ARABIC DIACRITIC-AWARE TEXT-AUDIO SEGMENTATION AND ALIGNMENT MODEL (DASAM)

Adel Sabour[*], **Abdeltawab Hendawi**[**], **Mohamed Ali**[*]
[*]*Computer Science and Systems, University of Washington, Tacoma, USA, sabour@uw.edu, mhali@uw.edu*
[**]*Computer Science and Statistics, University of Rhode Island, Rhode Island, USA, hendawi@uri.edu*

Email Correspondence: adelsabour@gmail.com

Received: May 29, 2024 Accepted: June 4, 2024 Published: June 30, 2024

Abstract: This paper introduces the Diacritic-Aware Segmentation and Alignment Model for Arabic (DASAM). Diacritics are vital for pronunciation and meaning in the Arabic language but are often ignored by current speech recognition systems. DASAM is designed for word-level segmentation and alignment in unseen audio and associating them with diacritic-marked Arabic text. The DASAM approach uses linguistic analysis based on intonation rules. DASAM then applies Dynamic Time Warping (DTW) to match the reference audio word with its position in the unseen sentence audio. The model outputs a list of words with their start and end times in the recording. Tested on the Qur'an dataset, DASAM outperforms Google Speech-to-Text (STT) in predicting word timings. It achieves higher accuracy in text-audio alignment, with R^2 values of 0.959 and 0.957 for word start and end times, respectively (compared to Google STT's 0.870 and 0.849). Additionally, DASAM employs advanced signal processing techniques and demonstrates robustness across various audio variations. These results establish that DASAM constitutes a fundamental building block for speech-to-text conversion and linguistic research in Arabic, particularly for applications involving diacritics.

Keywords: Quran; Speech Recognition; Signal Processing; Phoneme Detection; NLP Techniques; Position Detection

Recommended APA Citation:
Sabour, A., Hendawi, A. & Ali, M. (2024). Arabic Diacritic-Aware Text-Audio Segmentation and Alignment Model (DASAM). *Elkawnie*, *10*(1), 1-27. https://doi.org/10.22373/ekw.v10i1.23637

Introduction

The Arabic language features a diverse array of linguistic styles. Each style expresses a dialect and may carry additional meanings and clarifications to the text. Unlike the dialects, which might contain linguistic errors, these styles are free from such inaccuracies and often enhance the text's meaning. To the best of the authors' knowledge, no dataset other than the Qur'an has a single Arabic text written more than once in multiple linguistic styles. The Qur'an uniquely contains the same text in 20 linguistic styles, all authenticated by linguists. These texts maintain identical letter shapes but differ in diacritic marks, altering pronunciation and sometimes clarifying additional meanings.

The gap in resources poses a significant challenge in analyzing these variations comprehensively. While audio recordings exist for all 20 styles, only seven currently have corresponding digitized text representations. In the field of Natural Language Processing (NLP), Automatic Speech Recognition (ASR) is a technology that translates spoken language into written text. Developing an ASR system for Arabic that accommodates diacritics could bridge this gap. While various models have achieved success in Arabic speech recognition, they often overlook the importance of diacritics in the language. Arabic diacritics are used to clarify pronunciation, maintain the accent, and define intended meanings. Without diacritics, many words remain ambiguous or carry multiple potential meanings. Unfortunately, leading ASR platforms like Google STT Cloud (2024), Microsoft Azure Speech Services Microsoft (2024), and Amazon Transcribe Services (2024) typically do not support Arabic diacritics.

Our related work identified a scarcity of Arabic datasets annotated with diacritics at the word level Aboalnaser (2019); Sundus et al. (2019). Most available audio datasets are at the sentence level, which limits training effectiveness for an ASR model. Focusing on individual words allows models to learn the specific phonetic patterns associated with each word and its diacritics Zheng et al. (2003). This lack of suitable datasets motivated our pursuit of a solution. Audio segmentation and audio-text alignment are fundamental steps in ASR processes Stan et al. (2016). Audio segmentation involves dividing the audio track into smaller units such as phonemes, sentences, and words. The text-audio alignment synchronizes audio files with their corresponding textual representations. This synchronization ensures the creation of accurate and reliable audio-text datasets.

Robust Speech Recognition (SR) is vital. It contributes to NLP, Machine Learning, Sentiment Analysis, Text-to-Speech, and many others. SR involves multiple stages, including dividing continuous speech into smaller units like sentences, words, and phonemes. Accurate segmentation is crucial for aligning speech with corresponding text, enabling further analysis and improving technology built on these audios. DASAM, defined as the Diacritic-Aware Segmentation and Alignment Model for Arabic, is a component of the Quran Speech Recognition (QRSR) system (Sabour et al., 2023). DASAM transcends basic dataset generation by establishing a method for segmenting and aligning audio recordings with their corresponding written texts. Accurate segmentation isolates individual words, while alignment ensures that audio segments match the corresponding written words with diacritics. This alignment enables ASR models to learn the nuanced relationship between pronunciation and diacritics in Arabic. DASAM focuses on pinpointing the location (start and end time) of a specific word within a spoken sentence. It achieves this by analyzing and comparing sound characteristics, making it adaptable to various audio sources without requiring training data. DASAM relies on direct audio analysis, making it independent of training data and adaptable to various audio sources.

DASAM utilizes the Mel-Frequency Cepstral Coefficients MFCC features to pinpoint word location. MFCCs capture the essential characteristics of a sound Zheng et al. (2001). DASAM compares the MFCCs of an individual word (the "first sound") with the corresponding segment within a complete sentence (the "second sound"). This meticulous comparison allows DASAM to determine the word's precise start and end time within the sentence audio. DASAM is not an ASR and can't be directly comparable to ASR. DASAM can be compared with the word alignment feature offered by some ASR systems, like Google STT. Although these systems have different goals (ASR for speech-to-text conversion, and DASAM for word alignment), they share a similar result of identifying word boundaries. Our comparison targets the specific word timing results produced by both systems. This comparison allows us to assess the accuracy of DASAM's word boundary detection against an established industry standard. The Qur'an has a unique role in this process. With its well-defined text and established pronunciations, the Quran serves as a reliable "ground truth" for DASAM. DASAM uses the Quran's established timings as a benchmark to assess its word segmentation and alignment accuracy.

DASAM's ability to accurately predict the start and end times of words within audio recordings is notably superior, with R^2 values of 0.959 and 0.957 for start and end times respectively, compared to 0.870 and 0.849 for Google STT. This chapter illustrates DASAM's strengths through detailed statistical analysis, establishing its potential as a highly effective model for word-level alignment in Quranic diacritics tasks. DASAM's contributions extend beyond dataset creation. DASAM constitutes a fundamental building block for advancing NLP technologies by improving the accuracy of Arabic speech analysis. These include improved machine translation, enriched text-to-speech systems, and more effective speaker diacritization, and automatic speech summarization by recognizing nuanced linguistic features. Our model paves the way for in-depth linguistic analysis and more robust NLP applications.

Related Work

Aldarmaki and Ghannam (2023) examine diacritic recognition in Arabic ASR systems. They tackle the issue of incomplete diacritization due to missing marks in speech corpora. The study compares input diacritization and text-based diacritization as a post-processing step. Researchers fine-tune pre-trained ASR models with different diacritization conditions. They evaluate diacritic recognition performance using coverage and precision metrics. ASR diacritization outperforms text-based diacritization, especially with manual diacritization. This research relates to our work, exploring diacritic recognition in ASR systems. It highlights the benefits of ASR diacritization for improved performance.

Qasim and Abdulbaqi (2022) examine Arabic speech recognition using deep learning techniques. Their review highlights the importance of speech recognition for human-computer interaction. They emphasized that Arabic speech recognition has received less attention than other languages. Their paper explores deep learning techniques, including Deep learning techniques like DNNs, RNNs, and CNNs show promise in speech recognition.

Javed et al. (2020) addressed unsupervised phoneme segmentation in Classical Arabic speech. Their approach utilizes FICV features, a method that leverages the forward and inverse characteristics of the vocal tract. Evaluated on a Quranic Arabic dataset, FICV achieved a total error rate of 14.48% and an accuracy of 85.2% within a 10ms alignment error, outperforming existing hidden Markov model (HMM) based methods. This research highlights the potential of FICV features for speech segmentation in Classical Arabic, although focusing on phoneme-level rather than word-level boundaries.

Abdo et al. (2016) proposed a speaker-independent, semi-automatic system for segmenting Arabic speech into syllables. Their method utilizes MFCCs and a two-step approach for segmentation: a) Maxima Extraction: The system identifies potential syllable boundaries by locating maxima in the delta function of the first MFCC coefficient. b) Template Matching: It then employs template matching techniques using reference utterances to refine the boundaries identified in step 1. This approach achieved a segmentation success rate of approximately 91.5% on a dataset of 276 utterances, divided into 2544 syllables. While focusing on syllable-level segmentation compared to our word-level with diacritics, this research aligns with ours by targeting Arabic speech segmentation.

Abdo et al. (2014) introduced an algorithm for segmenting emphatic and non-emphatic sounds in continuous Arabic speech. Their method utilizes MFCCs and analyzes peaks in the delta MFCC to locate sound boundaries within the speech signal. The algorithm is designed for sub-syllable level segmentation and was evaluated on a Quran recitation corpus of 480 words, achieving up to 90% accuracy in boundary detection. However, it targets a lower segmentation level (sub-syllable) compared to our word-level focus with diacritic emphasis.

Absa et al. (2018) introduced a hybrid unsupervised segmentation algorithm for Quranic Arabic speech. Their method combines three basic speech features (entropy, zero-crossings, energy) and optimizes segmentation using a genetic algorithm. This approach achieved a 20% improvement in accuracy over traditional single-feature-based techniques. The work explores various thresholding techniques (median, mean, mode) for detecting syllable segments and discusses their strengths and weaknesses. Additionally, they propose a regression fusion scheme for robust boundary prediction. Their research focuses on frame-level segmentation and emphasizes the benefits of feature fusion and genetic algorithm optimization for Quranic Arabic speech processing.

The previously described research provides valuable insights into segmentation and alignment techniques for Quranic speech. We present this comparison in Table 1. This table shows how DASAM relates to other research in the field. By comparing DASAM to these studies, we can see its position within the existing body of work.

Table 1: Comparison of Segmentation and Alignment Techniques for Quranic Speech

Criteria	DASAM	Abdo et al. (2014)	Javed et al. (2020)	Absa et al. (2018)	Abdo et al. (2016)
Segmentation Focus	Word-level alignment with diacritics focus	Emphatic and non-emphatic sound segmentation	Phoneme segmentation using cosine distances	General speech segmentation for Quran recitation	Syllable segmentation for speech recognition
Methodology and Techniques	Linguistic intonation rules with DTW	Peak detection using MFCC	Cosine distance drives phoneme boundary using FICV	Feature fusion with Genetic Algorithm optimization	Template matching with maxima extraction from delta MFCC
Algorithm Type	Advanced automatic using DTW	Automatic with predefined rules	Unsupervised	Hybrid unsupervised	Semi-automatic
Dataset & Audio Focus	Quranic Arabic	Quranic Arabic	Quranic Arabic	Quranic Arabic	Quranic Arabic
Dataset Size	20K from two reciters, 260k words with audio variation	480 words from six speakers	725 words	KACST, 5935 files from 10 reciters	2544 words, from 12 reciters
Feature Sets	MFCC	MFCC	FICV, cosine distance scores	Entropy, zero crossings, energy	MFCC
Performance Metrics	R^2 95.9% and 95.7% for start and end times	90% accuracy	85.2% accuracy	20% improvement over base-line models	91.5% accuracy

This comparison in Table 1 delineates the distinct methodologies and achievements of each study. Table 1 illustrates DASAM's advancements in handling diacritic-sensitive segmentation.

From Raw Audio to Rich Data: The DASAM Dataset

We describe the development of the dataset, starting with raw audio data and progressing towards a resource enhanced for model evaluation. Available on our website, Sabour and Ali (2024).

1. The Raw Audio Data:

The starting point for DASAM was a collection of sentence-level audio files. Each audio file was paired with a separate file containing timestamps. These files contained values representing the words' sequence and starting, and ending positions of each word in the audio files. The actual words for these timings are merged with the data later in the pipeline. The audio files were assessed by listening. We curated a final reviewed dataset of 10,000 audio files from two different Quran reciters. The dataset's timing information was accurate enough, obviating the need for manual adjustments. This dataset forms the foundation for the evaluation of the DASAM alignment model.

2. **Our Enhanced Dataset:**

To enhance the raw audio data, we performed several steps:

2.1. **Segmentation:** Automated processing splits each audio file into word-level clips, creating a dataset of audio segments.

2.2. **Text Alignment:** Each word and its corresponding sentence were meticulously linked to their Arabic text with diacritics. This creates a comprehensive linguistic dataset.

2.3. **Feature Extraction:** We extracted MFCC's audio features for each segmented word. These features act as reference points when comparing unseen audio to identify words and their timing. The model compares the extracted features from the unseen audio to the features of these reference words to determine if a particular word exists in the unseen audio and pinpoint its starting and ending points within the recording.

2.4. **Audio Variations:** To increase the model's robustness to different speech variations, we created 13 audio variations. These variations were generated by modifying the original audio (e.g., changing pitch to resemble a child's voice or adding noise). A detailed explanation of audio variations and their impact on the model is presented in sections Handling Audio Variations and Statistical Insights: Unveiling the Details in the Charts.

Enhanced Dataset Summary: The final enhanced dataset includes the original audio files, the corresponding Arabic texts with diacritics, and a rich set of extracted audio features. We used the original 10,000 files for feature extraction. With the addition of audio variations, the total dataset size reaches 260,000 audio files (10,000 original files used for feature extraction multiplied by 13 variations for two reciters).

DASAM Model Overview

This section delves into the core functionalities of the Diacritic-Aware Segmentation and Alignment Model for Arabic (DASAM). DASAM tackles the challenge of segmenting and aligning words with diacritics within Arabic speech. The methodology employs a multi-stage process to achieve this. We'll explore each stage in detail, referencing the DASAM's Word-Audio Localization Process diagram (Figure 1) to visualize the overall workflow.

1. **Baseline Data Preparation:** We leverage existing ground truth data for the Holy Quran, containing the start and end times of each word for specific sentence-level audio files. These recordings were automatically segmented into individual word clips. Each clip is linked to its corresponding Arabic text with diacritics. These segments are visualized as “Reference Word Audios” in Figure 1.

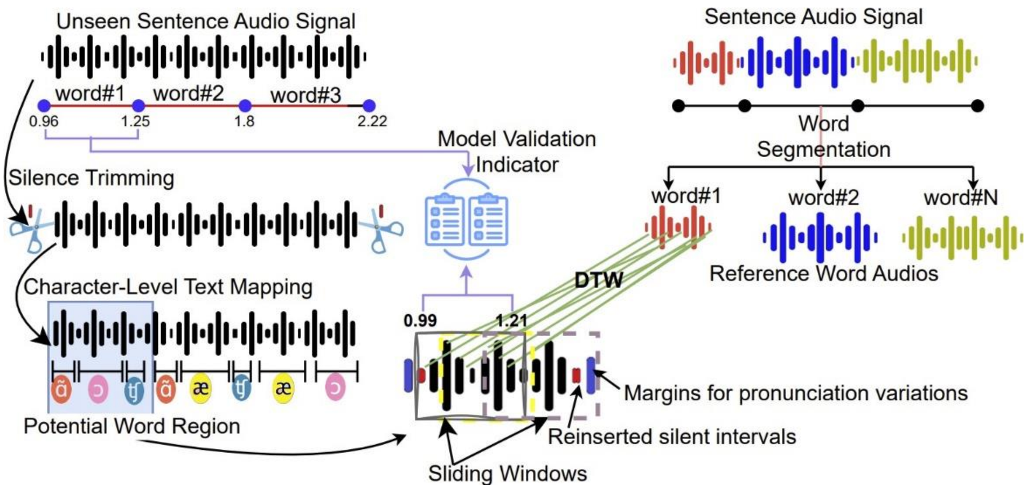

Figure 1: DASAM’s Word-Audio Localization Process.

2. **Feature Extraction:** In speech processing, audio analysis often involves converting the raw audio signal into a numerical representation that captures its essential characteristics. This simplifies sound comparison and matching tasks. To achieve this, we extract features from the audio data.
   We opted for MFCCs that capture and provide representations of the time-varying spectral characteristics for the audio speech Zheng et al. (2001). We extract these features from both the reference words and unseen phonetic phrases within the audio data.
3. **Silence Trimming:** Many audio recordings contain periods of silence without speech. To achieve accurate alignment, it’s crucial to identify and remove these silent intervals. This process, visualized in Figure 1 as “Silence Trimming,” ensures the alignment focuses solely on the spoken content within the unseen sentence audio signal. We employ a technique called Voice Activity Detection (VAD) to achieve silence trimming. VAD algorithms segment audio files by identifying periods of silence Dean et al. (2010). We utilize a segmentation algorithm based on amplitude analysis, as described in Lokhande et al. (2012). This algorithm effectively distinguishes between speech and silence by dynamically adjusting a silence threshold based on the audio amplitude.
   Silence removal is important because its presence can negatively impact the subsequent step of Character-Level Text Mapping. Silence could result in letters being assigned to incorrect spoken words or extending into silent gaps.

4. **Character-Level Text Mapping:** This stage maps characters and diacritics in the text to their corresponding audio segments. It considers the rules of Arabic phonetics El-Imam (2004), where letters can vary slightly in duration. Weights are assigned based on average pronunciation lengths (e.g., vowels are typically longer than consonants). This helps distribute characters across the audio, estimating each word's position within the sentence (see "Character-Level Text Mapping" in Figure 1). A key challenge in Arabic word-level alignment is the presence of similar-sounding words. This arises from the inherent characteristics of the language, where some letters share close pronunciations Rahman et al. (2024). Due to these phonetic similarities, directly comparing entire words within the audio can be inefficient. Character-Level Text Mapping tackles this challenge by leveraging knowledge of Arabic language structure. It acts as a pre-processing step before acoustic analysis. This process essentially "distributes" the words across the audio based on estimated pronunciation lengths for each character. By performing this mapping, we significantly reduce the search space for word-level alignment. Instead of searching for entire words within the whole sentence audio, the system focuses on smaller, more manageable segments corresponding to individual characters. This approach significantly increases the efficiency of the word-level alignment process.

Table 2: Moras Distribution for Arabic Word Pronunciation.

Word	كٓـــــهـــــيـــــعٓـــــصٓ				
letter	ـــصٓ	ـــعٓـــ	ـــيـــ	ـــهـــ	كٓـــ
#Moras	**6**	**6**	**2**	**2**	**6**

Table 2 shows an example of the number of moras required to pronounce each letter in a word. Mora is a unit of relative duration in spoken language, typically corresponding to the time it takes to pronounce a letter or a syllable. Although this is fixed and standard, it varies according to the reciter's estimates. Estimates vary, but the reciter should maintain the same proportionality between the number of moras for all letters regardless of the speed. The example in Table 2 shows a word that takes 22 moras to pronounce correctly. For example, if each mora takes a second, the reciter takes 22 seconds to pronounce this word. And if the reciter performs the phoneme in half a second, then it will take 11 seconds to pronounce it. The number of moras has several bases for its calculation.

Despite English not having fixed rules for mora timing we still can see examples that can express the mora's idea. Table 3 shows an example in English. It highlights differences in mora length in the simple sentence "Cute Express", compared to the International Phonetic Alphabet (IPA) for each letter.

Table 3: Moras Distribution of English Sentence Pronunciation.

English	C	u	t	e	e	x	p	r	e	s	s
IPA	k	yo͞o	t		i	k's	p	r	e	s	
#Moras	1	3	1	0	1	2	1	1	1	1	0

Character-Level Text Mapping Pipeline:

This pipeline details (Figure 2) the process of mapping characters and diacritics in Arabic text to their corresponding audio segments within an audio file. It considers the rules of Arabic phonetics, where letters can have slightly varying durations.

- Inputs:
  a) **Arabic Text with Diacritics:** The input text contains Arabic characters and diacritics.
  b) **Pronunciation Rules:** These rules govern how the length of pronunciation varies for each letter based on its position within a word (beginning, middle, end), neighboring characters, and diacritics Balula et al. (2021). These rules are typically derived from Tajwid, a science focused on the correct pronunciation of Arabic, especially in the Quran Al-Fadhli et al. (2023).
  c) **Audio File:** The audio file containing the spoken Arabic text for which word boundaries need to be identified.

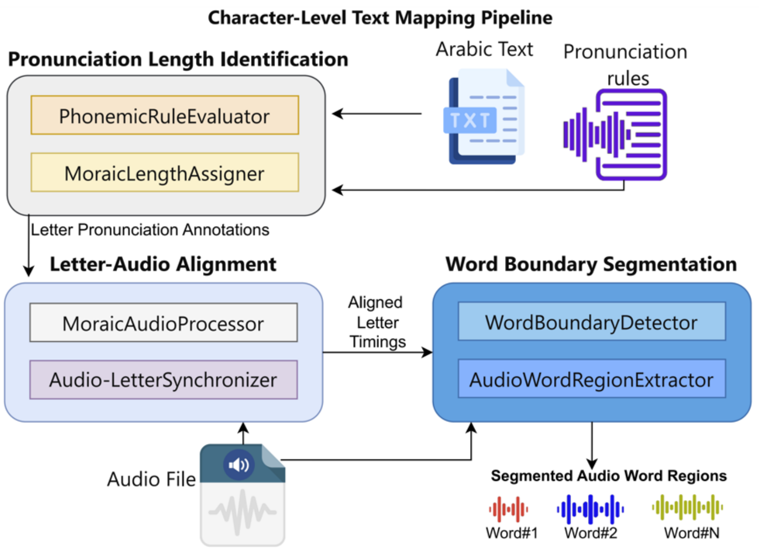

Figure 2: Character-Level Text Mapping Pipeline.

- Modules:
  a) **Pronunciation Length Identification:** This module determines the pronunciation length of the text. It consists of two components: *PhonemicRuleEvaluator* and *MoraicLengthAssigner*. These work together

to identify relevant pronunciation rules, analyze letter and diacritic interactions, and assign time units to each letter or phoneme.

- **PhonemicRuleEvaluator:** employs a combination of text parsing algorithms and rule-based analysis to detect diacritics and letter positions in the text. Determines the phonetic duration of Arabic script components in our model. Uses Contextual Phonetic Mapping to analyze letter and diacritic marker interactions. Scans Arabic text to identify pronunciation duration based on Pronunciation Rules. These rules account for diacritic adjacency and letter position in determining phonetic duration. Assign time units to each phoneme according to these rules. For example, a letter with a specific diacritic mark has a set pronunciation duration in a time unit.
- **MoraicLengthAssigner:** Based on the identified rules, the *MoraicLengthAssigner* assigns a "mora" value to each letter or phoneme. A mora is a unit of relative pronunciation length, roughly corresponding to the time it takes to pronounce a letter or syllable.
  Example: The Arabic letters "م" (m) and "ن" (n) followed by the diacritic sign shaddah like "مّ" and "نّ" require a prolonged pronunciation, equivalent to 2 moras. This rule is called *ghunah*. According to the rules, the *PhonemicRuleEvaluator* identifies this rule and triggers the *MoraicLengthAssigner* to assign 2 moras, accurately reflecting the required duration. However, the actual timing in seconds that corresponds to a mora can vary depending on the speaker's pace, though the relative pronunciation length (in terms of moras) remains consistent.

b) **Letter-Audio Alignment:** Aligns textual phoneme durations with audio data, ensuring synchronization of spoken content.
   - **MoraicAudioProcessor:** This module first obtains the total number of moras from the text analysis. It then extracts audio features (MFCCs) from the audio file and calculates the total audio length (excluding silence). Finally, it calculates the "Moraic Frame Rate" (MFR) by dividing the audio feature length by the total number of moras. This MFR represents the number of audio frames (time slices) per mora. The MFR is determined by the equation (1).

$$\text{MFR}=\frac{\text{Total Audio Length}}{\text{Total Number of Moras}} \quad \ldots\ldots\ldots\ (1)$$

where Total Audio Length is the length of the audio file excluding any periods of silence, and Total Number of Moras is obtained from the text analysis of the Arabic script. Consider a practical example to illustrate the calculation of the MFR. Assume a sentence requires ten moras for proper pronunciation. If a reader pronounces this sentence in 20 seconds, the MFR, according to equation number (1), would be 2 seconds per mora. In another scenario, if a different reader speaks more slowly and takes 30 seconds to pronounce the same sentence, the MFR would then be 3 seconds per mora. These examples demonstrate how the MFR can vary with the speaking rate, even though the number of moras required remains constant. This variability emphasizes the adaptability of the DASAM model to different speech rates while maintaining synchronization between text and spoken audio.

- **Audio-LetterSynchronizer:** This module uses MFR and frame rate information to convert theoretical mora boundaries into practical start and end times. This conversion is done for each letter within the audio file. The process involves two steps: (a) Calculating the temporal position of each mora using the MFR and frame rate. (b) Determining the start and end times for each letter by multiplying the duration of one mora by the starting and ending mora indices of that letter.

Calculation Example: Assume the MFR indicates there are X frames per mora, and the audio frame rate (frames per second) is Y. Then the time duration T in seconds for one mora is calculated as in (2):

$$T=\frac{X}{Y} \text{ (seconds per mora) } \ldots\ldots\ldots\ (2)$$

Given this duration T, the start and end times for any letter spanning from mora index a to b are calculated as: Given this duration T, the start and end times for any letter can be determined by multiplying T by the starting and ending mora indices of that letter. If a letter spans from mora index a to b, then the calculation will be as in (3 and 4):

$$\text{Start Time}=a\times T \ \ldots\ldots\ldots\ (3)$$

$$\text{End Time}=b\times T \ \ldots\ldots\ldots\ (4)$$

Application Example: Consider a scenario where a specific letter spans the third to the fifth mora in the text. The MFR is calculated to be 100 frames per mora with an audio frame rate of 1000 frames per second. The time duration T for one mora would thus be: The time duration T for one mora is given by eq (5)

$$T=\frac{100}{1000}=0.1 \text{ seconds per mora } \ldots\ldots\ldots\ (5)$$

Thus, for this letter:

Start Time = 3 × 0.1 = 0.3 seconds. End Time = 5 × 0.1 = 0.5 seconds. This calculation helps us determine the exact timing of each letter in the audio file. It transforms theoretical mora assignments into precise timing annotations. The Audio-Letter Synchronizer ensures the correct alignment of each letter’s pronunciation within the spoken sentence. This enhances the accuracy of subsequent word boundary segmentation.

c) **Word Boundary Segmentation:** Segments the audio into discrete words based on the synchronized letter timings.

- **WordBoundaryDetector:** This module takes the letter timings and the original text as input. It identifies word boundaries based on the letter timings and their corresponding text characters. This results in a list of words, each with its start and end time within the audio file.
- **AudioWordRegionExtractor:** This module extracts the corresponding audio segments for each identified word based on their start and end times.

- **Output:**

The final output consists of a list, where each element represents a word from the original text along with its corresponding audio segment extracted from the audio file. This pipeline effectively addresses the challenge of similar-sounding words in Arabic by leveraging pronunciation rules. It facilitates the identification of word boundaries within the audio file.

5. **Potential Segment Identification:** The possible segment is determined based on the linguistic distribution along the spoken phoneme of the sentence. This helps in reducing the length of the sound within which you need to search for the position of each word in the sentence. After silent intervals are reintegrated, segments are aligned with the original audio to maintain correct timing from the ground truth data. Margins around each word segment are adjusted for potential pronunciation variations, in preparation for sound matching and alignment. As depicted by “Reinserted silent intervals” and “Margins for pronunciation variations” in Figure 1.
6. **Refined Word Alignment:** Dynamic Time Warping (DTW) is a technique that helps align two sequences that may have different speeds or lengths Permanasari et al. (2019). Here, it compares the features of the potential word region with the reference word audio to find the best match. As shown by “DTW” in Figure 1, where it tries to find the closest location in the Potential Segment to the Reference Word Audio. Word boundary segmentation based on linguistic rules provides optimal pronunciation timings, but human variations and errors occur. Meanwhile, DTW may struggle with similar-sounding or repeated words Yalova et al. (2023). To overcome this, we combine linguistic segmentation with DTW. Linguistic rules determine the optimal pronunciation location, and we add margins for phonetic distortion. DTW then refines the result within these limits, preventing large errors and improving overall accuracy. By integrating both systems, we achieve greater improvement and flexibility with human estimates.
7. **Timing Data Extraction:** A sliding window technique pinpoints the precise start and end times of each word within the sentence-level audio Wei et al. (2023). By moving the window across the unseen sentence audio and comparing the features within the window to the reference word audio, the model determines the most likely start and end times of the word. This technique is visualized with “Sliding Windows” in Figure 1, comparing the times deduced by the model to the ground truth.

8. **Evaluation:** The effectiveness of the DASAM model is assessed by the Model Validation Indicator which compares the model's predicted word timings with the actual word boundaries to assess the model's accuracy. Then we compare this accuracy to Google STT alignment's results for the same audio data. The evaluation primarily focuses on the accuracy of detecting word start and end times.

This structured, multi-stage approach ensures that DASAM not only generates datasets but also enhances understanding and processing capabilities for Arabic speech recognition technologies.

DASAM's Operational Algorithm:

Algorithm 1: Audio-Text Alignment for Pronunciation Assessment

```
Input:    text (diacritic-annotated),
          phonetics (word-level),
          audio (unseen sentence audio),
          rules (pronunciation rules).
Output:   FinalResults: List of words with aligned audio timestamps
1 mfcc_sentence ← ExtractMFCC(audio, level="sentence")
2 mfcc_words ← ExtractMFCC(phonetics, level="word")
3 audio_trimmed ← TrimSilence(audio)
4 potential_regions ← Ø
5 foreach (word in text)
6     pronunciation_length ← PhonemicRuleEvaluator(text, rules)
7     moras ← MoraicLengthAssigner(pronunciation_length, audio)
8     mfr ← MoraicAudioProcessor(moras, mfcc_sentence)
9     letter_timings ← AudioLetterSynchronizer(mfr, mfcc_sentence)
10    word_boundary ← WordBoundaryDetector(letter_timings, text)
11    audio_region ← AudioWordRegionExtractor(word_boundary, audio)
12    Add to potential_regions the pair (word, audio_region)
13 segmented_regions ← Ø
14 foreach (region in potential_regions)
15    Add reinserted silence to region
16    Add margins for pronunciation variations
17    Add to segmented_regions
18 final_results ← Ø
19 foreach (word in segmented_regions)
20    foreach (sliding_window_word in word)
21       dtw_distance ← DTW(phonetics, sliding_window_word.audio)
22       min_distance ← ∞
23       if dtw_distance < min_distance then
24          min_distance ← dtw_distance
25          start_time_msec ← CalculateStartTime(sliding_window_word)
```

```
26          end_time_msec ← CalculateEndTime(sliding_window_word)
27          Add (sliding_window_word.text, start_time_msec,
                                  end_time_msec) to final_results
28 RETURN final_results
```

The Algorithm 1 provides an overview of the DASAM Model's workflow. It streamlines the detailed operations by omitting preliminary data preparation and final result comparison procedures. Focusing on the main processes facilitates a deeper understanding of the model's inner workings.

The first step involves feature extraction (lines 1, 2). The MFCCs are extracted from both the sentence audio and reference word audio segments. Subsequently, silence removal is performed on the unseen sentence audio (line 3). This step ensures that the linguistic alignment focuses solely on the spoken content.

Lines 4-12 address Character-Level Text Mapping. The algorithm analyzes the Arabic text with diacritics and pronunciation rules. This analysis estimates the pronunciation length for each character based on its position within a word and surrounding letters. This information is then used to define potential word regions within the sentence audio.

Lines 13-17 expand potential word regions. Silence intervals are re-inserted to align predicted times with standard times for comparison. Additionally, margins are added to account for variations in pronunciation.

Lines 18 to 27: A sliding window technique iterates through each potential word region. DTW uses the Euclidean distance function Deriso and Boyd (2023). It calculates a numerical distance between audio files. The DTW calculates the distance between the reference word audio segment and each window. This finds the most similar match. Based on the best match, the algorithm extracts the precise start and end times for the word within the sentence audio. The final output is a list containing each word with its corresponding start and end times within the unseen sentence audio. This algorithm highlights the key steps involved in segmenting and aligning words within Arabic speech using DASAM.

Handling Audio Variations

This section explores the concept of audio variations and their role in enhancing the robustness of the DASAM model. Audio variations are synthetic speech recordings created by modifying original audio to simulate real-world listening environments Alkayyali et al. (2023). These variations introduce factors like changes in pitch, background noise, and speaker characteristics Bakır et al. (2024). To improve the model's generalizability and adaptability to real-world scenarios. We used the audio variations with DASAM for several reasons:

- **Mimicking Real-World Speech:** They help us test DASAM under conditions that resemble real-world speech scenarios (noise, different genders, etc), making the model more adaptable.

- **Enhancing Dataset Diversity:** Variations effectively enlarge the dataset size, providing a wider range of speech samples for model evaluation Alkayyali et al. (2023). This leads to more reliable and generalizable results.
- **Building Robust Models:** By incorporating audio variations, we can assess DASAM's ability to handle speech variations, ensuring its effectiveness in diverse listening conditions.

Mathematical Model: Addressing Speech Variations

Phonetic variations can affect the ground truth of word-level text-audio alignment. The impact depends on the type of variation method used:

- **Non-Time-Altering Methods:** Some phonetic variations do not change the audio signal's duration. These include changes to spectral characteristics or subtle articulation. They do not alter the timing information. Therefore, the ground truth of word-level alignment remains unchanged.
- **Time-Altering Methods:** Other phonetic variation techniques involve stretching or compressing the audio. These directly impact the pronunciation and length of words and sentences. In such cases, the ground truth for word-level alignment must be adjusted. The amount of time shift can be calculated. It is based on the expansion or contraction factor applied by the chosen variation method. Each subsequent word in the aligned text needs a shift by the same amount. This maintains accurate alignment with the modified audio.

For audio variations that involve speed adjustments (acceleration or deceleration), we apply a mathematical model. This model calculates the adjusted word-level timing. Our model operates on two principles during speed adjustments:

- **Equal Displacement:** When the sentence speed is altered, every word is uniformly affected. That is, the start and end times of all words shift proportionally to the change in sentence duration.
- **Equal Rate of Displacement:** The rate at which the timings of each word are displaced is consistent across the sentence. This means the relative duration of each word compared to the sentence remains constant, regardless of the speed adjustment.

This approach ensures that the sequential integrity of words is preserved. Each word's relative duration within the sentence remains constant across all speed variations. Our confidence in this method stems from its alignment with the inherent properties of linear time scaling in audio processing. Here is how the model is formulated:

Let S_o represent the original sentence duration, and S_m denotes the modified sentence duration after speed adjustment. The original and modified timings for each word, represented by $t_{start,o}$, $t_{end,o}$ for the original, and $t_{start,m}$, $t_{end,m}$ for the modified, are related by the equations (6) and (7):

$$t_{start,\ m}=t_{start,\ o}\times\frac{S_m}{S_o} \quad \ldots\ldots\ldots (6)$$

$$t_{\text{end, m}} = t_{\text{end, o}} \times \frac{S_m}{S_o} \qquad \text{............ (7)}$$

This approach ensures that the temporal relationship between words remains consistent. It preserves the linguistic integrity of the sentence across variations. Example: Consider a sentence with an original duration of 20 seconds. Within this duration, the word “quick” is pronounced from the 2nd to the 4th second. If an audio variation involves halving the sentence’s duration to 10 seconds, the start and end times for “quick” adjust proportionally:

- Original Duration (S_o): 20 seconds
- Modified Duration (S_m): 10 seconds
- Original Timing for “quick”: 2 to 4 seconds
- Adjusted Timing for “quick”: 1 to 2 seconds

This example illustrates the application of the mathematical model. It adjusts word-level timings in response to speed variations. This enables us to derive accurate ground truth annotations for each audio variation.

Discussion and Comparative Analysis

This section discusses the performance of DASAM in word-level alignment. Alignment involves determining the precise start and end times for each word within an audio file. Here, we evaluate DASAM’s ability to perform this task by comparing its results with those from Google STT, a well-established industry standard.

We provided both DASAM and Google STT with the same audio files. Both systems returned a list of words along with their corresponding start and end times within the audio file. This common output format allows for a direct comparison of performance between the two systems.

1. Metrics Comparison: Understanding Key Performance Metrics

This section delves into the key performance metrics used to evaluate the word-level alignment capabilities of DASAM and Google STT. These metrics provide quantitative insights into how accurately each system identifies word start and end times within audio recordings. We will explain each metric and present the results in table 4 for easy comparison.

Table 4: Summarizing the performance of DASAM and Google STT on the key metrics

Metric	**Value**	
	DASAM	**Google STT**
MSE (Start) (ms^2)	61.4M	138M
MSE (End) (ms^2)	65.4M	177.3M
MAE (Start) (ms)	14.1K	27.6K
MAE (End) (ms)	15.9K	30.7K
R^2 (Start)	0.959	0.870
R^2 (End)	0.957	0.849

Metric	Value	
	DASAM	Google STT
Pearson Correlation (Start)	0.981	0.936
Pearson Correlation (End)	0.982	0.927

The key performance metrics employed in this analysis:

a) **Mean Squared Error (MSE):** MSE measures the average squared difference between the predicted word timings (start and end times) by a system and the actual timings in the reference data. Lower MSE values indicate better alignment accuracy, as smaller differences translate to less error. MSE is typically expressed in milliseconds squared (ms^2). An MSE of 0 ms^2 would indicate perfect alignment, where all predicted timings perfectly match the reference. Smaller values represent greater accuracy.
DASAM reports MSE of 61.4M/65.4M for start and end times respectively, while Google STT exhibits a higher MSE of 138M/177.3M, indicating DASAM's superior accuracy in error minimization.

b) **Mean Absolute Error (MAE):** MAE calculates the average absolute difference between the predicted and actual word timings. It provides a simpler interpretation of error compared to MSE, focusing on the magnitude of the errors without considering their direction (positive or negative). MAE is also expressed in milliseconds (ms). Similar to MSE, lower MAE values indicate better alignment.
DASAM's MAE values stand at 14.1K/15.9K, which are significantly lower than Google STT's 27.6K/30.7K, showcasing DASAM's closer predictions to true values.

c) **R-squared (R^2):** R^2 is a statistical measure that reflects the proportion of variance (spread) in the reference timings that can be explained by the predicted timings from the system. It essentially indicates how well the model's predictions capture the overall pattern of the actual timings. R^2 ranges from 0 to 1, with higher values closer to 1 signifying a stronger correlation between the predicted and actual timings. An R^2 value of 1 indicates that the model perfectly explains all the variance in the reference timings, while a value of 0 suggests no explanatory power.
With DASAM achieving 0.959/0.957 and Google STT at 0.870/0.849 for start and end times respectively, DASAM is shown to account for more variance in the dataset.

d) **Pearson Correlation Coefficient:** The Pearson Correlation Coefficient measures the linear correlation between the predicted and actual word timings. It indicates the strength and direction of the linear relationship between these two variables. Values range from -1 to +1.

- A coefficient of +1 indicates a perfect positive linear correlation, meaning that as the predicted timings increase, the actual timings also increase proportionally.
- A coefficient of -1 indicates a perfect negative linear correlation, where higher predicted timings correspond to lower actual timings.
- A value of 0 suggests no linear correlation between the predicted and actual timings.
- In the context of word-level alignment, values closer to +1 are desirable, indicating a strong positive correlation between the predicted and actual timings.

Both systems score highly, but DASAM slightly edges out with 0.981/0.982 against Google STT's 0.936/0.927, affirming the model's stronger predictive alignment.

2. Statistical Insights: Unveiling the Details in the Charts

This section presents a statistical charts analysis used to evaluate the DASAM model. These charts help illustrate the model's performance in handling audio variations compared to Google STT in word-level alignment. By examining these charts, we can gain deeper insights into the strengths and weaknesses of each system.

1. **Boxplot of Errors:** A boxplot is a visualization tool that summarizes the distribution of errors in a dataset. It depicts several key elements:
   - **Center line:** This line represents the median error value, which indicates the middle point when all errors are ordered from smallest to largest.
   - **Box:** The box encompasses the middle 50% of the data, with the bottom and top edges representing the first and third quartiles (Q1 and Q3), respectively. A narrower box signifies a tighter concentration of errors around the median.
   - **Whiskers:** These lines extend from the box and capture the remaining data points (outside the middle 50%). Longer whiskers indicate a wider spread of errors in the dataset.

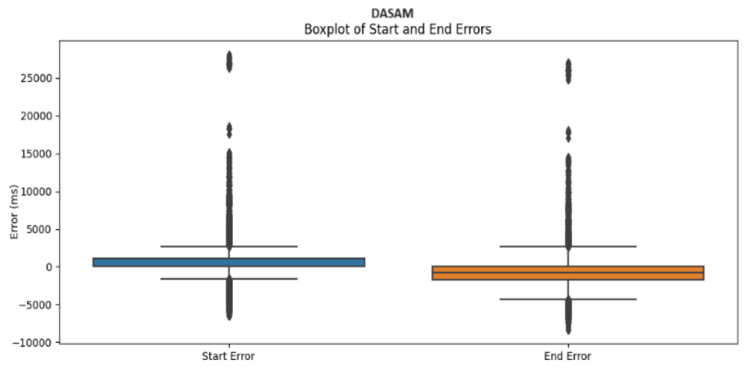

Figure 3: Distribution of Alignment Errors for DASAM: Showcasing the concentration of errors within a narrower range.

Figure 3, a boxplot of DASAM's errors, shows that errors are mostly concentrated between -10,000ms and 25,000ms. This tighter range suggests that DASAM's predictions are generally precise. Google STT has errors ranging from -20,000ms to 80,000ms (Figure 4). This wider spread indicates greater variability in prediction accuracy. A narrower boxplot for DASAM indicates more consistent performance compared to Google STT, particularly for start times.

2. **Visualizing Error Distribution with Histograms:**

The histogram visualizes the distribution of errors on the x-axis (error values) and the frequency of each error value on the y-axis (number of occurrences). A prominent peak around zero on the x-axis would indicate that many predictions are very close to the correct timings (low error values). A wider spread with a less pronounced peak suggests that errors are more scattered across a larger range.

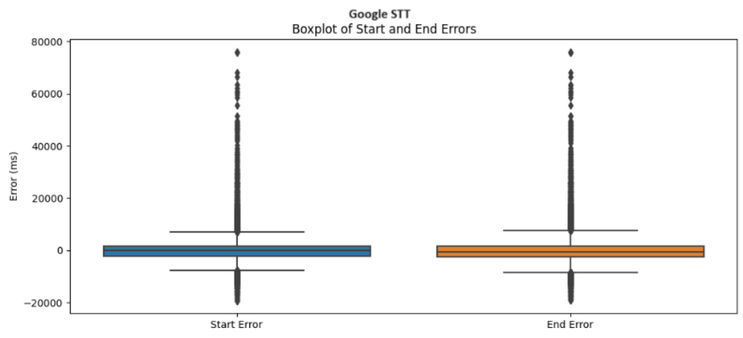

Figure 4: Distribution of Alignment Errors for Google STT: Illustrating wider variability in alignment errors.

Figure 5 illustrates a histogram of DASAM's errors, with a pronounced peak between 0ms and 8,000ms. This shows that most of DASAM's predictions are close to the actual timings. The histogram of Google STT (Figure 6) is more spread out with a peak from 0ms to 5,000ms.

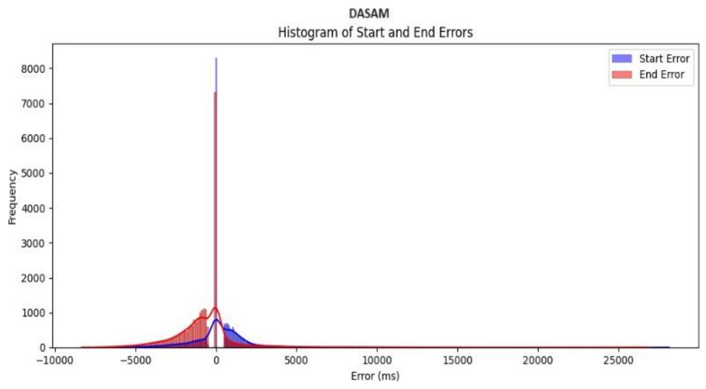

Figure 5: Histogram of DASAM's Alignment Errors: Peak concentration near zero error

This suggests that errors are more dispersed across a range. DASAM demonstrates a higher accuracy, with a significant number of predictions near the correct timing, indicated by the concentration of data around 0ms error.

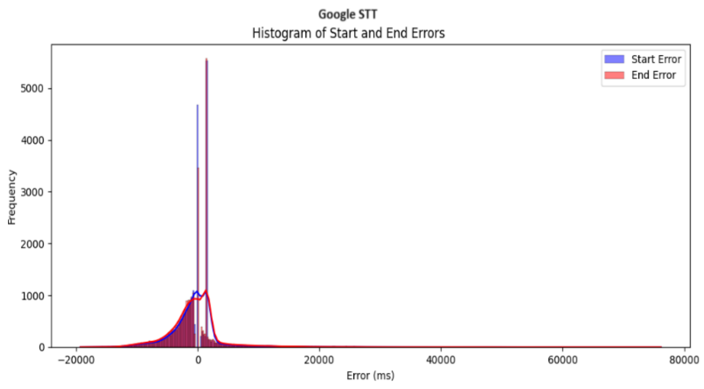

Figure 6: Histogram of Google SST's Alignment Errors: Less concentrated peaks indicating scattered errors

3. **Scatter Plot of Reference vs. Inferred Times:**

A scatter plot visually represents the relationship between the actual reference word timings and the system's inferred (predicted) timings. Each data point on the plot corresponds to a single word, with its horizontal position representing the reference time and its vertical position representing the inferred time. As shown in Figure 7, DASAM's data points are tightly clustered around the diagonal line, indicating high accuracy in predictions. Figure 8 shows a wider spread of data points for Google STT, especially for end times, indicating less accuracy. DASAM's predictions closely match the reference times, showing its superior accuracy in word alignment.

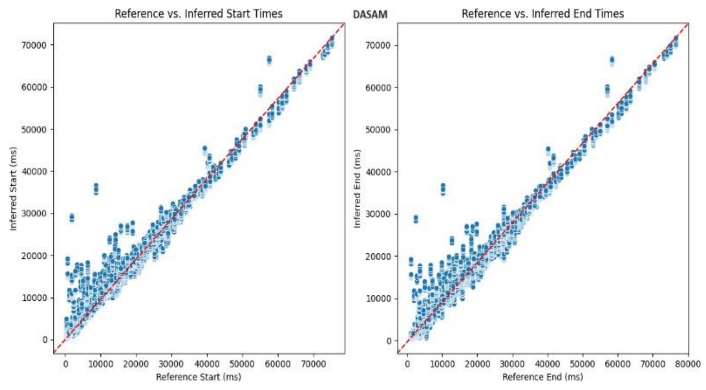

Figure 7: Scatter Plot of DASAM's Predictions: Scatter plot showing high alignment between predicted and reference timings.

4. **Unveiling Systematic Bias with Residual Plots:** A residual plot depicts the difference between predicted and actual values (residuals) on the y-axis and

the actual values (reference timings) on the x-axis. It helps identify any systematic bias in the errors.

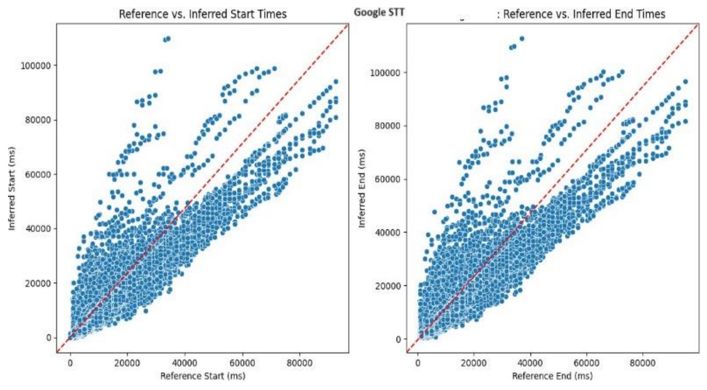

Figure 8: Scatter Plot of Google STT's Predictions: More spread indicating varied prediction accuracy.

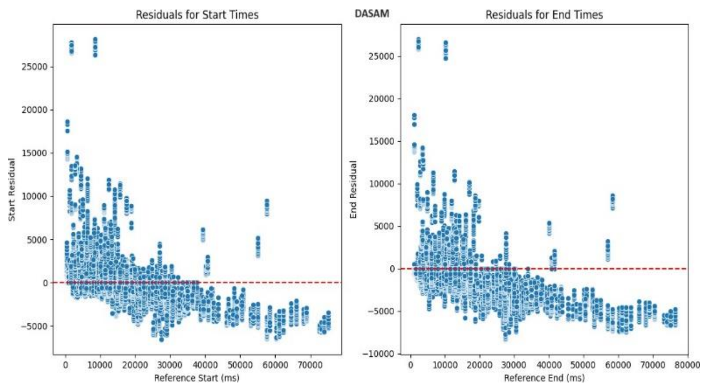

Figure 9: Residual Analysis for DASAM: Close clustering around zero indicating accurate predictions

In Figure 9, DASAM's residuals are clustered near the zero line, suggesting accurate predictions. Figure 10 illustrates that Google STT's residuals spread from -20,000ms to 80,000ms, indicating more significant deviations from true values. DASAM shows smaller and more consistent deviations from actual timings, demonstrating greater accuracy in word alignment.

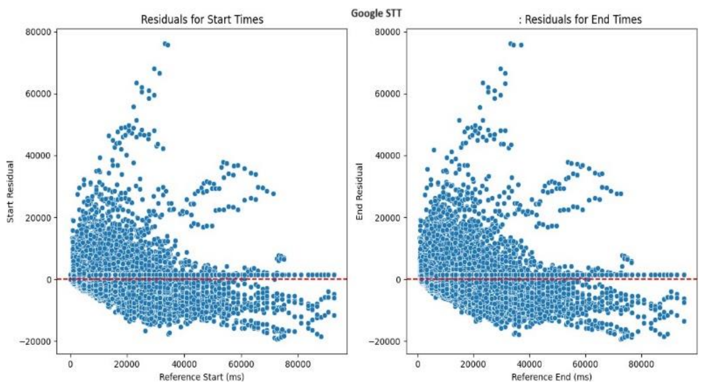

Figure 10: Residual Analysis for Google STT: Larger deviations from the zero-line shown in residuals

The residual plots (Figure 11) also reveal an interesting observation:

- **Anticipatory Behavior:** The residuals consistently below zero indicate the algorithms tend to proactively predict word times before they actually occur. This might be due to their sensitivity to subtle sounds like rising pitch at word beginnings or breath before pronunciation. Both Google ASR and DASAM exhibit this behavior.
- **Normal Distribution at Shorter Lengths:** In sentences under 40,000ms, recognition errors appear normally distributed, with both delayed and anticipatory predictions. Words at the beginning tend not to have a silent time before them.
- **Long Sentences and Prediction:** For sentences exceeding 40,000ms, anticipatory prediction errors persist, albeit less frequently, because there are fewer words present at this time, but delayed prediction errors decrease sharply.

This suggests that increased sentence length allows the algorithms to improve alignment accuracy and reduce late prediction, particularly for words towards the end. This could be due to the algorithms leveraging the additional information provided by longer sentences and the precise knowledge of sentence endings. Additionally, the distribution of more words over a longer timescale might facilitate their alignment.

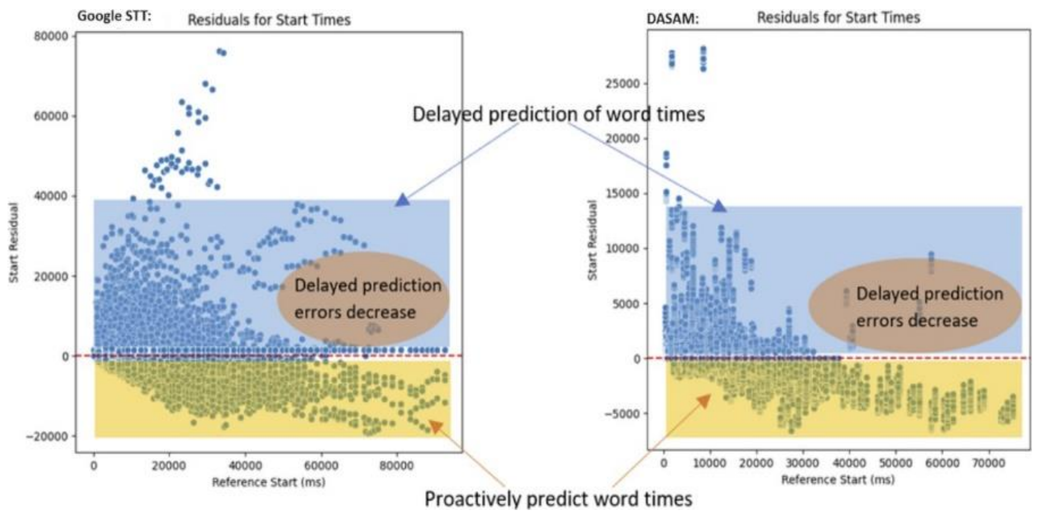

Figure 11: Interpreting Residual Plots: Unveiling patterns in prediction timing and systematic biases

5. **Trend of Scores:** These charts (Figures 12 & 13) depict the performance of both systems (DASAM and Google STT) across different audio variations. The X-axis typically represents the audio variations used for evaluation, while the Y-axis represents the corresponding performance score.

Figure 12 depicts fluctuations in Google STT's performance across different audio variations. Notably, there are dips in scores for variations 2, 5, 6, and 7. This suggests that these specific audio variations might pose greater challenges for Google ASR's word-level alignment capabilities.

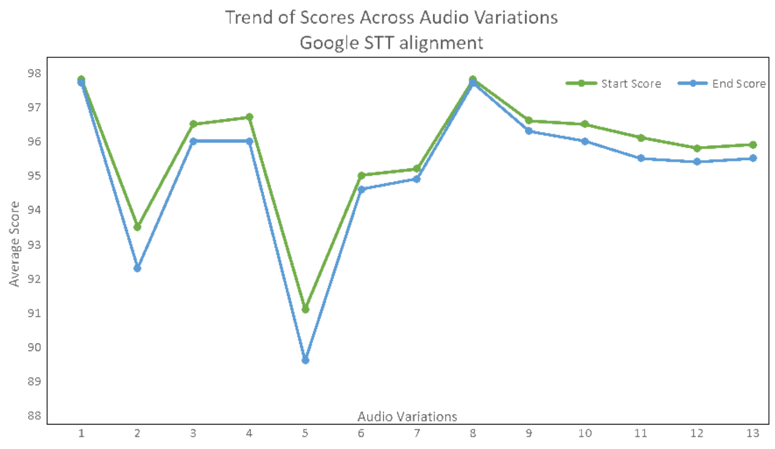

Figure 12: Google STT's Performance Trends.

The chart might appear to show a slight decrease in DASAM's start score from a high of 96.6 to 95.1, while the end score seems to increase inversely from 94 to 96.7.

DASAM surpasses Google STT in word-level alignment accuracy based on the comprehensive evaluation metrics employed in this study. The analysis of statistical charts further strengthens this conclusion by revealing DASAM's consistently high performance and robust error distribution. Additionally, the paper explores interesting observations regarding how sentence length influences alignment accuracy. Overall, DASAM presents itself as a promising system for various applications that require precise word-level alignment in audio recordings for Arabic with diacritics.

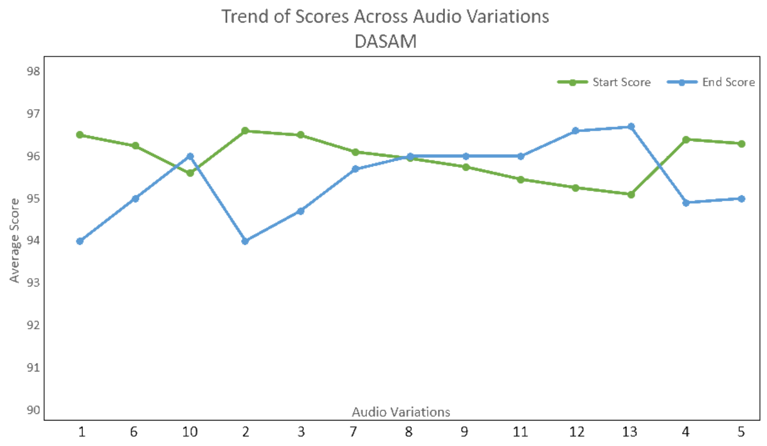

Figure 13: DASAM's Trend of Scores Across Audio Variations.

Limitations and Future Work

DASAM is not an STT like Google STT. Google STT has many functions. We compared DASAM to Google STT because both output words, start and end times. Although there are other systems for Segmentation and Alignment, we chose Google STT for comparison. This is because we could test and access it, unlike other systems. Our results show that DASAM outperformed Google STT in accuracy. DASAM's R^2 value is 0.959 for start time and 0.957 for end time. In comparison, Google STT's R^2 is 0.870 and 0.849, respectively. This is due to integrating intonation rules with Signal Processing. Muslims are more accurate in pronouncing Qur'anic letters. Thus, intonation rules are very successful here. However, they may not be as effective in environments with pronunciation errors. Further studies with different datasets are needed to test this. We created DASAM to enhance various Speech Recognition processes. In future work, we aim to develop a comprehensive system supporting speech-to-text and text-to-speech for Arabic with diacritics.

Conclusion

This part of the dissertation has evaluated the performance of DASAM, a Word-level Segmentation and Alignment Model for Arabic Diacritic-Based audio recordings. Our comparison with Google STT, a recognized benchmark in speech recognition, illustrates DASAM's superior accuracy. This superiority is supported by key metrics, most notably the R^2 values. DASAM's R^2 for start time is 0.959 compared to Google STT's 0.870, and for end time, DASAM scores 0.957 versus Google STT's 0.849. These values demonstrate that DASAM explains a greater proportion of variance in the actual word timings than Google STT, indicating higher accuracy and consistency in predictions.

The model also displays a higher concentration of predictions near the correct timings and exhibits less systematic bias compared to Google STT. Both systems demonstrated anticipatory behavior, likely reflective of their sensitivity to prespeech cues, which is a common characteristic in advanced speech recognition technologies. Additionally, longer sentences enhanced alignment accuracy, which is particularly beneficial for DASAM in reducing late predictions. The consistent performance of DASAM across various audio variations underscores its robustness and reliability for practical applications. Overall, this study establishes DASAM as a promising system for word-level alignment tasks, outperforming Google STT in accuracy and consistency, specifically for Arabic Diacritic-based audio recordings under the tested conditions.

Acknowledgement

I would like to thank Prof. Wei Cheng for his valuable contributions and insightful comments during this study. His expertise and feedback have greatly enhanced the quality of this research, and I appreciate his support.

References

Abdo, M. S., & Kandil, A. H. (2016). Semi-automatic segmentation system for syllables extraction from continuous Arabic audio signal. International Journal of Advanced Computer Science and Applications, 7(1).

Abdo, M. S., Kandil, A. H., & Fawzy, S. A. (2014, February). MFC peak based segmentation for continuous Arabic audio signal. In 2nd Middle East Conference on Biomedical Engineering (pp. 224-227). IEEE.

Aboalnaser, S. A. (2019, October). Machine learning algorithms in arabic text classification: A review. In 2019 12th international conference on developments in esystems engineering (dese) (pp. 290-295). IEEE.

Absa, A. H. A., Deriche, M., Elshafei-Ahmed, M., Elhadj, Y. M., & Juang, B. H. (2018). A hybrid unsupervised segmentation algorithm for arabic speech using feature fusion and a genetic algorithm (July 2018). Ieee Access, 6, 43157-43169.

Al-Fadhli, S., Al-Harbi, H., & Cherif, A. (2023). Speech Recognition Models for Holy Quran Recitation Based on Modern Approaches and Tajweed Rules: A Comprehensive Overview. International Journal of Advanced Computer Science & Applications, 14(12).

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

Alkayyali, Z. K., Idris, S. A. B., & Abu-Naser, S. S. (2023). A New Algorithm for Audio Files Augmentation. Journal of Theoretical and Applied Information Technology, 101(12).

Amazon Web Services. (2024). Supported languages and language-specific features - Amazon Transcribe. Retrieved from https://docs.aws.amazon.com/transcribe/latest/dg/supported-languages.html.

Bakır, H., Çayır, A. N., & Navruz, T. S. (2024). A comprehensive experimental study for analyzing the effects of data augmentation techniques on voice classification. Multimedia Tools and Applications, 83(6), 17601-17628.

Balula, N. O. M., Rashwan, M., & Abdou, S. (2021). Automatic speech recognition (ASR) systems for learning Arabic language and Al-quran recitation: a Review. International Journal of Computer Science and Mobile Computing, 10(7), 91-100.

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). International Speech Communication Association.

Deriso, D., & Boyd, S. (2023). A general optimization framework for dynamic time warping. Optimization and Engineering, 24(2), 1411-1432.

El-Imam, Y. A. (2004). Phonetization of Arabic: rules and algorithms. Computer Speech & Language, 18(4), 339-373.

Google Cloud. (2024). Cloud Speech-to-Text API - Language support. Retrieved from https://cloud.google.com/speech-to-text/docs/speech-to-text-supported-languages.

Javed, M., Baig, M. M. A., & Qazi, S. A. (2020). Unsupervised phonetic segmentation of classical Arabic speech using forward and inverse characteristics of the vocal tract. Arabian Journal for Science and Engineering, 45, 1581-1597.

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

Microsoft Azure. (2024). Language support for Computer Vision-Azure Cognitive Services [Note by Microsoft]. Retrieved from https://learn.microsoft.com/en-us/azure/ai-services/computer-vision/language-support#optical-character-recognition-ocr.

Permanasari, Y., Harahap, E. H., & Ali, E. P. (2019, November). Speech recognition using dynamic time warping (DTW). In Journal of physics: Conference series (Vol. 1366, No. 1, p. 012091). IOP Publishing.

Qasim, H., & Abdulbaqi, H. A. (2022, October). Arabic speech recognition using deep learning methods: Literature review. In Aip conference proceedings (Vol. 2398, No. 1). AIP Publishing.

Rahman, A., Kabir, M. M., Mridha, M. F., Alatiyyah, M., Alhasson, H. F., & Alharbi, S. S. (2024). Arabic Speech Recognition: Advancement and Challenges. IEEE Access.

Sabour, A., & Ali, M. (2024). Quran Research. Retrieved from https://quranresearch.org/ (Accessed: March 1, 2024).

Sabour, A., Hendawi, A., & Ali, M. (2023). Diacritic-Aware Alignment and Classification in Arabic Speech: A Fusion of Fuztpi and ML Models. JISTech (Journal of Islamic Science and Technology), 8(2), 169-191.

Stan, A., Mamiya, Y., Yamagishi, J., Bell, P., Watts, O., Clark, R. A., & King, S. (2016). ALISA: An automatic lightly supervised speech segmentation and alignment tool. Computer Speech & Language, 35, 116-133.

Sundus, K., Al-Haj, F., & Hammo, B. (2019, October). A deep learning approach for arabic text classification. In 2019 2nd international conference on new trends in computing sciences (ictcs) (pp. 1-7). IEEE.

Wei, G., Duan, Z., Li, S., Yu, X., & Yang, G. (2023). LFEformer: Local Feature Enhancement Using Sliding Window with Deformability for Automatic Speech Recognition. IEEE Signal Processing Letters, 30, 180-184.

Yalova, K., Babenko, M., & Yashyna, K. (2023). Automatic Speech Recognition System with Dynamic Time Warping and Mel-Frequency Cepstral Coefficients. In COLINS (2) (pp. 141-151).

Zheng, F., Zhang, G., & Song, Z. (2001). Comparison of different implementations of MFCC. Journal of Computer science and Technology, 16, 582-589.

Zheng, J., Franco, H., & Stolcke, A. (2003). Modeling word-level rate-of-speech variation in large vocabulary conversational speech recognition. Speech Communication, 41(2-3), 273-285