSION OF FUZTPI AND ML MODELS

Adel Sabour[1], Abdeltawab Hendawi[2], and Mohamed Ali[1]

[1] University of Washington, Tacoma, USA
[2] University of Rhode Island, USA
Email:[1] sabour@uw.edu , [2] hendawi@uri.edu, [3] mhali@uw.edu

ABSTRACT

This paper presents the Quran Speech Recognition (QRSR) system, achieving alignment and classifi- cation accuracies up to 96%. The system is designed to advance Arabic Automatic Speech Recognition (ASR) and Text-to-Speech (TTS) by focusing on the Arabic diacritic-annotated text. We address the limitations of existing Arabic ASR systems and introduce the Fuzzy Text Alignment and Rule-based Classifier (FTARC) for segmenting audio files and aligning text. The FuzTPI algorithm is integrated with Machine Learning models like Naive Bayes, Support Vector Machine, and Random Forest. This research aims to generalize the findings for broader Arabic text and contribute to an expanded audio dataset, thereby enhancing Arabic NLP and speech recognition capabilities.

Keywords: *The Quranic Arabic Annotated text, Machine Learning, Classification algorithms, Audio segmentation, Text-audio alignment, Speech Recognition.*

1. INTRODUCTION

Speech recognition is an essential component of Natural Language Processing (NLP) that converts spoken language audio into written [31]. On the other hand, Text-to-Speech (TTS) synthesizes spoken language from written text [9]. These speech components involve audio segmentation, where the soundtrack is split into phonemes, words, and sentences [32]. Additionally, audio-to-text alignment aims to synchronize the audio files with the corresponding text, creating a reliable audio-text dataset [3]. Various models and systems have shown promising results in the domain of Arabic speech recognition; however, their focus does not encompass diacritics. This limitation is primarily attributed to the lack of accessible datasets containing diacritized Arabic texts. Hence, the primary motivation behind our research endeavors is to address this challenge and explore potential solutions.

This paper introduces the Quran Speech Recognition (QRSR) system. Its name is

derived from the Qur'an, as it utilizes audio recitations of the Qur'an in Arabic as the foundation for its operations. The QRSR focuses on enhancing Arabic Automatic Speech Recognition (ASR) and TTS through the preparation of a diacritic Arabic text-audio dataset. The idea revolves around using the VAD (Voice Activity Detection) algorithm to divide audio files based on areas of silence [25]. The content of these audio files is then processed using available Speech Recognition systems. The available Arabic ASR systems cannot handle more than the second level of Arabic (see Figure 1). As shown in Figure 1, existing speech recognition systems (Amazon Transcribe, Google Speech-to-Text, and MS Azure Speech-to-text) produce erroneous results and fail to support diacritical characters.

After obtaining ASR transcripts, alignment is done with fifth-level Arabic text that includes diacritical marks. However, comparing texts with different levels of Arabic and containing errors poses challenges for the classification process (determining the position of sound in relation to the text) and the alignment process (identifying each audio file and its affiliation with the text). The goal is to provide Audio files aligned with Arabic diacritic- annotated text that supports Speech Recognition and Text-to-Speech (TTS) systems. This support aims to handle Arabic diacritic-annotated text, which is not addressed by existing Arabic ASR systems. To address these challenges, we proposed a Fuzzy Text Alignment and Rule-based Classifier (FTARC) approach. This approach offers a practical solution for segmenting audio files, aligning texts, and classifying based on predefined labels. This approach achieves an accuracy rate of up to 90.78% according to our testbed. To further improve accuracy, we integrate the FuzTPI algorithm, initially used in FTARC, with Machine Learning (ML) models. We examined the effectiveness of several Machine Learning (ML) algorithms, including Naive Bayes [15], Support Vector Machine (SVM) [20], and Random Forest [27], for enhancing the classification of Arabic texts. Integrating FuzTPI with ML models significantly enhances the efficiency of the alignment-driven classification process. Among the four models implemented, the FuzTPI- Random Forest model achieves the highest accuracy, reaching 96%.

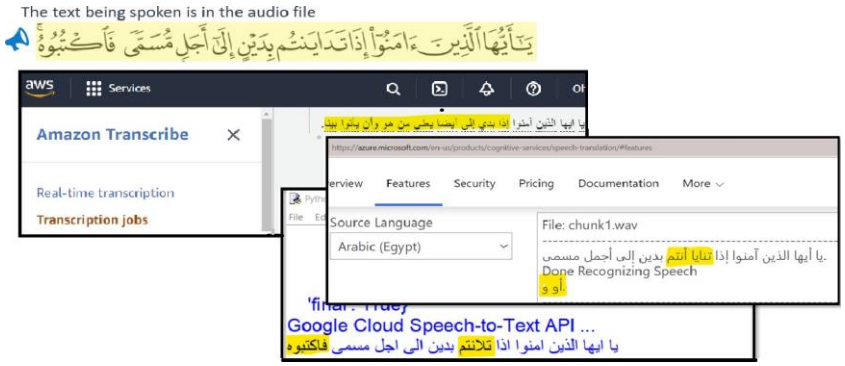


Figure 1: Errors and lack of support for diacritical characters in Arabic speech recognition.

The proposed approach, which combines fuzzy text alignment, and integration with ML models, aims to improve Speech Recognition and TTS systems to support the Arabic diacritic-annotated text. In this part of the research, the Qur'an serves

as a case study, utilizing the availability of Quranic diacritic-annotated text and audio recitations. However, the generalization of the work to the general Arabic text is one of our goals.

The contributions of this paper can be summarized as follows:

- Introduction of the FuzTPI algorithm, which enhances ML models and improves the alignment-driven classification process in Arabic speech recognition.
- Development of the Quran Speech Recognition (QRSR) system, utilizing Quran audio and text to advance Arabic speech recognition and contribute to natural language processing (NLP) systems.
- Training and fine-tuning classification algorithms, including ML models, to classify the un-diacritic Automatic Speech Recognition (ASR) transcript into subsentences aligned with diacritic-annotated Arabic sentences.
- Alignment of segmented audio with diacritic-annotated Arabic text representations using existing Speech Recognition systems, addressing the challenge of diacritic-aware recognition not being supported by current Arabic Speech Recognition systems.
- Application of FuzTPI-Random Forest to boost the performance of the QRSR system, achieving an accuracy rate of 96%.
- Contribution to the development of an expanded textual audio dataset, offering potential enhancements for Arabic speech recognition on a broader scale.

The rest of this paper is structured as follows: Section 2 reviews relevant work in three main areas. Section 3 outlines our QRSR system. Section 4 introduces our FTARC method. Section 5 discusses the fusion of algorithms for improved speech classification. Section 6 presents performance evaluations. Finally, Section 7 offers conclusions and future directions.

2. RELATED WORK

This section surveys research in three key areas: Arabic text classification, Arabic speech recognition, and Quranic speech recognition. Each area presents unique challenges and advancements that are relevant to our study.

a. State-of-the-Art Arabic Speech Recognition

The study of Aldarmaki and Ghannam (2023) [2] investigates diacritic recognition in Arabic ASR systems. It addresses the problem of ASR models not producing full diacritization due to the absence of diacritical marks in existing speech corpora. This research is relevant to our work on diacritic recognition. However, our approach incorporates the Fuzzy Text Alignment and Rule-based Classifier (FTARC), targeting more specific challenges like text-audio alignment.

Research of Humayun et al. (2023) [12] focuses on dialect classification in Arabic speech. This research is relevant to our work due to its focus on classifier combination. Yet, our research narrows down to aligning Arabic diacritic-annotated text, making it more specialized.

The paper Qasim and Abdulbaqi (2022) [23] reviews Arabic speech recognition, emphasizing the need for advanced Arabic ASR systems. Our research contributes by specifically addressing text-audio alignment and diacritic annotation.

b. Speech Recognition of the Quranic Research

In Larbi (2013) [18], researchers focused on phonemic search in the Quran for Hafs's Riwayh. The methodology we employ yields a faster response time and higher accuracy, thanks to our FTARC approach.
In Muhammad et al. (2010) [22], they propose a solution to measure recitation accuracy. Their dataset and metrics differ from ours, as we employ multiple classifiers and algorithms for a more comprehensive evaluation.

c. Arabic Text Classification and Speech Recognition

Research AboAlnaser (2019) [1] focuses on text classification methods for Arabic documents. Our approach is more targeted and might benefit from the standardization efforts they discuss.
The research Wahdan et al. (2020) [29] presents a systematic review of Arabic text classification. Our research targets unique text-versus-text classification, serving as a guidepost for neural networks and evaluation metrics.
Research Bhogale et al. (2023) [6] proposes a framework for aligning long audio segments with their corresponding transcriptions. We specialize in short segments with diacritical marks, offering a distinct methodology.
Research Sundus et al. (2019) [28] employs deep learning for Arabic text classification. While it shares classification as a common theme, our research targets nuanced textual comparisons and therefore requires a distinct methodology.

3. QRSR SYSTEM: METHODOLOGY AND KEY OPERATIONS

The QRSR System objective is to segment and align Arabic audio with diacritic-annotated text. This process aims to create a substantial dataset of diacritics in Arabic audio-text pairs. The resulting dataset will serve as a valuable resource to enhance and improve diacritic-aware ASR and TTS systems. The system is comprised of six modules, as depicted in Figure 2.
We start by preparing our dataset that has *diacritic-annotated text* (at the 5^{th} level of Arabic annotation) along with its corresponding audio recitations as *audio files*. This dataset serves as a resource for model training and evaluation purposes. Audio files are usually of long duration, which hinders processing and aligning based on sentences. The audio files are segmented into smaller audio clips based on detected periods of silence by using *Voice Activity Detection*, or *VAD*, algorithm [10]. Each audio clip can represent a sentence, part of a sentence, or several sentences the reciter connected together in one connected recitation. Each audio clip is processed by one of the available ASR systems, specifically the Google Cloud Speech-to-Text API, to generate a textual transcript referred to as "ASR Transcript". The *ASR transcript* produced by current ASR services is featured by two things: (a) the ASR transcript without diacritic letters and (b) has a low degree of accuracy and shows several errors in the Arabic speech (as shown in figure 1). We align the generated ASR transcript to the original text (i.e., *diacritic-annotated text)* that we have for the audio files. A set of models is trained to determine the alignment between the diacritic-annotated sentence and the ASR transcript. The architecture of the QRSR System is illustrated in Figure 2 and is

discussed in the following subsections.

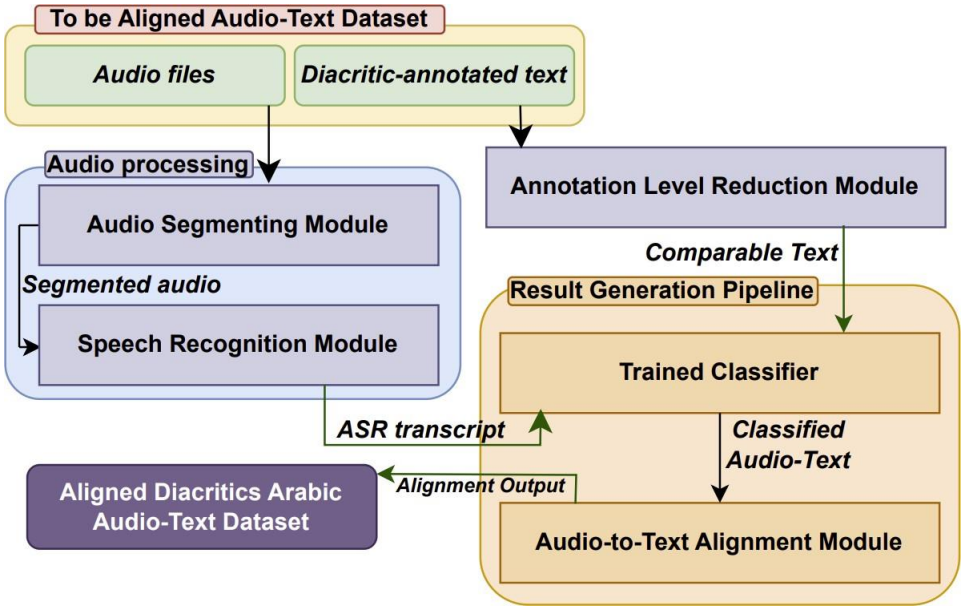


Figure 2: The architecture of Arabic speech to diacritic-annotated text alignment system.

The upcoming six subsections will provide detailed explanations of these modules. However, the system's overall concept can be summarized as follows.

3.1 To be Aligned Audio-Text Dataset

Arabic audio files and their corresponding texts are collected to establish a dataset. Quranic texts and phonetics are chosen for their accurately written diacritical letters and precise pronunciation. The dataset comprises 9,596 audio files and records. All audio files utilized in the dataset are high-quality recordings without any noise interference. The noise interference is not within the scope of your study. The label indicates the alignment of the ASR transcript of the audio file with the verse. The labels specify the position of the reading within the verse, such as the beginning, middle, or end. The dataset is divided into a 70% training set (6,715 records), a 15% test set (1,442 records), and a 15% validation set (1, 439). We used the validation dataset to fine-tune the settings that control our models, which we refer to as hyperparameters. The process of adjusting these hyperparameters is important in achieving the best performance from our models. The division of the dataset is performed separately for each label to ensure a fair representation of each label in both the training and test data. This approach ensures that each label is adequately represented in both sets.

Table 1: Labels alignment and unalignment cases between audio and text.

#	Label	Explanation
1	FAFV	**FullAudioFullVerse:** The entire audio file corresponds to a complete verse.
2	VPOA	**VersePartOfAudio:** The audio contains a complete verse along with other verses.

3	VPOEA	**VersePartOfEndAudio:** The complete verse is at the end of the audio, which also includes other verses before it.
4	PVLP	**PartialVerseLastPartRecited:** The audio contains a portion of the verse, specifically the last part of it.
5	PVNF	**PartialVerseNotFinished:** The audio contains a portion of the verse but does not include the last part of it.
6	Uncertain	**Uncertain:** The text extracted from the audio is unclear regarding its association with a verse. However, there is still a low similarity ratio.
7	ANR	**AudioNotRecognized:** The audio file content is unidentifiable and therefore excluded from the classification process. It was not included in the dataset from the beginning.
8	UA	**UnspecifiedAudio:** The audio could not be linked to the verse.
9	ANAV	**AudioNotAVerse:** This audio does not represent a specific verse. It may contain opening or closing words unrelated to any verse. As a result, it is excluded from the classification process since it requires comparison with all verses, rather than a single classification.
10	MA	**MissingAyah:** We could not establish a link between this verse and any audio files.

The sentences or verses display variations in length. Longer sentences may be read in parts, where one long sentence corresponds to multiple audio clips separated by periods of silence. On the other hand, shorter sentences may be recited together without periods of silence, resulting in one audio clip. This variation introduces challenges in segmenting and aligning the audio at the sentence level. To address this, during the dataset preparation phase, we carefully label each audio clip based on whether it contains a complete sentence, a partial sentence, multiple sentences, or falls into other categories. This labeling process involves a manual review of both the audio clips and their corresponding text. The resulting labels are summarized in Table 1.

To gain insights into the label representation in the collected dataset, we assess their distribution, and a bar chart is given in Figure 3. The x-axis represents the class labels and the y-axis represents the frequency of each label in the dataset.

The preparation of the training and evaluation dataset involves a semi-automatic process of comparing and labeling the ASR transcript with the comparable text. The system streamlines text processing and generates ASR transcripts, while also assisting with straightforward label assignments. An example of a straightforward label is “AudioNotRecognized,” which indicates that the content of an audio file cannot be recognized. For the remaining labels, manual data labeling is conducted.

During manual data labeling, we encountered 10 labels, which are summarized in Table 1. These labels demonstrate the alignment of the text with the corresponding

audio content. To create the training and evaluation data set, we select some sentences and audio clips from the full data set such that the labels are equally represented in the training data set. That is to prevent the models from being more inclined to predict the majority label than the minority labels [4].

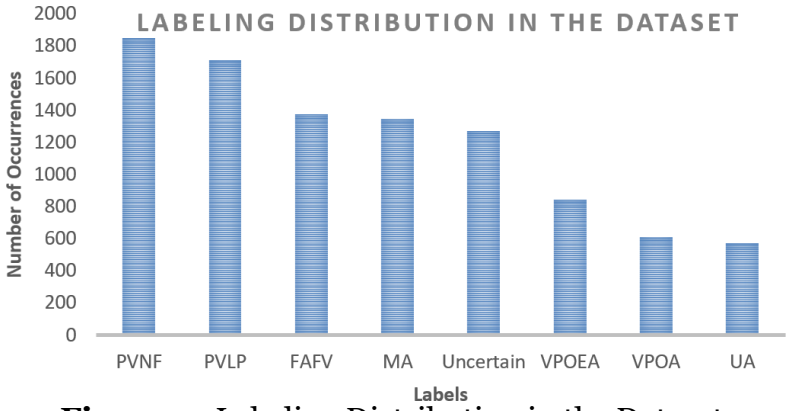


Figure 3: Labeling Distribution in the Dataset.

3.2 Audio Segmentation Module

We obtain the audio files of the Quran recitation from various sources on the web. These audio files are recorded over the years for the purpose of listening to the Quran recitations and not for the purpose of computer-aided audio processing. The widely-available audio files contain the recitations of the chapters of the Quran. Such lengthy audio files need to be segmented into smaller audio clips that can be matched and aligned to sentences of the corresponding text. The goal of the Audio Segmentation Module is to divide the audio file into smaller clips based on detected periods of silence in the audio track.

The segmentation process is accomplished through *Voice Activity Detection* (*VAD)* algorithms that detect periods of silence [10]. In this module, we use a segmentation algorithm [21] that distinguishes between silence and non-silence segments by analyzing the amplitude levels. The algorithm dynamically utilizes a variable amplitude threshold value to differentiate silence from non-silence. This segmentation algorithm is particularly suitable for audio with natural pauses, as observed in Quranic recitation. The output of this module is segmented into small audio clips obtained by splitting long audio files according to periods of silence.

3.3 Speech Recognition Module

We utilize speech recognition technologies, specifically the Google Speech-to-Text API, to convert audio files into text only, and we refer to the output as the ASR transcript. The ASR transcript that is generated by current systems is not perfect as it lacks diacritics and contains conversion errors. However, it is considered one step in the direction of converting speech into text. This ASR transcript is then used to create the ground truth and is also sent to the Audio-To-Text Alignment module for additional processing. The workflow of the process is depicted in Figure 2.

3.4 Annotation Level Reduction Module

The Quranic text is written in annotation fifth-level Arabic, while the ASR

transcript is in second-level Arabic (as shown in table 2). The difference in writing annotation levels poses challenges to comparing these texts. Additionally, the Quranic text follows the Ottoman Drawing Style (ODS) in its writing. This ODS style results in variations in certain characters compared to the standard Arabic writing. To address this, we developed a module that reduces the annotation level for comparisons. The module generates a *comparable text*, a second- level Arabic text derived from the Quranic diacritic-annotated text. By using the *comparable text*, it becomes easier to compare the original diacritic-annotated text (after reducing its annotation level from the fifth level to the second level) with the ASR transcript at the same level of annotation.

Table 2: Levels of details and annotations in the Arabic writing styles.

a) First drawing of writing.	b) Dotted letters.	c) Diacritic letters (tashkeel).	d) Pronunciation marks.	e) Tajwid marks

Text processing methods are employed to adjust the annotation level for improved comparability. The following steps are taken:

- **Tokenization:** The text was divided into individual sentences/verses.
- **Text cleaning:** The diacritical letters are removed and adjustments are made to the Ottoman text to match standard texts.
- **Noise removal:** Unpronounceable signs, such as verse numbering, are removed from the texts for better suitability in comparisons.

The algorithm 1 addresses the challenge of comparing diacritic-annotated Quranic text written in annotation fifth-level Arabic with ASR transcripts written in second-level Arabic. To facilitate this comparison, a module has been developed that reduces the annotation level for better comparability. The algorithm consists of four steps.

Step 1: Transforming ODS to Classical Arabic Fifth Level In this step, the algorithm converts the Quranic text from Ottoman Drawing Style (ODS) to Classical Arabic (CA) fifth level. The algorithm iterates through each character in the input text and checks if it exists in the ArabicAnnotationLevelMap, which holds the mapping between ODS letters and CA fifth-level letters. If a match is found, the corresponding CA fifth-level letter is appended to the ConvertedCAText. Otherwise, the character is appended as it is. This process ensures that the text is transformed to the desired annotation level.

Step 2: Tokenization The algorithm tokenizes the diacritic-annotated text into individual verses. It uses the DiacriticSentenceTokenizer function to split the text into tokenized diacritic sentences, which are stored in the TokenizedDiacriticSentence list.

Step 3: Removing Diacritical Letters In this step, the algorithm removes diacritical letters from the tokenized sentences. It iterates through each sentence in the TokenizedDiacriticSentence list and removes any characters present in the ArabicDiacriticMarks list. This ensures that only the base letters remain, making the text suitable for comparison. The resulting cleaned sentences are stored in the TokenizedSentence list.

Step 4: Noise Removal The algorithm performs noise removal to improve the suitability of the text for comparison. It iterates through each sentence in the TokenizedSentence list and applies a series of regular expression-based cleanup patterns. These patterns target specific noise elements present in the text, such as verse numbering, and remove them. The cleaned sentences are stored in the ComparableSentence list.
Finally, the algorithm outputs the ComparableSentences list, which contains the reduced, cleaned, and com- parable text in the second-level Arabic format. This output can be easily compared with the ASR transcript for further analysis or evaluation.

Algorithm 1: Annotation Level Reduction Algorithm

```
Input  : ArabicFifthLevelDiacriticText: Diacritic-annotated text at the fifth level
Output: ComparableSentences: List of comparable text reduced to the second level
//Initialization:
ArabicAnnotationLevelMap ← Structure that holds Ottoman Drawing Style (ODS) letters
ArabicDiacriticMarks ← List of Arabic diacritic marks
ODSPatternCleanupExpressions ← List of regular expressions for cleaning ODS text
//Processing :
foreach text in ArabicFifthLevelDiacriticText
    //Step 1: Transform ODS to CA Fifth Level
    foreach character in text
        if character exists in ArabicAnnotationLevelMap then
            Append corresponding CA fifth-level letter to ConvertedCAText
        else
            Append character as it is to ConvertedCAText
    //Step 2: Tokenization
    TokenizedDiacriticSentence ← DiacriticSentenceTokenizer(ConvertedCAText)
    //Step 3: Remove diacritical letters
    foreach mark in ArabicDiacriticMarks
        // Replace all occurrences of mark in TokenizedSentence with an empty string
        TokenizedSentence ← TokenizedDiacriticSentence.replace(mark, ")
    //Step 4: Noise removal
    foreach pattern in ODSPatternCleanupExpressions
        // Remove all matches of pattern from TokenizedSentence using regular expressions
        ComparableSentence ← RegExp.Replace(pattern, ", TokenizedSentence)
    Append ComparableSentence to ComparableSentences
Output:
    Return ComparableSentences
```

3.5 Trained Classifier

The classification process involves categorizing and labeling the data by analyzing the alignment patterns between the ASR transcript, which represents the written form of the speech, and the comparable text, which represents the simple form of the diacritic annotated text. Ultimately, the classification is applied to the audio and diacritic annotated text, leveraging the alignment patterns between the ASR transcript and the comparable text. We aim for the classifier model to learn from the provided data, including the erroneous text, and make predictions based on it. Hence, we refrained from correcting the linguistic errors present in the ASR transcript. The classifier is expected to account for the presence of errors and still perform effectively in our QRSR system. Arabic ASR transcripts often contain errors in the inferred text from audio. Table 3 provides examples of challenges

encountered during the classification process caused by the ASR errors.
During the inference phase, the data is obtained from the comparable text from the Annotation Level Re- duction Module and the ASR transcript from the Speech Recognition Module. Data is classified based on the labels provided in table 1. The first five labels in table 1 demonstrate an alignment relationship between the audio and text, showcasing different types of alignment. These labels represent various alignment relations, such as the full audio file corresponding to a full verse (FAFV) and the verse being a part of the audio file (VPOA), among others. Figure 4 provides a visual representation of the first five labels.

Table 3: Some Cases that Reflect the Challenges in Text Classification.

Text (Diacritics omitted)	ASR transcript	Class
أوءاباؤنا الأولون	اواب جاءنا الاولون	1
Note: Fragmented and misspelled words make the text difficult to detect.		
قل إنما أنا بشر مثلكم يوحى إلي أنما إلهكم إله وحد فاستقيموا إليه واستغفروه وويل للمشركين	بس تلقيم اليه واستغفروه	6
Note: The text contains both correct and incorrect parts, and it is essential to clarify that it does not represent the beginning or end of the full text.		
لهم من فوقهم ظلل من النار ومن تحتهم ظلل ذلك يخوف الله به عباده يعباد فاتقون	ياعبادي فتتكون	7
Note: The text lacks coherence, even though an Arabic speaker might expect it at the end of the text.		

The last five labels in table 1 demonstrate different cases for an unalignment of the text and the ASR transcript. These labels involve cases like audio content not recognized (ANR), audio not corresponding to a verse (ANAV), and others.

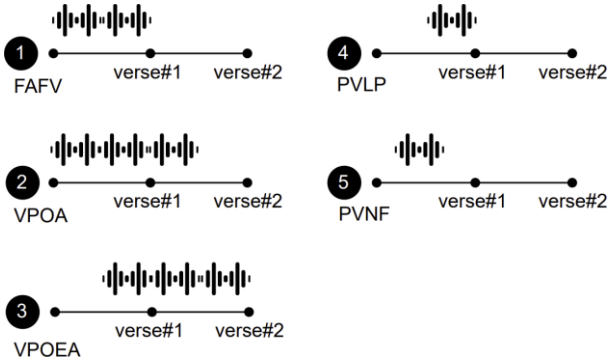


Figure 4: Visual representation of alignment relationships between audio and text.

3.6 Audio-to-Text Alignment Module

The module has two purposes: aligning Arabic speech to diacritic-annotated text and preparing a dataset for enhancing Arabic Speech Recognition. The inputs are the ASR transcript, the comparable text, and the predicted labels. These inputs are sent from the Trained classifier.

The Alignment Module retrieves the actual audio files corresponding to the ASR transcript and the Arabic diacritical text corresponding to the comparable text. If necessary, the audio files are merged to match a full sentence. Finally, the module combines the audio files with their corresponding diacritical text, resulting in a diacritic Arabic audio-text dataset.

Algorithm 2: Audio-to-Text Alignment Algorithm

Inputs : ASRTranscriptList, ComparableTextList
Output: Diacritic Arabic audio-text dataset

```
Initialize empty lists: audio_list, text_list
foreach ASRTranscript in ASRTranscriptList and ComparableText in ComparableTextList do
    label ← predictLabels(ASRTranscript, ComparableText)
    audio_file ← fetchAudioFiles(ASRTranscript)
    diacritical_text ← fetchDiacriticalText(ComparableText)
    push diacritical_text into text_list
    audio_file ← combineAudioFiles(audio_file)
    push audio_file into audio_list
Return mergeAudioAndText(audio_list, text_list)
```

The Audio-to-Text Alignment Algorithm 2 consists of several steps to collect system inputs, align audio and text, and generate the final output.

Step 1: Algorithm Inputs The algorithm takes two inputs: *ASRTranscriptList* and *ComparableTextList*. *ASRTranscriptList* represents the ASR transcript obtained from the Speech Recognition Module, while *Compara- bleTextList* represents the comparable text obtained from the Annotation Level Reduction Module. These inputs serve as the foundation for aligning audio and text.

Step 2: Label Prediction The algorithm iterates through each pair of *ASRTranscript* and *ComparableText* in their respective lists. For each pair, it predicts the labels by calling the *predictLabels* function, which utilizes the Classification Module.

Step 3: Retrieving Audio Files and Diacritical Text The algorithm retrieves the actual audio files corresponding to each *ASRTranscript* by calling the *fetchAudioFiles* function. Similarly, it retrieves the Arabic diacritical text corresponding to each *ComparableText* by calling the *fetchDiacriticalText* function.

Step 4: Building Audio and Text Lists The algorithm initializes two empty lists: *audio list* and *text list*. For each pair of *ASRTranscript* and *ComparableText*, it pushes the corresponding diacritical text into the *text list*. Similarly, it combines the audio files by calling the *combineAudioFiles* function, which merges the audio files if necessary, and pushes the resulting *audio file* into the *audio list*.

Step 5: Generating the Diacritic Arabic Audio-Text Dataset The algorithm returns the diacritic Arabic audio-text dataset by calling the *mergeAudioAndText* function, which takes the *audio list* and *text list* as inputs. This function combines the audio files with their corresponding diacritical text, resulting in a diacritic Arabic audio-text dataset ready for further analysis, training, or evaluation. By aligning audio and text, the algorithm facilitates various downstream tasks and applications, such as training speech recognition models, improving language understanding, and enabling audio search and retrieval.

4. FUZZY TEXT ALIGNMENT AND RULE-BASED CLASSIFIER (FTARC)

We have developed a Fuzzy Text Alignment and Rule-based Classifier (FTARC). FTARC is composed of two algorithms. The first algorithm is a Fuzzy Text Position Inference (FuzTPI) Algorithm 3. This algorithm utilizes fuzzy matching techniques to generate

numerical indicators. These numerical indicators provide insights into alignment patterns and the presence of text segments within each other. The second algorithm is a rule-based classifier, which leveraging from the Fuzzy numerical indicators to classify the texts. We aim to improve the accuracy of the classification process by utilizing FTARC. In this section, we will discuss two key components: the FuzTPI Algorithm and the Rule-based Classifier. We will explore how these algorithms contribute to the task at hand. Furthermore, we will present and analyze the results obtained from FTARC.

a. Fuzzy Text Position Inference (FuzTPI) Algorithm

The proposed FuzTPI algorithm facilitates the comparison of two texts of varying length and content by leveraging fuzzy matching techniques [7, 11].

Algorithm 3: Fuzzy Text Position Inference Algorithm.

Input: ASRTxt, ArTxt

```
Specify the alignment indicator
    alignmentIndicator ← (length(text1) >length(text2)) ? 1:2
Determine the longest and shortest text
    longerText ← (length(ASRTxt) >length(ArTxt)) ? ASRTxt : ArTxt
    shorterText ← (length(ASRTxt) >length(ArTxt)) ? ArTxt : ASRTxt
End-of-text matching ratio
    shortenedLongerText ← longerText[length(shorterText):]
    EndOfTextMatching ← Fuzzy(shortenedLongerText, shorterText)
Matching percentage of the beginning of the text
    shortenedLongerText ← longerText[:-length(shorterText)]
    BeginOfTextMatching ← Fuzzy(shortenedLongerText, shorterText)
Middle text matching ratio
    trimmedLongerText ← CutToMatchTheMiddle(longText, length(shorterText))
    MiddleOfTextMatching ← Fuzzy(trimmedLongerText, shorterText)
Last Text Part in ASR Transcript Verification
    lastPart ← " ".join(ArTxt.split()[-n:]) or ArTxt
    IsLastPartInASR ← PartialFuzzy (lastPart, ASRTxt) >threshold
Measure fuzzy text matching
    FuzzyTextMatching ← Fuzzy(ASRTxt, ArTxt)
Measure fuzzy partial matching
    FuzzyPartialMatching ← PartialFuzzy(longerText, shorterText)
Count occurrences of the short text in the long text
    OccurrencesCount ← CountOccurrences(longerText, shorterText)
Return fuzzy numbers
    return All fuzzy numbers that have been calculated
```

The algorithm provides numerical indicators that reflect the text alignment patterns. This algorithm depends on two main functions to determine these numerical indicators, fuzzy full-text matching and fuzzy partial-text matching.

The *fuzzy full-text matching* function, denoted as *Fuzzy*(longerText, shorterText), can be defined as equation (1). This equation calculates the similarity between a longer text and a shorter text [16]. It considers every possible pair of characters between the two texts and computes the membership value (μ) indicating their degree of similarity. The numerator sums up these membership values for all character pairs, capturing the overall similarity. The denominator normalizes the result by dividing the numerator by the maximum length between the longer and shorter texts. This normalization accounts for differences in text length when comparing them.

The *fuzzy partial-text matching* function, denoted as PartialFuzzy(longerText, shorterText), can be defined as equation (2). This equation focuses on the similarity between a longer text and a shorter text when the shorter text is a part of the longer text. Similar to the equation (1), it calculates the membership value (μ) for each pair of characters between the longer and shorter texts. The numerator sums up these membership values, indicating the overall similarity. However, in this case, the denominator is the length of the shorter text itself. This normalization allows for a direct comparison of the similarity between the shorter text and the corresponding part of the longer text [16].

$$Fuzzy(longerText, shorterText) = \frac{\sum_{i=1}^{|shorterText|} \sum_{j=1}^{|longerText|} \mu(longerText[j], shorterText[i])}{\max(|longerText|, |shorterText|)} \quad (1)$$

$$PartialFuzzy(longerText, shorterText) = \frac{\sum_{i=1}^{|shorterText|} \sum_{j=1}^{|longerText|} \mu(longerText[j], shorterText[i])}{|shorterText|} \quad (2)$$

where:

- longerText[j] represents the j-th character in longerText.
- shorterText[i] represents the i-th character in shorterText.
- μ(longerText[j], shorterText[i]) represents the membership value indicating the degree of similarity be- tween longerText[j] and shorterText[i].
- |longerText| represents the total number of characters in longerText.
- |shorterText| represents the total number of characters in shorterText.

By using these equations (1, 2), the FuzTPI 3 provides numerical indicators that reflect the alignment patterns between texts, capturing the degree of similarity between them. Initially, the algorithm identifies the longest text and determines the alignment indicator, which determines whether the text is part of the audio or vice versa. The algorithm measures the matching degree of the shorter text at the beginning, end, and middle of the longer text, providing insights into the relative positioning of the texts. Furthermore, the algorithm determines the full and partial matching degree between the two texts. Additionally, the algorithm counts the number of occurrences of the shorter text within the longer text, providing further information about their presence and potential repetitions. By offering these insights the fuzzy algorithm equips the classifier with essential alignment data.

b. The Rule-based Classifier

The rule-based classifier starts by retrieving a list of Arabic sentences, along with their corresponding audio files and ASR transcripts. The system then initializes relevant variables and iterates through the lists. To ensure consistency, it applies text normalization techniques to the input texts. Additionally, it utilizes the numerical outputs generated by the Fuzzy Algorithms 3. Based on predefined cases outlined in Table 4, the rule-based classifier classifies the data using these numbers.

The rule-based classifier, enhances the classification process, enabling the generation of aligned diacritized Arabic texts and audio files. The purpose of this classifier is to assign appropriate classifications to the data based on predefined rules and numerical fuzzy outputs generated by the FuzTPI Algorithm 3.

Initially, the classifier retrieves a list of Arabic sentences along with their corresponding audio files and ASR transcripts. To ensure consistency, the system applies text normalization techniques to the input texts. The first set of conditions on table 4 focuses

on evaluating the availability of the ASR transcript and the degree of similarity between the two texts. If the ASR transcript is not available or the full text and transcript exhibit a high degree of similarity, the classifier returns specific labels indicating the alignment status. Subsequent conditions take into account the alignment of partial text, considering factors such as the alignment indicator and the presence of the last part of the text in the transcript. Additional conditions build upon the previous classifications and consider the degree of similarity between adjacent texts and ASR transcripts. These conditions aim to capture nuances in the alignment process and ensure accurate classification.

Table 4: Rule-Based Conditions and Return Values.

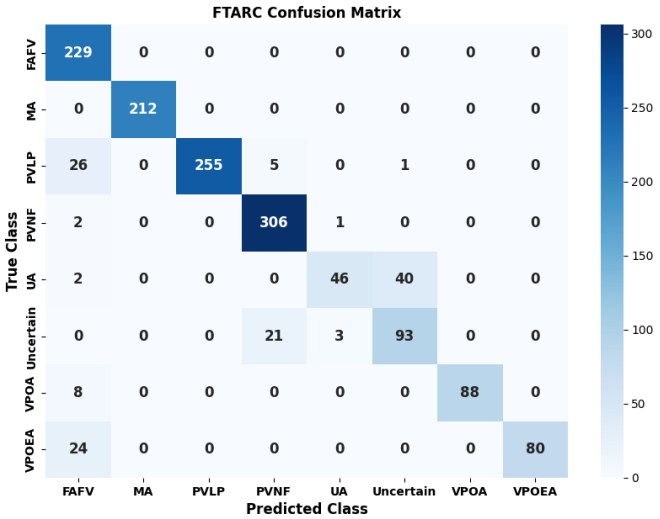
Condition	Return Value
ASRText is None	ANR
FuzzyFullTextMatching > Threshold	FAFV
FuzzyPartialMatching > Threshold and alignmentIndicator == 1 and IsLastPartInASR	VPOEA
FuzzyPartialMatching > Threshold and alignmentIndicator == 1 and not IsLastPartInASR	VSNF
FuzzyPartialMatching > Threshold and alignmentIndicator == 2 and IsLastPartInASR	PVLP
FuzzyPartialMatching > Threshold and alignmentIndicator == 2 and not IsLastPartInASR	PVNF
IsDataClassifiedBefore and NextTextFuzzyPartialMatching > Threshold	Uncertain
IsDataClassifiedBefore and PreTextFuzzyPartialMatching > Threshold	StepBackPreText()
IsDataClassifiedBefore and PreASRFuzzyPartialMatching > Threshold	StepBackPreASR()
IsDataClassifiedBefore and IsPreASRPartOfText	Uncertain
IsDataClassifiedBefore and PreTextFuzzyPartialMatching > Threshold and IsLastPartInASR	PVLP
IsDataClassifiedBefore and PreTextFuzzyPartialMatching > Threshold and not IsLastPartInASR	PVNF
IsDataClassifiedBefore and PreTextFuzzyPartialMatching < Threshold	IncreaseSkippedASRCount(), UA
not IsDataClassifiedBefore and FuzzyPartialMatching < Threshold	IncreaseSkippedASRCount(), ANAV
IsASRLimitExceeded	MA

The final set of conditions handles cases where the ASR transcript is ignored beyond predefined limits or when the alignment falls below the specified threshold. These conditions further refine the classification process, accounting for specific alignment scenarios. Based on the classifications received from the rule-based classifier, the system merges the audio files into complete sentences that align with the list of Arabic sentences. As a result, the system generates a structured output consisting of diacritized Arabic texts accompanied by their corresponding audio files. The threshold used to accept the agreement between ASR transcripts and Arabic text plays an important role in the FTARC algorithm. Through experimentation on our dataset, we observed that different threshold values affect the alignment accuracy differently. When the threshold is set above 80%, the algorithm tends to lose sentences that are actually almost complete and similar. This is due to ASR transcript errors, causing the convergence percentage to fall below 80%. On the other hand, when the threshold is set at 70% or lower, the algorithm starts to agree on similar sentences, but some words are missing, leading to incomplete alignments. After conducting extensive experiments, we found that a threshold of 75% strikes the right balance between capturing similar sentences and maintaining

completeness.

c. FTARC Performance

Figure 5: Confusion Matrix of FTARC Algorithm.



The performance evaluation of the FTARC showed an accuracy of 90.77%, precision of 91.96%, recall of 90.77%, and an F1 score of 90.59%. The high values for these metrics indicate that the FTARC achieved a good level of accuracy in its classification results when compared to the manually entered results.

Analyzing the confusion matrix in figure 5, it is evident that the FTARC classifier exhibited accuracy in recognizing the majority of the classes (which appeared in the diagonal elements). It demonstrated moderate accuracy in recognizing UA. This result reflects its accuracy, which is 90.7%. These findings highlight the algorithm's overall success in classification, while also indicating potential areas for improvement in accurately distinguishing between certain classifications.

5. FUSION OF FUZTPI ALGORITHM AND ML MODELS FOR IMPROVED ARABIC SPEECH CLASSIFICATION

The high accuracy of FTARC is attributed to the FuzTPI Algorithm, which effectively addresses typographical errors in the Arabic ASR transcript by generating fuzzy numbers that indicate the degree of text present in the second text. While ML models outshine rule-based models in their generalizability and applicability across different classification tasks, rule-based models offer domain-specific solutions based on predefined rules and conditions. In this section, we aim to leverage the strengths of both approaches by integrating the FuzTPI Algorithm with ML models. This integration involves generating fuzzy numbers using FuzTPI for the textual data and allowing ML models to learn from these fuzzy numbers and their corresponding taxonomy, bypassing the need to directly handle the ASR transcript with its errors. Figure 6 illustrates the integration of the FuzTPI algorithm with the ML model. FuzTPI takes the ASR transcript and Arabic comparable text as input, generating fuzzy numbers as output. The ML model solely uses fuzzy numbers as input and produces labeling predictions.

This integration helps safeguard the ML model from unexpected errors in the Arabic ASR transcript. Integration is done through a method for serializing and deserializing called pickle [26]. This means the FuzTPI is "bundled" with the ML model in a way that it can be deployed as one unit. By integrating FuzTPI with the ML model, we eliminate the need to program or manage the fuzzy inference engine separately. Everything is handled within the FuzTPI-ML Integration Unit.

Text classification has gained significant attention due to using it in a wide range of applications [1, 14, 17]. In the classification task, ML models are widely utilized in various applications [1, 17]. Popular ML models for text classification tasks such as Naive Bayes (NB), Support Vector Machine (SVM), and Random Forest (RF). In this part, we will present the results of applying the NB, SVM, and RF models when they learn directly from the output of FuzTPI. By examining the performance of these ML models with fuzzy numbers as input, we aim to assess their effectiveness in enhancing the alignment-driven classification of Arabic speech.

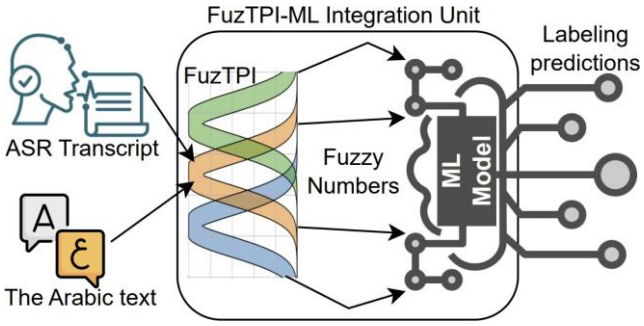


Figure 6: Integration of FuzTPI with ML Model.

a. FuzTPI-Driven Naive Bayes

Naive Bayes is a widely used and versatile classification algorithm known for its simplicity and efficiency [1]. By applying Bayes' theorem, it determines the most probable class for a given instance, making it particularly valuable in scenarios with limited training data [10]. We utilize the MultinomialNB variant of Naive Bayes for text classification tasks [30]. MultinomialNB is particularly effective in handling discrete features such as word frequencies and counts. In our algorithm, we further optimize the performance of MultinomialNB by employing GridSearchCV. The GridSearch is a technique that systematically explores the hyperparameter space to identify the best combination of parameters for the classifier [5]. This allows us to enhance the accuracy and effectiveness of the text classification process.

FuzTPI-Driven Naive Bayes Performance:

The Naive Bayes is trained on the data generated from FuzTPI. The model performance is evaluated on the testing dataset, and it achieved the performance measures: Accuracy Ratio of 55.13%, Precision of 49.24%, Recall of 55.13%, and F1 Score of 50.55%. These metrics assess the algorithm's classification performance across our labels.

The confusion matrix in Figure 7 reveals that the FuzTPI-NB classifier struggles to accurately recognize the "Uncertain" label, as indicated by the scattered predictions across the matrix. Additionally, it completely fails to distinguish the MA class, classifying all instances as PVNF. However, the classifier demonstrates relatively higher accuracy in identifying the remaining classes compared to the average performance. Overall, the FuzTPI-NB classifier achieves an average accuracy of 55%.

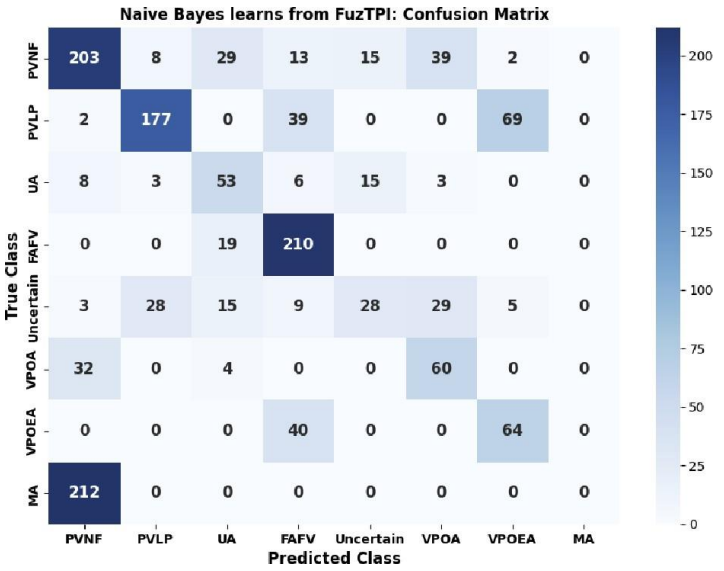


Figure 7: Confusion Matrix of FuzTPI-Driven Naive Bayes Algorithm.

b. FuzTPI-Driven SVM

SVM is a widely used supervised ML algorithm for classification tasks. The main objective of SVM is to maximize the margin between different categories, creating a clear separation between them [1, 17]. SVM identifies support vectors, which play a crucial role in determining the boundaries between classes [17]. SVM can be applied to features derived from textual data, such as word frequencies, n-grams, or other linguistic characteristics. Therefore, we chose SVM to apply to the problem of Arabic diacritic-annotated text. SVM performs well on datasets of small to medium sizes and is particularly suitable for datasets with a moderate number of samples. We utilized TF-IDF (Term Frequency-Inverse Document Frequency) vectorization [13], to transform the text data into numerical features, allowing for analysis and modeling using SVM [19].

FuzTPI-Driven SVM Performance:

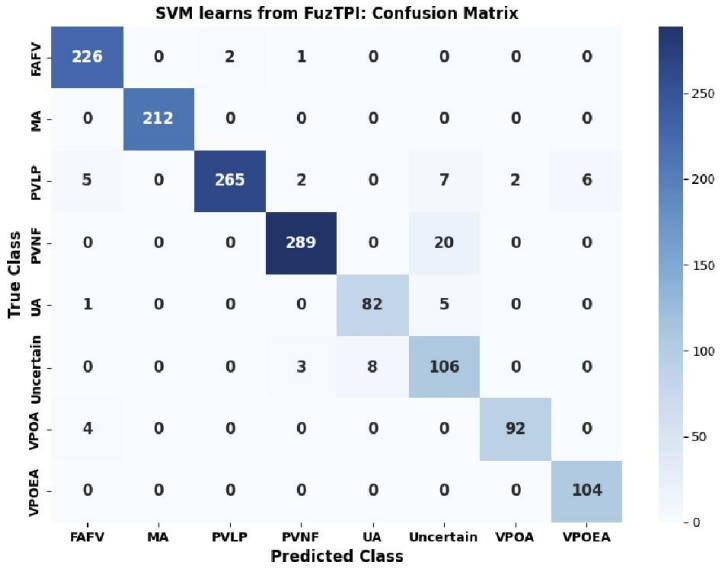


Figure 8: Confusion Matrix of FuzTPI-Driven SVM Algorithm.

The SVM is trained on the data generated from FuzTPI. The mode performance is

evaluated on the testing dataset, and it achieved the performance measures: Accuracy Ratio of 95.42%, Precision of 95.78%, Recall of 95.42%, and F1 Score of 95.51%. These metrics assess the algorithm's classification performance across our labels.
The confusion matrix of the FuzTPI-SVM classifier in figure 8 reveals a distinct diagonal line, indicating the high accuracy of its predictions. However, the classifier encounters challenges in recognizing the "Uncertain" class. It also shows some errors in correctly identifying the PVNF class. On the other hand, the classifier performs well in recognizing the remaining classes, resulting in an accuracy rate exceeding 95.4%.

c. FuzTPI-Driven Random Forest

The Random Forest algorithm is a powerful learning algorithm widely used for classification tasks. It leverages the predictions of multiple decision trees to enhance accuracy [17]. We implemented the Random Forest algorithm, and employed grid search, a technique that systematically explores different parameter combinations. Through grid search, we identified the best parameters that resulted in the optimal configuration for the Random Forest algorithm [24].

FuzTPI-Driven RF Performance:
The RF is trained on the data generated from FuzTPI. The mode performance is evaluated on the testing dataset, and it achieved the performance measures: Accuracy Ratio of 95.90%, Precision of 96.51%, Recall of 95.90%, and F1 Score of 96.02%. These metrics assess the algorithm's classification performance across our labels.
The confusion matrix of the FuzTPI-RF classifier exhibits a diagonal line pattern, indicating its high accuracy in classification. However, the classifier encounters challenges in correctly recognizing the "Uncertain" class, as shown in figure 9. Additionally, it demonstrates some errors in identifying the PVNF and PVLP classes. Nevertheless, the classifier performs well in recognizing the remaining classes, resulting in an accuracy rate approaching 95.9%.

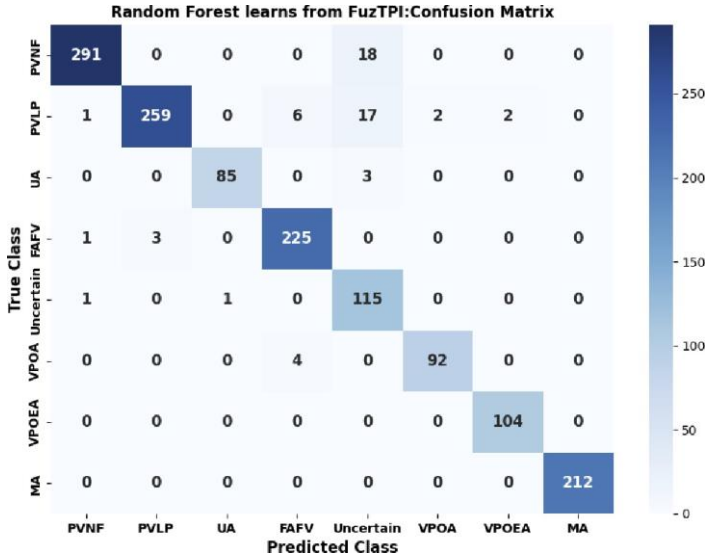


Figure 9: Confusion Matrix of FuzTPI-Driven RF Algorithm.

6. RESULTS AND DISCUSSION

This section presents the performance evaluation results and a discussion of the implemented classifiers. We compare the classifiers, analyze their performance, and discuss the implications of our findings. Additionally, the performance metrics of the Alignment process.

Analyzing Performance Measures: We present an analysis of the performance measures according to the testing dataset, including Accuracy, Precision, Recall, and F1 Score, for a range of models used in Arabic speech classification. Analyzing the table 5, we observe variations in the performance measures across the different models.

Table 5: Performance Comparison of Arabic Speech Classifiers

Classifier	Accuracy	Precision	Recall	F1 Score
FuzTPI-NB	55.13%	49.24%	55.13%	50.54%
FTARC	90.78%	91.97%	90.78%	90.60%
FuzTPI-SVM	95.42%	95.78%	95.42%	95.51%
FuzTPI-RF	**95.90%**	**96.51**	**95.90 %**	**96.02%**

The FuzTPI-NB achieved an accuracy of 55.13%. The FTARC classifier achieved an accuracy of 90.78% and demonstrated high precision, recall, and F1 Score. The FuzTPI-SVM model exhibited further improvement, with an accuracy of 95.42% and high precision, recall, and F1 Score. Notably, the FuzTPI-RF model showcased the highest performance among all classifiers, achieving an impressive accuracy of 95.90%. It also displayed superior precision, recall, and an F1 Score of 96.02, indicating its robustness in accurately classifying Arabic speech data. Note that the results of the FuzTPI-SVM and the FuzTPI-RF models are very close, so there is no clear winner model between them.
The accompanying figure, Figure 10, visually represents the comparative analysis of classifier performance. The bar chart reinforces the observations made in the table, clearly illustrating the variations in performance measures among the models. Overall, the results highlight the effectiveness of the FTARC, FuzTPI-RF, and FuzTPI-SVM models in improving the classification accuracy of Arabic speech. These models leverage advanced techniques and algorithms to enhance the alignment-driven classification process, resulting in superior performance compared to traditional ML classifiers.

Class Difficulty Analysis:

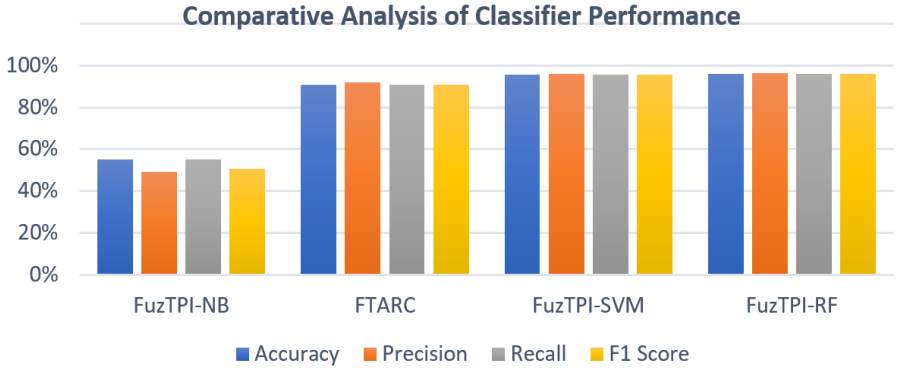


Figure 10: Comparative Analysis of Classifier Performance.

Table 6: Comparison of Successful and Unsuccessful Predictions by Classifier.

Classifier	FuzTPI-NB	FTARC	FuzTPI-SVM	FuzTPI-RF
Correct	795	1309	1376	**1383**
Incorrect	647	133	66	**59**

In the analysis of classifiers' performance, it is important to take into consideration the challenging classifications encountered. This information is presented in Figure 11, which provides insights into the number of correct and incorrect predictions made for each class across all utilized classifiers.

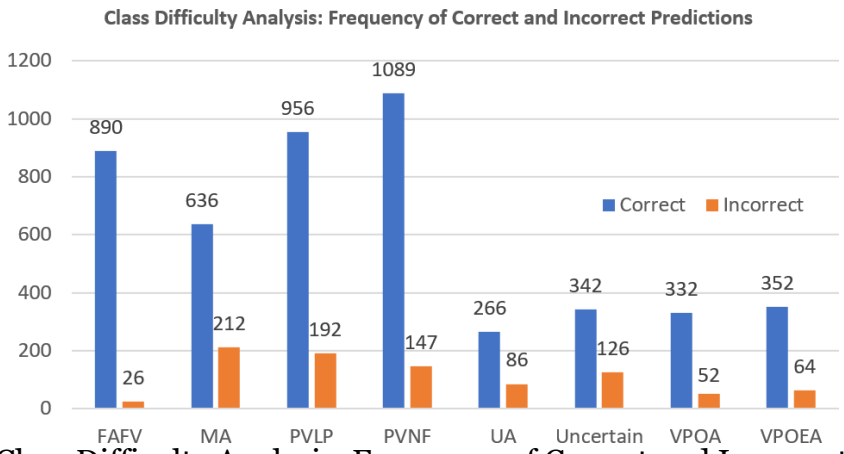


Figure 11: Class Difficulty Analysis: Frequency of Correct and Incorrect Predictions.

Among the classes, FAFV, PVLP, and PVNF stand out as the easiest to classify, as they have a high frequency of correct predictions and a low frequency of incorrect predictions. This suggests that these classes possess distinct features that enable accurate classification by the classifiers. On the other hand, the UA, and Uncertain classes presented significant challenges, resulting in a higher number of incorrect predictions compared to correct predictions. These classes are closely related to the texts with exhibit limited similarity. For MA, after reviewing the data, it is found that the incorrect predictions are all from FuzTPI-NB, and this is a point to improve the performance of this model. Enhancing the detection and classification of these classes could lead to substantial improvements in overall performance.

Analyzing Successful and Unsuccessful Classifications:

An analysis is conducted to assess the success and failure rates of the classifiers. Figure 12 provides clear visualizations of the distribution of successful and unsuccessful predictions.

Based on the data in Table 6, we can observe that the FuzTPI-RF classifier achieved the highest number of correct predictions with a total of 1,383. It is followed by the FuzTPI-SVM classifier with 1,376 correct predictions and the FTARC classifier with 1,309 correct predictions. The lowest number of correct predictions recorded by the NB classifier with 795.

Audio-Text Alignment Performance:

As in 2 of System Architecture, after obtaining the best Trained Classifier, we move to the last stage, which is Audio-to-Text Alignment. The Alignment module takes input from the winner-trained classifier, in our case FuzTPI-RF. Based on these classifications, it begins to align the audio files against the texts, potentially combining audio files to form complete verses. The final audio file is then linked to its Arabic diacritic-annotated text.

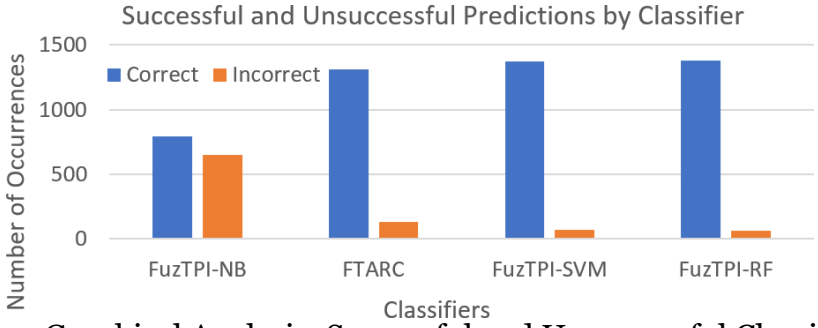


Figure 12: Graphical Analysis: Successful and Unsuccessful Classifications.

This module is based on the algorithm 2. To evaluate it, a dataset consisting of 1550 audio

files is created. Each file is annotated to identify whether it corresponds to a complete verse, a part requiring merging, or the end of a verse, etc. This is essential to measure the accuracy of the algorithm and facilitate error tracking, improvement, and measurement. In the alignment process, the QRSR system achieved an accuracy of 95.61%, with an alignment count of 1482 correct states and 68 incorrect states. The detailed performance measures for alignment are summarized in Table 7.

Table 7: Performance Measures for Alignment

Alignment Performance	Accuracy	Precision	Recall	F1 Score
Value	95.61%	86.88%	96.42%	90.80%

The table 7 illustrates the overall effectiveness of the alignment module. An accuracy of 95.61% indicates that the vast majority of audio files are correctly aligned. The high recall value of 96.42% suggests that most of the relevant audio files are correctly identified, while the precision of 86.88% shows the proportion of true positive alignments among the total positive alignments. The F1 score, a harmonic mean of precision and recall, provides a comprehensive view of the model's balance between precision and recall.

7. CONCLUSION AND FUTURE DIRECTIONS

This paper presented the QRSR system, its objectives, and the challenges it addresses. We explored several Machine learning algorithms and proposed the FTARC approach to enhance Arabic speech recognition. The evaluation results and discussion shed light on the performance of the implemented classifiers, identified challenging classifications, and highlighted the potential of incorporating the FTARC algorithm into other ML techniques. The study introduced the Fuzzy Text Alignment and Rule-based Classifier (FTARC) approach is proposed, achieving an accuracy rate of up to 90% in segmenting and aligning audio files. We combined the FuzTPI algorithm and Machine Learning (ML) models. This combination led to audio segmentation, text-audio classification, which achieved up to 96% accuracy, and text-audio alignment, which achieved 95.61%. The research used different classifiers, including Naive Bayes, Support Vector Machine (SVM), and Random Forest. Among them, the FuzTPI-Random Forest model had the highest performance. The results between FuzTPI-SVM and FuzTPI-RF are close, indicating both models were effective. This research contributes to the advancement of Arabic NLP systems, Arabic speech recognition systems, and segmentation/localization techniques, particularly in the context of Quranic studies. Our research contributes also to the development of an expanded textual audio dataset that can have a broader impact on Arabic speech recognition systems.

REFERENCES

[1] Aboalnaser, S. A. (2019). Machine learning algorithms in arabic text classification: A review. In *2019 12th international conference on developments in esystems engineering (dese)* (pp. 290–295).

[2] Aldarmaki, H., & Ghannam, A. (2023). Diacritic recognition performance in arabic asr. *arXiv preprint arXiv:2302.14022*.

[3] Anguera, X., Perez, N., Urruela, A., & Oliver, N. (2011). Automatic synchronization of electronic and audio books via tts alignment and silence filtering. In *2011 ieee international conference on multimedia and expo* (pp. 1–6).

[4] Baer, T., & Kamalnath, V. (2017). Controlling machine-learning algorithms and their biases. *McKinsey Insights*.

[5] Belete, D. M., & Huchaiah, M. D. (2022). Grid search in hyperparameter optimization of machine learning models for prediction of hiv/aids test results. *International Journal of Computers and Applications*, *44* (9), 875–886.

[6] Bhogale, K., Raman, A., Javed, T., Doddapaneni, S., Kunchukuttan, A., Kumar, P., & Khapra, M. M. (2023). Effectiveness of mining audio and text pairs from public data for improving asr systems for low-resource languages. In *Icassp 2023-2023 ieee international conference on acoustics, speech and signal processing (icassp)* (pp. 1–5).

[7] Chan, A. P., Chan, D. W., & Yeung, J. F. (2009). Overview of the application of "fuzzy techniques" in construction management research. *Journal of construction engineering and management, 135* (11), 1241–1252.

[8] Dean, D., Sridharan, S., Vogt, R., & Mason, M. (2010). The qut-noise-timit corpus for evaluation of voice activity detection algorithms. In *Proceedings of the 11th annual conference of the international speech communication association* (pp. 3110–3113).

[9] Dutoit, T. (1997). *An introduction to text-to-speech synthesis* (Vol. 3). Springer Science & Business Media.

[10] Gu, J., & Lu, S. (2021). An effective intrusion detection approach using svm with Naive bayes feature embedding. *Computers & Security*, *103,* 102158.

[11] Herrera-Viedma, E., Cabrerizo, F. J., Kacprzyk, J., & Pedrycz, W. (2014). A review of soft consensus models in a fuzzy environment. *Information Fusion*, *17* , 4–13.

[12] Humayun, M. A., Yassin, H., & Abas, P. E. (2023). Dialect classification using acoustic and linguistic features in arabic speech. *IAES International Journal of Artificial Intelligence*, *12* (2), 739.

[13] Islam, M. S., Jubayer, F. E. M., & Ahmed, S. I. (2017). A support vector machine mixed with tf-idf algorithm to categorize bengali document. In *2017 international conference on electrical, computer and communication engineering (ecce)* (pp. 191–196).

[14] Jiang, M., Liang, Y., Feng, X., Fan, X., Pei, Z., Xue, Y., & Guan, R. (2018). Text classification based on deep belief network and softmax regression. *Neural Computing and Applications*, *29,* 61–70.

[15] Kim, S.-B., Han, K.-S., Rim, H.-C., & Myaeng, S. H. (2006). Some effective techniques for naive bayes text classification. *IEEE transactions on knowledge and data engineering*, *18* (11), 1457–1466.

[16] Kostanyan, A. (2017). Fuzzy string matching with finite automat. In *2017 computer science and information technologies (csit)* (p. 9-11). DOI: 10.1109/CSITechnol.2017.8312128

[17] Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Text classification algorithms: A survey. *Information*, *10* (4), 150.

[18] Larbi, G. (2013). Voice search in the holy quran. In *2013 taibah university international conference on advances in information technology for the holy quran and its sciences* (pp. 413–418).

[19] Liew, C. S., Abbas, A., Jayaraman, P. P., Wah, T. Y., Khan, S. U., et al. (2016). Big data reduction methods: a survey. *Data Science and Engineering* , *1* (4), 265–284.

[20] Liu, Z., Lv, X., Liu, K., & Shi, S. (2010). Study on svm compared with the other text classification methods. In *2010 second international workshop on education technology and computer science* (Vol. 1, pp. 219–222).

[21] Lokhande, N. N., Nehe, N. S., & Vikhe, P. S. (2012). Voice activity detection algorithm for speech recog- nition applications. In *Ijca proceedings on international conference in computational intelligence (iccia2012), vol. iccia* (Vol. 6, pp. 1–4).

[22] Muhammad, W. M., Muhammad, R., Muhammad, A., & Martinez-Enriquez, A. (2010). Voice content matching system for quran readers. In *2010 ninth mexican international conference on artificial intelligence* (pp. 148–153).

[23] Qasim, H., & Abdulbaqi, H. A. (2022). Arabic speech recognition using deep learning methods: Literature review. In *Aip conference proceedings* (Vol. 2398, p. 050029).

[24] Radzi, S. F. M., Karim, M. K. A., Saripan, M. I., Rahman, M. A. A., Isa, I. N. C., & Ibahim, M. J. (2021). Hyperparameter tuning and pipeline optimization via grid search method and tree-based automl in breast cancer prediction. *Journal of Personalized Medicine*, *11* (10), 978.
[25] Ramırez, J., Segura, J. C., Benıtez, C., De La Torre, A., & Rubio, A. (2004). Efficient voice activity detection algorithms using long-term speech information. *Speech communication*, *42* (3-4), 271–287.
[26] Singh, P. (2021). Deploy machine learning models to production. *Cham, Switzerland: Springer*.
[27] Sun, Y., Li, Y., Zeng, Q., & Bian, Y. (2020). Application research of text classification based on random forest algorithm. In *2020 3rd international conference on advanced electronic materials, computers and software engineering (aemcse)* (pp. 370–374).
[28] Sundus, K., Al-Haj, F., & Hammo, B. (2019). A deep learning approach for arabic text classification. In *2019 2nd international conference on new trends in computing sciences (ictcs)* (pp. 1–7).
[29] Wahdan, A., Hantoobi, S., Salloum, S. A., & Shaalan, K. (2020). A systematic review of text classification research based on deep learning models in arabic language. *Int. J. Electr. Comput. Eng*, *10* (6), 6629–6643.
[30] Xu, S., Li, Y., & Wang, Z. (2017). Bayesian multinomial Naive bayes classifier to text classification. In *Advanced multimedia and ubiquitous engineering: Mue/futuretech 2017 11* (pp. 347–352).
[31] Yu, D., & Deng, L. (2016). *Automatic speech recognition* (Vol. 1). Springer.
[32] Zhang, T., & Kuo, C.-C. J. (2001). Audio content analysis for online audiovisual data segmentation and classification. *IEEE Transactions on speech and audio processing*, *9* (4), 441–457.