Arabian Journal for Science and Engineering (2019) 44:9043–9056
https://doi.org/10.1007/s13369-019-04024-0

RESEARCH ARTICLE - COMPUTER ENGINEERING AND COMPUTER SCIENCE

Diacritics Effect on Arabic Speech Recognition

Sa'ed Abed[1] · Mohammad Alshayeji[1] · Sari Sultan[1]

Received: 19 September 2018 / Accepted: 2 July 2019 / Published online: 10 July 2019

Abstract
Arabic is the native language for over 300 million speakers and one of the official languages in United Nations. It has a unique set of diacritics that can alter a word's meaning. Arabic automatic speech recognition (ASR) received little attention compared to other languages, and researches were oblivious to the diacritics in most cases. Omitting diacritics circumscribes the Arabic ASR system's usability for several applications such as voice-enabled translation, text to speech, and speech-to-speech. In this paper, we study the effect of diacritics on Arabic ASR systems. Our approach is based on building and comparing diacritized and nondiacritized models for different corpus sizes. In particular, we build Arabic ASR models using state-of-the-art technologies for 1, 2, 5, 10, and 23 h. Each of those models was trained once with a diacritized corpus and another time with a nondiacritized version of the same corpus. KALDI toolkit and SRILM were used to build eight models for each corpus that are GMM-SI, GMM SAT, GMM MPE, GMM MMI, SGMM, SGMM-bMMI, DNN, DNN-MPE. Eighty different models were created using this experimental setup. Our results show that Word Error Rates (WERs) ranged from 4.68% to 42%. Adding diacritics increased WER by 0.59% to 3.29%. Although diacritics increased WERs, it is recommended to include diacritics for ASR systems when integrated with other systems such as voice-enabled translation. We believe that the benefit of the overall accuracy of the integrated system (e.g., translation) outweighs the WER increase for the Arabic ASR system.

Keywords Automatic speech recognition (ASR) · Arabic ASR · Arabic diacritics · Word error rate (WER) · KALDI toolkit · SRILM

1 Introduction

Automatic speech recognition (ASR) is the automated transcription of speech into text. Over the last decade, ASR technologies emerged in different areas such as personal computers, mobile phones, security systems, health, robotics, military, education, dictation, among others. ASR has a promising future because speech is one of the simplest ways of communication that avoids complex user interfaces. In addition, ASR systems are under active development and adopted by a wide range of applications due to their functionality and simplicity. For instance, it is used in customer care applications where a user can interact with a voice-enabled service instead of human interactions. This helps with serving higher number of customers and reduce lengthy service queues. Another example is using ASR in smart homes to control heating, lighting, and other appliances. We are witnessing an increasing adoption and development of ASR systems that is expected to continue to grow in the future.

Figure 1 shows a simple ASR system architecture. Input signal processing and feature extraction steps include noise filtration, converting the input signal to frequency domain, and extracting the signal's feature vectors. For feature extraction, most of ASR systems use mel-frequency cepstral coefficient (MFCC) [1] or relative spectral transform-perceptual linear prediction (RASTA-PLP) [2]. The next stage is decoding, it uses statistical models such as hidden Markov model (HMM) or other techniques such as dynamic time wrapping (DTW) [3]. The acoustic model (AM) generates an AM score using the extracted features components. A hybrid of HMM and Gaussian mixture model (HMM-GMM) is widely used for acoustic modeling. Recently, a hybrid of HMM and deep neural network (HMM-DNN) is replacing the HMM-GMM

✉ Sa'ed Abed
s.abed@ku.edu.kw

Mohammad Alshayeji
m.alshayeji@ku.edu.kw

Sari Sultan
SariSultan@ieee.org

[1] Computer Engineering Department, Kuwait University, P.O. Box 5969, 13060 Safat, Kuwait

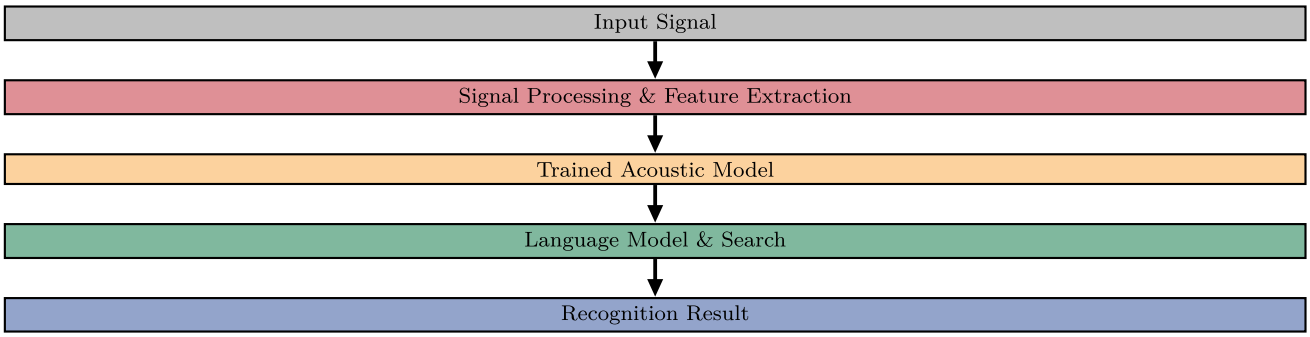

Fig. 1 Architecture of ASR system

approach because it became more feasible and significantly reduced error rates owing to the perpetual enhancement in the computational power and availability of large training sets. It achieved one-third error rates reduction compared to conventional HMM-GMM models [4–7]. The language model (LM) estimates the probability of the proposed word sequence by learning the relation between words from a training corpus [4]. LMs are usually statistical n-grams models that estimate the distribution of a language. Finally, the system will show the recognized word(s). Several open-source tools are available to help researchers to build and test ASR systems such as HTK [8], CMU Sphinx [9], and KALDI [10]. Several tools are also available for language modeling such as SRILM [11], CMU Sphinx LM tool [9], and KALDI LM tool [10].

Arabic is one of the world's most common languages and one of the six official languages of United Nations (UN) [12,13]. It is the fourth most spoken language after Mandarin, Spanish, and English. Arabic has a special set of diacritics that change letters' pronunciation and a word's meaning. Research on Arabic ASR is limited compared to other languages such as English and Mandarin. Based on our literature review, we noticed that researchers usually omitted diacritics for two primary reasons. First, adding diacritics increases the Word Error Rate (WER) for the AM and increases the LM's perplexity. Second, Arabic readers can understand a word's meaning from its context without diacritics in most cases. However, diacritics are crucial is some cases where a words' meaning cannot be discerned from the context. Researchers that applied diacritics to their Arabic ASR systems found that they always increased WERs [14–18]. Although the aforementioned reasons are correct, omitting diacritics will limit the usability of the ASR system for several applications such as voice-enabled translation.

To the best of our knowledge, no studies addressed nondiacritized Arabic ASR systems' performance when they are integrated with voice-enabled service such as translation. This can be problematic because ASR systems are not always standalone systems as they might be integrated with other services that depends on the ASR system's output. The problem is in the assumption that the Arabic ASR system's output will be used by Arabic human readers, which is not always the case where the ASR system could be integrated with others systems such as voice-enabled translation. For instance, (عِلْم) means science, (عَلِمَ) means he knew, (عَلَّمَ) means he taught, (عَلَم) means a flag. A word can be recognized by ASR to confer one meaning, spoken by TTS with a second meaning, and translated using machine translation to have a third meaning.

In this paper, we study the effect of diacritics on Arabic ASR systems and discuss how they affect other integrated systems. We build 80 different ASR models for 1, 2, 5, 10, 23 h corpora using state-of-the-art technologies. We used KALDI toolkit and SRILM to build eight models for each corpus: GMM-SI, GMM SAT (fMMLR), GMM MPE, GMM MMI, SGMM, SGMM-bMMI, DNN, and DNN-MPE. The first six models were trained using CPU and the last two were trained using GPU. Each model is trained with and without diacritics. A total of 80 models is created. We used phoneme-based modeling approach as it is widely used for Arabic language [15,19–21], and to help to compare our results with other researchers such as [15,22].

The scope of this work is focused on comparing our models' WERs for the diacritized and nondiacritized versions as well as comparing our results with other researchers results. Integrating the acoustic models with other systems such as TTS or translation is considered for future work. Our results show that WERs ranged from 4.68% to 42%. Including diacritics increased the WER by 0.59% to 3.29% compared to the nondiacritized version of the same corpus.

In this context, the primary contributions of this paper are the following.

- Study diacritics effect on Arabic ASR systems.
- Develop Arabic ASR models using state-of-the-art technologies with and without diacritics to analyze the effect of diacritics.
- Provide recommendations when diacritics should be used in ASR systems and when to be excluded.
- Investigate the possible effects of Arabic diacritics on integrated systems that depend on the Arabic ASR

system's output such as voice-enable translation and speech-to-speech.

The rest of the paper is organized as follows. Section 2 discusses the background material. The related work is discussed in Sect. 3. In Sect. 4, we formulate the problem we are trying to solve, the questions we are trying to answer, and the open problems that could be tackled in future work. Section 5 discusses our research methodology. In Sect. 6, we present the experimental results and recommendations. Finally, Sect. 7 concludes this paper some future research directions.

2 Background

In this section, we provide background material on ASR systems. In particular, Sect. 2.1 discusses the ASR types. Section 2.2 discusses the performance metrics and evaluation of ASR systems.

2.1 ASR Types and Mechanisms

ASR systems have two types: (i) discrete word and (ii) continuous speech recognition. Each type is divided into two categories. First, speaker dependent where the system learns the unique characteristics of a person's voice. New users should train the system by speaking certain words or paragraphs first. Second, speaker independent, the aim of these systems is to recognize anyone's voice for a specific language. No training is needed by the end user before using the system. It is usually used by interactive applications where the business needs limit asking the user to read few sentences or pages before the speech recognition starts. This kind of systems are usually pre-trained with large variety of recordings. Training is performed using a statistical models such as HMM.

Speaker-dependent systems achieve better WERs than speaker-independent systems because the former is adapted to an individual user, which can be a factor of two or three times lower than a speaker-independent system when using the same amount of training data [23]. Speaker-independent systems might struggle when a new user uses the system. It is not practical to train the system again for each new user. Thus, speech adaptation techniques are used to alleviate this burden by quickly tuning the system parameters to the new user. This leads to speaker adaptive training (SAT), which produces smaller models estimated variances, higher training set likelihoods, and improved results compared to trained systems without adaptation. However, it requires higher amount of storage and more computation power [23–25]. Several techniques are available for speaker adaptation such as maximum a posteriori parameter estimation (MAP) [26] and maximum likelihood linear regression (MLLR) [27]. Reviews of speaker adaptation techniques are presented in [23,28].

We study some of the methods used for speech recognition in this section. First, DTW is used to find the best alignment between two time series so that a time series can be changed nonlinearly by widening or decreasing it along its time axis (i.e., wrapping) [29]. Wrapping can then be used to calculate the corresponding regions to find the similarity between them. In speech recognition, DTW is used to decide which waveform represents the spoken phrase. The scheme is used to calculate the difference of the time adjustment of two words. DTW has some limitations such as high complexity of $O(n^2v)$, hard to evaluate two elements from two distinct sequences. It is suitable for simple applications [29].

Second, HMMs are finite state machines where transition between states is made instantly at equal periods (clocks). With every movement, the system produces observations where two processes are taking place: the transparent process, characterized by the observation string (feature sequence), and the hidden process characterized by the state string. The main issues of this technique are the sequence of time, the comparison methods, the assumption that successive observation are independent, constant length observation frames, and large number of parameters [30–37]. Most systems use a hybrid of HMM and GMM to build their ASR systems. HMMs are used to determine the temporal variability of speech while GMMs are used to determine how HMM states fits the frames acoustic input [38].

Third, as alternative to GMMs, artificial neural networks (ANNs) are used in many applications due to their powerful parallel-distributed memories and processing, fault constancy, and distinguished pattern learning ability [39]. The complexity of ANNs increases as their generality rises. A deep neural network (DNN) is an ANN with multiple hidden layers between the input and output layers [6]. DNN generates compositional models, where extra layers enable a composition of features from lower layers, adding a big learning ability and increase the potential of modeling complex patterns of speech data [40]. DNNs showed an impressive success in large vocabulary speech recognition since 2010 where large output layers of the DNN based on context-dependent HMM states. Large-scale ASR is considered the most convincing successful case of deep learning in the recent history [5,41,42]. Disadvantages of DNN compared to GMM includes that it is hard to utilize large clusters to train them on large datasets [38]. Table 1 shows a comparison of the aforementioned types.

2.2 Performance Metrics and Tools

Accuracy and speed are important factors in ASR systems. Since the last two decades, systems kept benefiting from Moore's law validity using the continuous increase in com-

Table 1 DTW, GMM, and DNN comparison

Technique	Advantages	Disadvantages
DTW	Works well with small datasets	Deteriorate with large datasets
	Speaker specific	Comparing sample against stored word patterns and decide the best match is difficult due to many reasons: synchronization, symmetrical distance, among others
	Efficient in time-series similarity measurement to reduce the shift and time alteration influence	For large dictionaries, the O(n2v) complexity increases delays tremendously
HMM-GMM	Computationally feasible	Evaluation problem
	Simple for training algorithms to determine the model parameters compared to fixed speech data training sets	Hidden state determination (decoding)
	Simplicity to change the model to be suitable for specific words	Number of parameters is highly sensitive to input feature dimension [63]
	Simple to implement in ASR systems	Vulnerable to parameter estimation problem with high input feature dimension, which deteriorates the performance [63]
HMM-DNN	Excellent pattern recognition capabilities	Parallelization
	Efficient in categorizing short-time units mainly in isolated words	Hard to be used on large clusters of machines
	Lower WER than other techniques	Requires more processing power (usually require GPUs)
	Number of parameters is not highly sensitive to input feature dimension [63]	
	Less vulnerable to parameter estimation problem with higher input feature dimension [63]	
	Needs smaller datasets to get similar performance of GMM [38]	

puting resources. An inverse relationship exists between the system's accuracy and speed, as the former demanded to be higher, maintaining a fast recognition rates becomes a challenging task. High-speed speech recognition plays a pivotal role in real-time ASR systems. Speech decoding is one of the most computational and time-consuming phases of speech recognition process. The ASR delay should be less than a tenth of a second in order to make the user feel the system is reacting instantaneously [43]. Such real-time systems are beneficial for numerous applications such as live transcription and speech-to-speech systems that made the fictional idea of the global speaker become true as recently published by Microsoft research [44]. ASR faces several challenges that reduce its performance and accuracy such as the following.

- Vocabulary and nature of the alphabet: Language that has a large vocabulary increases errors. In general, it is easy to distinguish a small word set, but when the size grows the error rates increase. Arabic language has a very rich vocabulary that make speech recognition more challenging.
- Speech recognition has to select between discrete and continuous speech. In software tools, it is easier to recognize words when they are pronounced separately. Continuous speech is harder to detect because words pronunciation can be vague. Understanding a paragraph is more challenging than understanding a sentence or a word.
- Adverse conditions: Performance of ASR systems may vary due to environmental noise, sound distortions (e.g., echo), and speaking style (e.g., fast or slow). Accuracy depends on many conditions and the surrounding environment. Human voice has nonlinear properties influenced by many parameters such as gender, background and emotions. Thus, languages can vary in terms of pronunciation, accent, volume, speech, and speed. All of these factors increase the ASR system's complexity.
- Homonyms: Homonym is defined as two words with different semantics but sound similar such as "hair" and "her," "cup" and "cop." An ASR system cannot distinguish between these words based on their sound only. Therefore, several training systems consider the context, which has great impact on ASR system's performance.
- A limitation in grammatical language: The presence of diacritics changes the meaning of spoken words. For instance, كُتِبَ means written while كُتُبْ means books. Arabic language has 28 letters. Each letter can be pronounced differently with different diacritic marks, which are Harakat (Fatha, Dammah, Kasrah), Tanween (Fath, Dam, Kasr), shadda, mad, and sukun. Table 2 shows the Arabic alphabet and its correspondent IPA phonemes.

Table 2 IPA phonemes representation for Arabic letters [14]

Isolated	Beginning	Middle	End	Name	Phoneme
ا	ا	ا	ا	'alif	
ب	بـ	ـبـ	ـب	baa'	
ت	تـ	ـتـ	ـت	taa'	
ث	ثـ	ـثـ	ـث	thaa'	
ج	جـ	ـجـ	ـج	gym	
ح	حـ	ـحـ	ـح	Haa'	
خ	خـ	ـخـ	ـخ	khaa'	
د	د	ـد	ـد	daal	
ذ	ذ	ـذ	ـذ	dhaal	
ر	ز	ـز	ـز	raa	
ز	ر	ـر	ـر	zayn	
س	سـ	ـسـ	ـس	syn	
ش	شـ	ـشـ	ـش	shyn	
ص	صـ	ـصـ	ـص	Saad	
ض	ضـ	ـضـ	ـض	Daad	
ط	طـ	ـطـ	ـط	Taa'	
ظ	ظـ	ـظـ	ـظ	Zaa'	
ع	عـ	ـعـ	ـع	'ayn	
غ	غـ	ـغـ	ـغ	ghayn	
ك	كـ	ـكـ	ـك	kaaf	
ق	قـ	ـقـ	ـق	qaaf	
ف	فـ	ـفـ	ـف	faa'	
ل	لـ	ـلـ	ـل	laam	
ن	نـ	ـنـ	ـن	nuwn	
م	مـ	ـمـ	ـم	mym	
ه	هـ	ـهـ	ـه	haa'	
و	و	ـو	ـو	waaw	
ي	يـ	ـيـ	ـي	yaa'	
ء	أ	ـئـ	ـئ	hamza	

Kaldi speech recognition toolkit is a free and open-source toolkit for ASR [45]. Many researchers used Kaldi tool in their research owing to its ease of use, documentation, and available scripts. Kaldi has important features such as integration with finite state transducers (build against OpenFST toolkit [46]), open license, and complete recipes for building speech recognition systems. It supports various types of acoustic modeling such as GMMs and DNNs.

Figure 2 shows a simple representation for the architecture of an HMM-based system. Each part of the system will be discussed in details, some of the parts are summarized from [47]. The feature extraction phase takes the input audio waveform and coverts it into a fixed size feature vectors (Y) that are compact and efficiently represents the input signal, $\boldsymbol{Y_{1:T}} = \boldsymbol{y_1}, \ldots, \boldsymbol{y_T}$. The decoder tries to find a sequence of words $w_{1:L} = w_1, \ldots, w_L$ that most likely have generated Y as in Eq. 1. Some systems models this directly as in discriminative models [48]. However, since it is difficult to model in most cases, Bayes rule is used to transform Eq. 1 to an equivalent problem as in Eq. 2. The AM determines $P(Y|w)$ and $P(w)$ is determined by the LM.

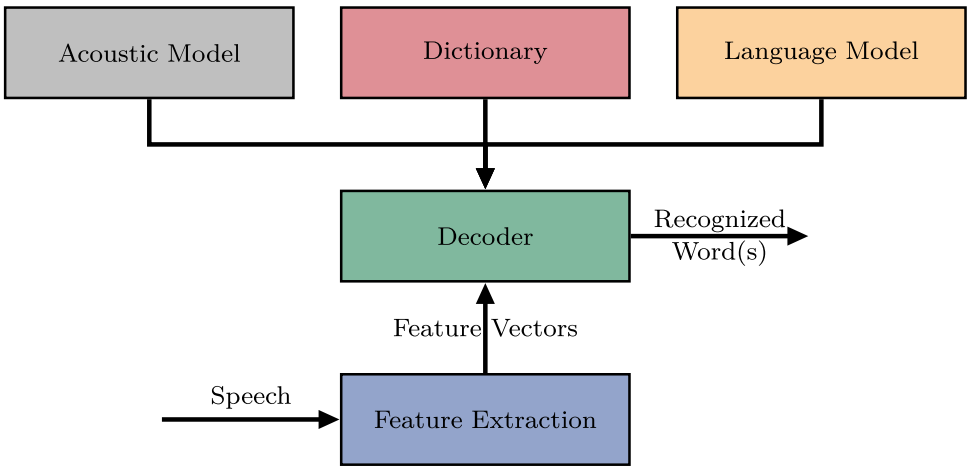

Fig. 2 Architecture of a HMM-based Recognizer

$$\hat{w} = \underset{x}{\operatorname{argmax}}\{P(w|Y)\} \tag{1}$$

$$\hat{w} = \underset{x}{\operatorname{argmax}}\{P(Y|w)P(w)\} \tag{2}$$

Speech is non-stationary signal. Thus, the extracted voice features should be estimated for short intervals, such as 10-30 ms, in which speech can behave stationary. Several techniques are available for feature extraction such as linear predictive coding (LPC) [49], mel-frequency ceptral coefficients (MFCC) [1], and perceptual linear prediction (PLP) [2]. MFCC is the predominant technique in the literature. Figure 3 shows the MFCC steps. It starts with framing, in which it uses a specific frame length (e.g., 10 ms) with overlapping window (e.g., 25 ms). Overlapping window is used to effectively extract information of two adjacent windows. Hamming window is frequently used in the windowing step. Then, Fast Fourier Transformation (FFT) is applied to the window to get its frequency content. The generated frequencies are then filtered by a Mel-scale filter that imitates the human ear. It uses frequency filters spaced linearly at low frequencies and logarithmically at high ones. Finally, the Discrete Cosine Transform (DCT) is applied to the logarithm of the filtered frequencies. This usually uses first N coefficients as 13 for the feature vectors. The extracted feature vectors are used in the next steps of speech recognition.

3 Related Work

In this section, we discuss related studies on Arabic ASR. We first address similar studies to our work. Then, we address other related topics for readers to better understand the holistic view of research on the area.

Ali et al. [20] developed an Arabic ASR system using KALDI toolkit. To the best of our knowledge, the authors are among the few researchers who addressed DNN in Arabic ASR. The authors used broadcast news corpus that contains 200 h of training data, which consisted of news conversations and reports (i.e., GALE phase 2 corpus). They achieved a best-case WER of 15.81% for reports, and WER of 32.21% for conversations, and overall combined WER of 26.95%. They used phoneme-based modeling because it is superior to grapheme-based systems. They used MFCC features without energy combined with a standard 13-dimensional cepstral mean-variance normalized (CMVN). Their models are three-state context-dependent models. Regarding the LM and the lexicon, they started by normalizing the selected Arabic text sources to remove common Arabic mistakes. They used an nondiacritized lexicon, LM, and Buck Walters transliteration for transcriptions. The lexicon is collected from news archive through collaboration with Aljazeera news. Best results were achieved using DNN combined with minimum phone error (MPE) technique. We used similar setup as proposed by the authors because it covers wide variety of state-of-the-art models that are used since the last decade, but we used different corpus because they use a nondiacritized one.

Abushariah et al. [16,17] conducted three experiments with and without diacritics on a training set of about 8 h. In all of their results, the nondiacritized systems achieved better WER by 0.33–1.2%. They used a high-performance design for ASR using CMU Sphinx toolkit. MFCC was used for the feature extraction. The system was trained with 7 h of balanced Arabic speech corpus, and tested using 1 h for a similar speaker but with different sentences. They obtained a WERs of 11.27–10.07%, with different speakers and similar sentences the system obtained WERs of 5.78–5.45%, and with different speakers and different sentences the WERs were 15.59–14.44%. Abushariah et al. [18] also achieved an enhancement of 1.31% in WERs when used an nondiacritized ASR system compared to diacritized one. Biadsy et al. [50] tested whether diacritization is important or not. They used Arabic dialect for their experiment of around 230 h with fully diacritized lexicon and LM. The fully diacritized

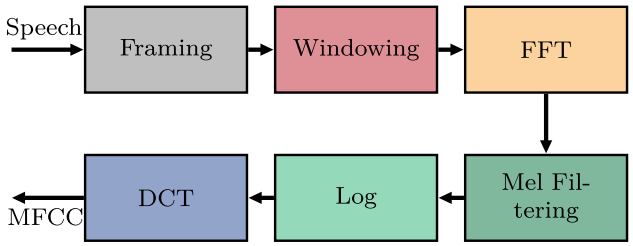

Fig. 3 MFCC Steps

system achieved 29.9% WER, and the nondiacritized preceded with 24.6%, that is 5.3% enhancement. They used an automatic diacritizer, which introduces a new error that could have contributed to the increased WER. Our corpus, however, is manually diacritized.

Kirchhoff et al. [14] underscored the importance of diacritics for Arabic language. They conducted their experiment on the LDC Callhome corpus using with diacritics (they called it romanized) and without diacritics (they called it script). The script version outperformed the romanized one by 4.1%. Moreover, the authors stated that the results would be better by using manually romanization instead of automatic one since the overall error rate would be reduced. Others said diacritics had no noticeable effect on the Arabic ASR system's WER [51].

Satori [36] presented an Arabic ASR system based on CMU Sphinx 4 using the HMM. Hyassat and Zitar [19] developed an Arabic ASR system for the Holy Quran using CMU Sphinx 4. It has three types of corpus: the holy Quran corpus, Arabic commands corpus, and Arabic digits corpus. Alghamdi et al. [15] developed an Arabic broadcast news transcription system using CMU Sphinx, with a corpus of 7 h for training and 0.5 h for testing. They achieved a WER of 8.61%.

Vergyri et al. [52] studied using morphology-based language models for Arabic in several phases of ASR systems. Two models were tested with an N-best list-rescoring platform: Class-based and single-stream factored language models (FLM). There were some difficulties using morphology-based LMs in an LVCSR system for Arabic language, which increases the WER. If the factored word representations cannot be decoded, then the FLM potential cannot be fully realized. This is considered as the main disadvantage of FLM. Soltau et al. [53] addressed the vowels in Arabic language. They tested a large vocabulary size consisted of 129,000 word, in which they achieved 19.8% WER. When they increased the size to 589,000 words, they achieved a WER of 18.3%.

Recently, several studies focused on Arabic dialects. This is an important topic because the MSA is not used anymore in normal life and it is mainly used in formal situations and news. Djellab et al. [54] proposed a linguistic overview for Algerian dialect and introduced a new corpus that is a two years efforts of collecting phone conversations from different areas in Algeria. The authors also studied recent recognition approaches such as GMM-UBM and i-vector to assess the effectiveness of such mechanisms for dialect recognition. Ali et al. [21] introduced a dialectical ASR challenge for the Egyptian dialect (MGB-3). The corpus for this challenge encompasses Egyptian YouTube videos from several genres such as comedy, cooking, family, sports, and drama. The length of the corpus is 16 h that is split evenly between those genres. The challenge focuses on Arabic dialect recognition to distinguish between different dialects such as Egyptian, Levantine, North African, Gulf, and MSA.

Mohammed and Khidhir [55] developed a real-time Arabic speaker-independent ASR system. They used MATLAB to identify some Arabic letters by developing a fast and simple technique. The technique used MFCC for feature extraction and Euclidean distance to compare the sounds with the used dataset. The authors achieved 89.6% recognition rate.

Furtună [29] discussed how the voice recognition for group of words requires comparing voices entered with the dictionary. A dynamic comparison method was used to allocate an optimal correspondence to the temporal scales of the two words. DTW was found to be effective for isolated words recognition in a limited dictionary. A major limitation of DTW is the complexity O(n2v), which may not be satisfactory for a larger dictionary. Moreover, it was difficult to evaluate two features from two different series. Alkhatib et al. [56] built a mobile application that focuses on detecting mispronounced Arabic words and further guide the reader for the correct pronunciation. The authors used MFCC and DTW algorithms in building their system.

Although software tools can be suitable for real-time performance, many reasons motivate using hardware approaches to achieve better performance. Some applications used in call centers require handling a large number of concurrent requests Gorin et al. [57]. Other application like off-line transcription or dictation require handling multiple speech streams simultaneously and may offer an important economic gain. Field programmable gate array (FPGA) can be used for real-time applications due to its processing power. Several researchers discussed implementing hardware speech recognition system to gain better recognition speeds as well as training time. Software- and hardware-based systems have pros and cons over each other, and no one based on our literature tried to make a hybrid system to benefit from each part. On the one hand, the software-based systems are more user friendly, which can produce more functionality. On the other hand, hardware approaches as FPGA might provide better performance due to its processing power. However, hardware recognizers have several limitations for speech recognition such as the following Price et al. [58].

Table 3 Diacritics effect on word's meaning

Meaning	Word
Wrote	كَتَبَ
Has been written	كُتِبَ
Books	كُتُبْ

Table 4 Diacritics effect on word's meaning, complex example

Meaning	Word
If	إنْ
Suffer	أَنَّ
Sufferer	آنٌ
The time has come	آنَ
The appropriate time to do something	آنُ

– Cannot be easily reprogrammed as the speech recognition algorithms is continuously evolving.
– The required memory size and bandwidth for speech recognition are high.

Few researches implemented Arabic ASR systems using FPGA. Elmisery et al. [59] published a hardware and software comparison for Arabic ASR system using the HMM. The FPGA recognizer showed better performance as six times faster than the software implementation. This is a promising result that motivates researchers to work in hardware implementations instead of the predominant software applications.

4 Problem Definition

The Arabic language is one of the morphologically rich languages. Its diacritics play an important role in defining a word's meaning. Table 3 shows a simple example of diacritics effects on the word's meaning (this word can have nine meanings as shown by [14], but we used the three most common uses). A more complex example is shown in Table 4, in which a two letters word can have five meanings by changing the diacritics.

Based on our literature review, diacritics are being omitted from Arabic ASR systems. This is for two main reasons. First, using diacritics in ASR increases the AM's WER and LM's perplexity. Second, Arabic reader can discern a word's meaning from its context in most cases. This delegates the diacritization process to the user who is reading the recognized text. Although this process might be carried on by a native speaker naturally, it is still not decisive and does not guarantee the exact intended results. More importantly, the ASR outputs are not always consumed by native speakers and it could be redirected to another automated system such as voice-enabled translation and speech-to-speech applications. For example, a simple experiment using Google translate showed that Google ASR recognized the sentence (Sari's Books) (كُتُبُ سَارِي) as (ساري كتب) without diacritics, which by a native speaker can have at least three meanings and cannot be discerned by context: (كَتَبَ سَارِي) Sari wrote, (كُتُبُ سَارِي) Sari's books, and (كُتِبَ سَارِي) Sari was written. However, the same system reads the recognized words, using TTS, as Sari was written (كُتِبَ سَارِيْ). Furthermore, it translated the recognized words to English as (Sari Wrote). Thus, a spoken sentence can be recognized by ASR to confer one meaning, spoken by TTS with a second meaning, and translated using machine translation to have a third meaning.

Research were oblivious to the risks of omitting diacritics in similar applications. We believe this is because they treated an Arabic ASR system as a standalone system. This limits its usability for several applications such as speech-enabled translation, and speech-to-speech applications. For example, one of the promising future applications in ASR is speech-to-speech that was perceived to be science fiction for a long time. This application converts spoken words from a language to spoken words of another language (e.g., Arabic to English). For speech-to-speech application, there are three possible scenarios based on the ASR output. Figure 4 shows these scenarios, next we discuss each of them.

First, when the Arabic ASR produces diacritized results with error rate (E_{d}). The results will pass through machine translation phase that adds translation error ($E_{\mathrm{d\_mt}}$). The translated results will then be sent to a TTS engine, which will add its error ($E_{\mathrm{d\_tts}}$). The final error for this type will be as in Eq. 3.

$$F_d = E_{\mathrm{d}} + E_{\mathrm{d\_mt}} + E_{\mathrm{d\_tts}} \tag{3}$$

Second, if the Arabic ASR system results are nondiacritized, then we can either use it as is or add diacritics automatically (assuming this helps the translator and/or TTS). If the system used the output without automatic diacritization, then the basic error of the ASR (E_{wd}) is propagated to the machine translation error ($E_{\mathrm{wd\_tts}}$). Afterward, the translated results will be passed to the TTS engine with error of ($E_{\mathrm{wd\_mt}}$). The final error for this type will be as in Eq. 4.

$$F_{\mathrm{wd}} = E_{\mathrm{wd}} + E_{\mathrm{wd\_mt}} + E_{\mathrm{wd\_tts}} \tag{4}$$

Third, when the nondiacritized ASR output is diacritized using automatic diacritizer such as KACST Automatic Diacritized [60] or Mishkal [61]. This will introduce a new error ($E_{\mathrm{wd\_a}}$). The translation error for this type will be ($E_{\mathrm{wd\_a\_mt}}$). The TTS error will be ($E_{\mathrm{wd\_a\_tts}}$). The final error for this type

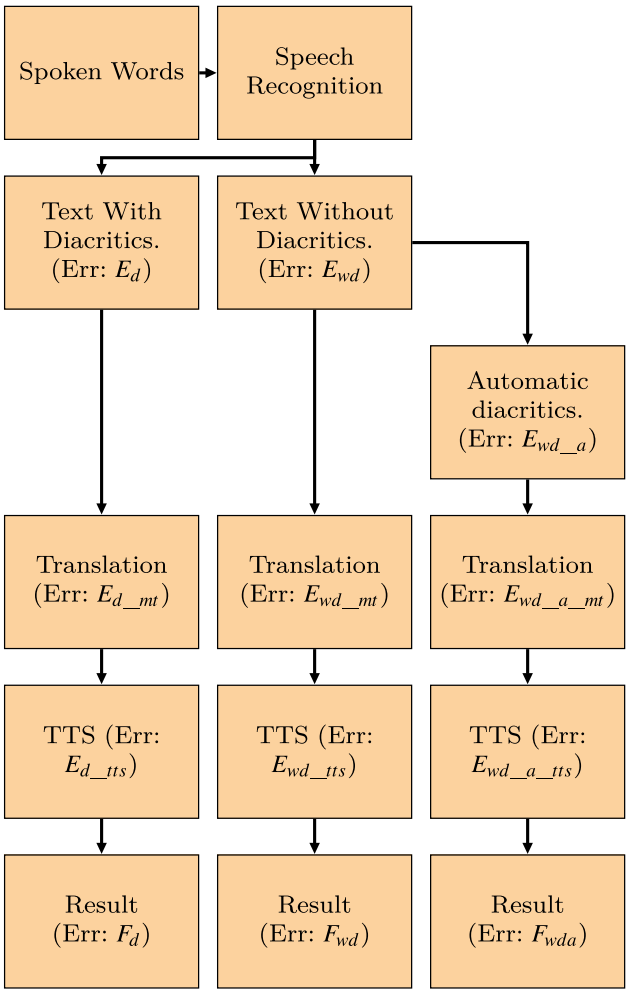

Fig. 4 Speech-to-Speech Scenarios using Diacritized and Nondiacritized ASR

will be as in Eq. 5.

$$F_{\text{wda}} = E_{\text{wd}} + E_{\text{wd\_a}} + E_{\text{wd\_a\_mt}} + E_{\text{wd\_a\_tts}} \tag{5}$$

To the best of our knowledge, no one investigated the three aforementioned options to find the suitable case for speech-to-speech applications for Arabic language. We believe that regarding the ASR part, including diacritics will increase the WER slightly ($E_{\text{d}} > E_{\text{wd}}$). This is showed in the literature by several studies and confirmed by our experimental results.

We believe that the translation error can be formulated as in Eq. 6. We used the product instead of summation because the error will propagate from the ASR error and amplify on translation where it reduces the optimal accuracy by $(100 - ASR_{\text{WER}})$. For example, with 10% WER for the ASR system, the maximum accuracy for the translator would be 90% relative to the spoken words subtracting a factor that is added to the translator WER (δ as in Eq. 7). Hence, Eqs. 3, 4, and 5 would not provide accurate representation for the system's overall error. The error $E_{\text{mt\_basic}}$ refers to the translator error assuming it receives 100% correct input, which is inherited from its training. However, this should be also categorized based on the input to be $E_{\text{mt\_basic\_diac}}$ for diacritized input and $E_{\text{mt\_basic\_Nondiac}}$ for nondiacritized input. For the TTS engine, the error will be propagated in a similar way.

$$E_{\text{mt}} = \delta \cdot E_{\text{mt\_(diac|nonDiac)}} \tag{6}$$

$$\delta = (1 + ASR_{\text{WER\_(diac|nonDiac)}}) \tag{7}$$

$$F_{\text{d\_diac}} = E_{\text{d}} + (\delta \cdot E_{\text{mt\_diac}}) + (\beta \cdot E_{\text{tts\_diac}}) \tag{8}$$

$$\beta = (1 + E_{\text{mt\_(diac|nonDiac)}}) \tag{9}$$

In this study, we study the effect of diacritics on Arabic ASR. This will help to better understand whether the effect is large or small. The effect can be better understood when integrated with other systems such as translation. However, other parts such as translation, automatic diacritization, and TTS are considered for future work. We expect that the best combination will be using diacritized ASR system when integrated with other parts such as translation or TTS. This is against the current prevalent convention in the literature for diacritics to be omitted. We believe this is because other studies did not integrate the ASR system with other systems, in which diacritics can affect the overall system's error rates.

5 Methodology

Figure 5 shows our experiments work flow. First, we obtained Arabic news transcription corpus (Audio + diacritized text) from the authors of [15,22]. The corpus contains 6 h of speech for 4754 sentences. The used diacritics are Sukoon, Dhamma, Fatha, Kasra, Shaddah, and Tanween (Kasr, Fath, and Dham). Second, we use the transcription to generate our own audio corpus of 17 h for the same 4754 sentences. Thus, the total length of our corpus is 23 h of 4754 sentences with 193 speakers. Third, we created subset corpora of 1, 2, 5, 10, 23 h. Each of which has two version of transcriptions: diacritized and nondiacritized. This creates a total of ten corpora.

Fourth, we created a diacritized and nondiacritized versions of the lexicon and LM for each corpus. Finally, each of the ten sets is trained to create eight different models that are: GMM-SI, GMM SAT (fMMLR), GMM MPE, GMM MMI, SGMM, SGMM-bMMI, DNN, DNN-MPE. The first six models were trained using CPU, and the last two were trained using GPU. We used KALDI toolkit with similar a setup to the one used by [20]. It uses classic 13-dimensional MFCC features that are spliced to include ±four neighboring frames and followed by dimensional reduction to 40 using LDA [62]. 40 is believed to be the optimal input dimension for GMM systems [63]. However, the authors indicated that using larger dimensions is helpful for DNN, which can be

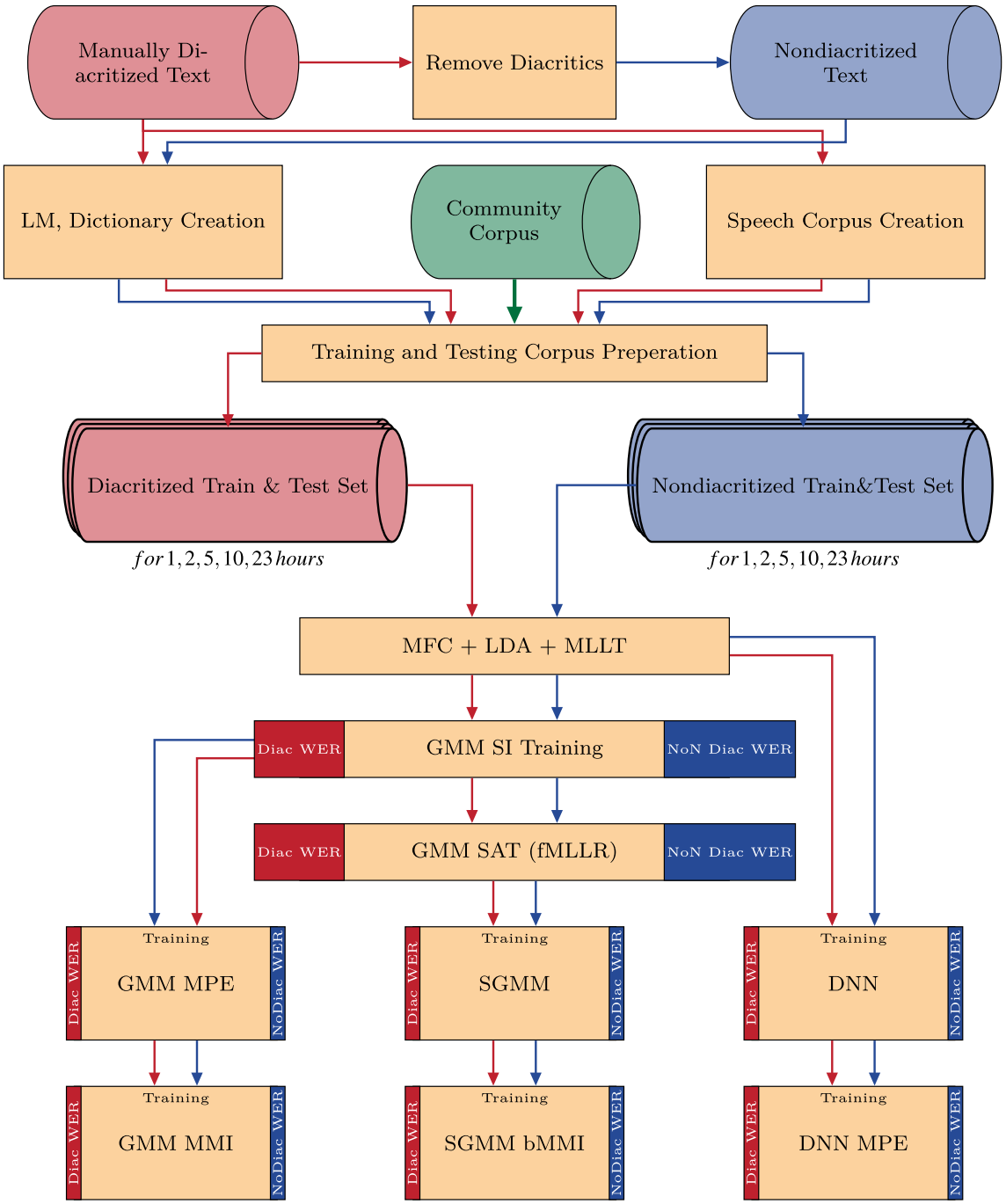

Fig. 5 Experiments Flow

studied in further research. We used the same dimension for all models. For DNN, we used five hidden layers each of which layer has 2048 nodes, and learning rate of 0.008 similar to [20]. The final features are then de-correlated using MLLT [64] (also known as STC [65]). Speaker adaptation is also applied using feature—space maximum likelihood linear regression (fMLLR) [66] (also known as constrained MLLR [67]). For the GMM models, they contain 512k Gaussians with 8k states and the SGMM has 5k states similar to [20].

Ali et al. [20] provided their source code, which is adopted from state-of-the-art methods to build ASR systems. We used those similar model building techniques and modified some parts to work with our corpus and language modeling. This is important in order to build a baseline for comparison. However, there are major differences between our

Table 5 Test corpus specification

Corpora (h)	Training	Testing	Total duration
1	52 m:37 s	7 m:33 s	1 h:0 m:11 s
2	1 h:45 m:3 s	15 m:4 s	2 h:0 m:8 s
5	4 h:22 m:34 s	37 m:37 s	5 h:0 m:12 s
10	8 h:45 m:9 s	1 h:15 m:4 s	10 h:0 m:13 s
23	20 h:40 m:20 s	1 h:52 m:38 s	22 h:32 m:58 s

Table 6 Machine specifications

CPU	Intel i7-7800X
RAM	32 GB
GPU	GTX 1080 TI, 3584 CUDA cores (for DNN training)
OS	Ubuntu 17.10
Disk	SSD
CUDA Version	9.1.85
GCC Version	6.4 (downgraded from 7.x for KALDI and CUDA to work)
Linux Kernel	4.13.0-17

implementations. First, for language modeling, the authors used MADA toolkit, while we used SRILM because it is widely used in the literature and MADA is an in-house tool developed by authors with limited exposure and usage by researchers in the field. Second, the authors used a non-diacritized corpus, while we tested both diacritized and nondiacritized corpora. Another important difference about the corpus is that they used a closed-source and paid corpus, while we used a publicly available corpus from [15,22].

6 Experimental Results

In this section, we discuss our experiments setup (Sect. 6.1) and results (Sect. 6.2).

6.1 Experiments Setup

We used 12.5% of each corpus for testing and the rest for training as shown in Table 5. The variability between different corpus sizes was controlled through making the validation dataset incremental. For example, a small corpus validation set is a subset of each larger corpus. We prepared the training environment where we faced numerous challenges such as running the graphics card CUDA library on Ubuntu, which was not supported because the graphics card was new. However, we managed to prepare a working environment using a local machine with the specifications shown in Table 6.

Finally, we categorized and analyzed the results based on WERs as will be discussed in the next section. The results for each model is compared between the diacritized and nondiacritized version. This should clarify the effect of diacritics on Arabic ASR.

6.2 Results and Discussion

Table 7 shows the WERs for our 80 models. The WERs ranged from 42 to 4.68%. The lowest WER was using DNN-MPE with the largest corpus, and the highest WER was for the smallest corpus using SGMM-bMMI. DNN-MPE provided the best WER for most of our models with 40–50% reduction of traditional GMM models. It shows that the WER is always decreasing as the corpus size increases. This is valid for both diacritized and nondiacritized corpus. The DNN and DNN+MPE outperformed the other techniques. Table 8 shows the accuracy for our 80 models. It is calculated based on Eq. 10 [15]. Accuracy ranged from 69.89% (for the 1 h diacritized model) to 95.32% (for the 23 h nondiacritized model). It shows that the accuracy is always increasing as the corpus size increases. This is valid for both diacritized and nondiacritized corpus.

$$\text{Accuracy} = 100 - \text{WER} \tag{10}$$

We conducted another two experiments, in which we produced another sixteen models to compare our results with similar datasets we received from the authors of [15]. Comparison is shown in Table 9. We included only four of the produced models in the comparison, two of which are similar to the used by the authors and the other two are our best results (because all were better than the base GMM and worst than our DNN-MPE). For the same model used by the authors, we achieved slightly better results ($< 1\%$). With our DNN-MPE model, we achieved 53% enhancement for the nondiacritized model and 48% for the diacritized model.

Using diacritics increased WER in all cases (except for few cases due to using a small dataset). WER increased by adding diacritics for the 8 models by 3.29% for the 1 h corpus, 0.59% for the 2 h corpus, 1.07% for the 5 h corpus, 1.48% for the 10 h corpus, and 0.92 for the 23 h corpus. Although this error is supposed to decrease as the corpus size increases, we noticed some fluctuation from 2 to 23 h. This might be due to increased ambiguity in the LM, which sometimes resolved by introducing sentences with similar words and sometimes increased by using new words. This fluctuation is yet not decisive and needs to be further investigated. Although the diacritics increase WER by about 1%, it avoid any ambiguity in the result. Authors who studied diacritized Arabic ASR also achieved similar results such as Alghamdi et al. [15], which in their study the diacritized model increased WER by 1.5%. In Abushariah et al. [16,17] study, the diacritized

Table 7 Word error rates (WERs) for the 80 trained model

Model	1 h		2 h		5 h		10 h		23 h	
	Diac (%)	non-Diac (%)	Diac (%)	non-Diac (%)	Diac (%)	non-Diac (%)	Diac (%)	non-Diac (%)	Diac (%)	non-Diac (%)
GMM	30.11	29.13	21.25	24.08	19.02	18.96	15.02	13.94	10.81	10.25
GMM+fMLLR	26.02	22.59	17.28	18.65	16.49	16.27	14.61	13.69	9.10	7.91
GMM+MPE	28.48	23.40	21.02	21.25	17.61	15.21	13.16	11.62	9.62	8.52
GMM+bMMI (0.05)	27.33	23.57	20.26	19.72	18.60	17.61	13.42	11.26	9.67	8.20
SGMM+fMMLR	24.55	23.90	15.90	11.47	9.93	9.32	8.49	7.15	6.48	6.36
SGMM+bMMI (0.1)	42.72	34.82	17.43	15.14	14.92	13.05	12.01	9.53	7.26	6.03
DNN	15.06	12.77	12.16	11.09	11.37	10.02	9.20	7.88	5.78	4.89
DNN+MPE	14.57	12.27	11.39	10.55	10.47	9.35	8.76	7.71	5.53	4.68
Average Increase in WER by adding diacritics △										
△	3.29		0.59		1.07		1.48		0.92	

Table 8 Accuracy for the 80 trained model

Model	1 h		2 h		5 h		10 h		23 h	
	Diac (%)	non-Diac (%)	Diac (%)	non-Diac (%)	Diac (%)	non-Diac (%)	Diac (%)	non-Diac (%)	Diac (%)	non-Diac (%)
GMM	69.89	70.87	78.75	75.92	80.98	81.04	84.98	86.06	89.19	89.75
GMM+fMLLR	73.98	77.41	82.72	81.35	83.51	83.73	85.39	86.31	90.9	92.09
GMM+MPE	71.52	76.6	78.98	78.75	82.39	84.79	86.84	88.38	90.38	91.48
GMM+bMMI (0.05)	72.67	76.43	79.74	80.28	81.4	82.39	86.58	88.74	90.33	91.8
SGMM+fMMLR	75.45	76.1	84.1	88.53	90.07	90.68	91.51	92.85	93.52	93.64
SGMM+bMMI (0.1)	57.28	65.18	82.57	84.86	85.08	86.95	87.99	90.47	92.74	93.97
DNN	84.94	87.23	87.84	88.91	88.63	89.98	90.8	92.12	94.22	95.11
DNN+MPE	85.43	87.73	88.61	89.45	89.53	90.65	91.24	92.29	94.47	95.32

Table 9 Results comparison with Alghamdi et al. [15]

	Model type	NonDiac WER (%)	Diac WER (%)
[15]	GMM	8.61	10.14
Our results	GMM	8.42	9.92
	DNN-MPE	4.31	4.85

model increased WER by 1.15%. Hence, we recommend the following.

1. Using diacritics for every application that require the Arabic ASR system to be integrated with other systems, in which the other system produces better results using diacritics. For example, speech-to-speech.
2. Avoid diacritics in cases where the diacritized system produced much larger WER than the nondiacritized system and there is no integration with other systems that depends on the ASR output.

Those recommendations need to be further evaluated in a production setup that is integrated with other modules such as translation and TTS to understand the actual effect of the system. Hence, for future work direction we recommend to investigate each part of the speech-to-speech process that have not yet been studied. The studies should take into consideration integrating these modules together to understand how the error from one system can propagate or further amplify the error of the other modules.

7 Conclusion

In this paper, we studied the effect of diacritization on Arabic ASR. We built 80 models using GMM-SI, GMM SAT (fMMLR), GMM MPE, GMM MMI, SGMM, SGMM-bMMI, DNN, and DNN-MPE. We built those models using KALDI toolkit, and SRILM was used for language modeling. We used diacritized and nondiacritized versions and checked how diacritics affected WERs. Our results show that diacritics increased WER for all of our models (except few models

with small corpus). The average increased WERs were 3.29% for the 1 h corpus, 0.59% for the 2 h corpus, 1.07% for the 5 h corpus, 1.48% for the 10 h corpus, and 0.92% for the 23 h corpus. In addition, we analyzed some cases where this increase in the WER might reduce the system's overall error. For example, speech-to-speech recognition integrates Arabic ASR, machine translation, and TTS. We believe that it might be more beneficial to use diacritized ASR system in similar cases.

For the future work, we recommend further investigation of the diacritization effect on Arabic voice-enabled translation and speech-to-speech systems. In addition, we used manually diacritized corpus in our models. This will show the minimal effect of diacritization on Arabic ASR systems. However, in a production setup it is probable that the ASR system will use automatic diacritization process. Studying the effect of this automation is considered for future work.

Acknowledgements This work was supported and funded by Kuwait University, Research Project No. (QE 06/14).

References

1. Davis, S.; Mermelstein, P.: Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE Trans. Acoust. Speech Signal Process. **28**(4), 357–366 (1980)
2. Hermansky, H.: Perceptual linear predictive (PLP) analysis of speech. J. Acoust. Soc. Am. **87**(4), 1738–1752 (1990)
3. Rabiner, L.; Juang, B.: An introduction to hidden markov models. IEEE ASSP Mag. **3**(1), 4–16 (1986)
4. Dong, Y.; Deng, L.: Automatic Speech Recognition. Springer, Berlin (2012)
5. Dahl, G.E.; Dong, Y.; Deng, L.; Acero, A.: Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. IEEE Trans. Audio Speech Lang. Process. **20**(1), 30–42 (2012)
6. Hinton, G.; Deng, L.; Dong, Y.; Dahl, G.E.; Mohamed, A.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T.N.; et al.: Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process. Mag. **29**(6), 82–97 (2012)
7. Seide, F.; Li, G.; Yu, D.: Conversational speech transcription using context-dependent deep neural networks. In: Interspeech, pp. 437–440 (2011)
8. University of Cambridge. The hidden markov model toolkit (HTK) (2015). http://htk.eng.cam.ac.uk/
9. CMU. Cmusphinx speech recognition toolkit. http://cmusphinx.sourceforge.net/
10. KALDI. Kaldi speech recognition toolkit (2015). http://kaldi.sourceforge.net/
11. Stolcke, A.; et al.: Srilm-an extensible language modeling toolkit. In: Interspeech, vol. 2002, p. 2002 (2002)
12. United Nations. http://www.un.org/en/sections/about-un/official-languages/ (2016). Accessed 17 July 2016
13. Gordon Jr, Raymond G.: Ethnologue: Languages of the world, Dallas, Texas.: SIL international (2005). Online version: http://www.ethnologue.com
14. Kirchhoff, K.; Bilmes, J.; Henderson, J.; Schwartz, R.; Noamany, M.; Schone, P.; Ji, G.; Das, S.; Egan, M.; He, F. et al.: Novel speech recognition models for Arabic. In: Johns-Hopkins University Summer Research Workshop (2002)
15. Alghamdi, M.; Elshafei, M.; Al-Muhtaseb, H.: Arabic broadcast news transcription system. Int. J. Speech Technol. **10**(4), 183–195 (2007)
16. Abushariah, M.A.M.; Ainon, R.N.; Zainuddin, R.; Elshafei, M.; Khalifa, O.O.: Natural speaker-independent arabic speech recognition system based on hidden Markov models using sphinx tools. In: International Conference on Computer and Communication Engineering (ICCCE), pp. 1–6 (2010)
17. Abushariah, M.A.A.M.; Ainon, R.; Zainuddin, R.; Elshafei, M.; Khalifa, O.O.: Arabic speaker-independent continuous automatic speech recognition based on a phonetically rich and balanced speech corpus. Int. Arab J. Inf. Technol. (IAJIT) **9**(1), 84–93 (2012)
18. Abushariah, M.A.M.; Ainon, R.N.; Zainuddin, R.; Elshafei, M.; Khalifa, O.O.: Phonetically rich and balanced speech corpus for arabic speaker-independent continuous automatic speech recognition systems. In: 10th International Conference on Information Sciences Signal Processing and their Applications (ISSPA), pp. 65–68 (2010)
19. Hyassat, H.; Zitar, R.A.: Arabic speech recognition using sphinx engine. Int. J. Speech Technol. **9**(3–4), 133–150 (2006)
20. Ali, A.; Zhang, Y.; Cardinal, P.; Dahak, N.; Vogel, S.; Glass, J.: A complete kaldi recipe for building Arabic speech recognition systems. In: Spoken Language Technology Workshop (SLT), IEEE, pp. 525–529 (2014)
21. Ali, A.; Vogel, S.; Renals, S.: Speech recognition challenge in the wild: Arabic MGB-3. In: IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pp. 316–322 (2017)
22. AlShafei, M.: KFUPM Arabic speech processing group (2016). http://www.ccse.kfupm.edu.sa/~elshafei/AASR.htm. Accessed 17 Dec 2017
23. Woodland, P.C.: Speaker adaptation for continuous density hmms: A review. In: ISCA Tutorial and Research Workshop (ITRW) on Adaptation Methods for Speech Recognition, pp. 11–19 (2001)
24. Anastasakos, T.; McDonough, J.; Schwartz, R.; Makhoul; J.: A compact model for speaker-adaptive training. In: Proceedings of Fourth International Conference on Spoken Language, ICSLP 96. vol. 2, pp. 1137–1140 (1996)
25. Pye, D.; Woodland, P.C.: Experiments in speaker normalisation and adaptation for large vocabulary speech recognition. In: IEEE International Conference on Acoustics, Speech, and Signal Processing, 1997. ICASSP-97, vol. 2, pp. 1047–1050 (1997)
26. Gauvain, J.-L.; Lee, C.-H.: Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains. IEEE Trans. Speech Audio Process. **2**(2), 291–298 (1994)
27. Leggetter, C.J.; Woodland, P.C.: Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. Comput. Speech Lang. **9**(2), 171–185 (1995)
28. Shinoda, K.: Speaker adaptation techniques for automatic speech recognition. In: Proceeding of APSIPA ASC (2011)
29. Titus Felix Furtună: Dynamic programming algorithms in speech recognition. Revista Informatica Economică nr **2**(46), 94–99 (2008)
30. Chakraborty, C.; Talukdar, P.H.: Issues and limitations of hmm in speech processing: a survey. Int. J. Comput. Appl. **141**(7), 13–17 (2016)
31. Melnikoff, S.J.; Quigley, S.F.; Russell, M.J.: Implementing a hidden Markov model speech recognition system in programmable logic. In: International Conference on Field Programmable Logic and Applications, pp. 81–90. Springer, Berlin (2001)
32. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. Proc. IEEE **77**(2), 257–286 (1989)

33. Holmes, W.: Speech Synthesis and Recognition. CRC Press, Boca Raton (2001)
34. Swee, L.H.: Implementing speech-recognition algorithms on the TMS320C2xx platform (1988)
35. Buchsbaum, A.L.; Giancarlo, R.: Algorithmic aspects in speech recognition: an introduction. J. Exp. Algorithm. (JEA) **2**, 1 (1997)
36. Satori, H.: Arabic speech recognition system based on CMUSphinx, pp. 28–35 (2007)
37. Juang, B.H.; Rabiner, L.R.: Hidden markov models for speech recognition. Technometrics **33**(3), 251–272 (1991)
38. Hinton, G.; Deng, L.; Yu, D.; Dahl, G.E.; Mohamed, A.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T.N.; Kingsbury, B.: Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process. Mag. **29**(6), 82–97 (2012). https://doi.org/10.1109/MSP.2012.2205597. ISSN 1053-5888
39. Saeed, K.; Nammous, M.: Heuristic method of Arabic speech recognition. In: Proceedings of IEEE 7th International Conference on DSPA, pp. 528–530 (2005)
40. Deng, L.; Dong, Y.: Deep learning: methods and applications. Found. Trends Signal Process. **7**(3–4), 197–387 (2014)
41. Yu, D.; Deng, L.; Dahl, G.: Roles of pre-training and fine-tuning in context-dependent DBN-HMMS for real-world speech recognition. In: Proceedings of NIPS Workshop on Deep Learning and Unsupervised Feature Learning (2010)
42. Deng, L.; Li, J.; Huang, J.-T.; Yao, K.; Yu, D.; Seide, F.; Seltzer, M.; Zweig, G.; He, X.; Williams, J. et al.: Recent advances in deep learning for speech research at microsoft. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 8604–8608 (2013)
43. Nielsen, J.: Usability Engineering, 1st edn. Morgan Kaufmann, Burlington (1994)
44. Rashid, R.: Microsoft research shows a promising new breakthrough in speech translation technology, (2012). http://blogs.microsoft.com/next/2012/11/08/microsoft-research-shows-a-promising-new-breakthrough-in-speech-translation-technology/
45. Povey, D.; Ghoshal, A.; Boulianne, G.; Burget, L.; Glembek, O.; Goel, N.; Hannemann, M.; Motlicek, P.; Qian, Y.; Schwarz, P. et al.: The kaldi speech recognition toolkit. In: IEEE 2011 Workshop on Automatic Speech Recognition and Understanding, number EPFL-CONF-192584. IEEE Signal Processing Society (2011)
46. Allauzen, C.; Riley, M.; Schalkwyk, J.; Skut, W.; Mohri, M.: Openfst: a general and efficient weighted finite-state transducer library. In: International Conference on Implementation and Application of Automata, pp. 11–23. Springer, Berlin (2007)
47. Gales, M.; Young, S.: The application of hidden Markov models in speech recognition. Found. Trends Signal Process. **1**(3), 195–304 (2008)
48. Gales, M.J.F.: Discriminative models for speech recognition. In: 2007 Information Theory and Applications Workshop, pp. 170–176 (2007)
49. Tremain, T.E.: The government standard linear predictive coding algorithm: LPC-10. Speech Technol. **1**(2), 40–49 (1982)
50. Biadsy, F.; Moreno, P.J.; Jansche, M.: Google's cross-dialect arabic voice search. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4441–4444 (2012)
51. Vergyri, D.; Kirchhoff, K.: Automatic diacritization of Arabic for acoustic modeling in speech recognition. In: Proceedings of the Workshop On Computational Approaches to Arabic Script-based Languages, pp. 66–73. Association for Computational Linguistics (2004)
52. Vergyri, D.; Kirchhoff, K.; Duh, K.; Stolcke, A.: Morphology-based language modeling for Arabic speech recognition. INTERSPEECH **4**, 2245–2248 (2004)
53. Soltau, H.; Saon, G.; Kingsbury, B.; Kuo, J.; Mangu, L.; Povey, D.; Zweig, G.: The IBM 2006 gale Arabic ASR system. In: IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07, vol. 4, pp. IV–349–IV–352 (2007)
54. Djellab, M.; Amrouche, A.; Bouridane, A.; Mehallegue, N.: Algerian modern colloquial arabic speech corpus (AMCASC): regional accents recognition within complex socio-linguistic environments. Lang. Resour. Eval. **51**(3), 613–641 (2017)
55. Mohammed, Z.Y.; Khidhir, A.S.M.: Real-time Arabic speech recognition. Int. J. Comput. Appl. **81**(4), 43–45 (2013)
56. Alkhatib, B.; Kawas, M.; Alnahhas, A.; Bondok, R.; Kannous, R.: Building an assistant mobile application for teaching Arabic pronunciation using a new approach for Arabic speech recognition. J. Theor. Appl. Inf. Technol. **95**(3), 478 (2017)
57. Gorin, A.L.; Riccardi, G.; Wright, J.H.: How may I help you? Speech Commun. **23**(1), 113–127 (1997)
58. Price, M.; Glass, J.; Chandrakasan, A.P.: A 6 mw, 5000-word real-time speech recognizer using wfst models. IEEE J. Solid-State Circuits **50**(1), 102–112 (2015)
59. Elmisery, F.A.; Khalil, A.H.; Salama, A.E.; Hammed, H.F.: A FPGA-based hmm for a discrete Arabic speech recognition system. In: Proceedings of the 15th International Conference on Microelectronics, 2003. ICM 2003, pp. 322–325 (2003)
60. Alghamdi, M.; Muzaffar, Z.; Alhakami, H.: Automatic restoration of arabic diacritics: a simple, purely statistical approach. Arab. J. Sci. Eng. **35**(2), 125 (2010)
61. http://tahadz.com/mishkal. Accessed 17 July 2017
62. Duda, R.O.; Hart, P.E.; Stork, D.G.: Pattern Classification, 2nd edn. Wiley, New York (2001)
63. Rath, S.P.; Povey, D.; Veselỳ, K.; Cernockỳ, J.: Improved feature processing for deep neural networks. In: INTERSPEECH, pp. 109–113 (2013)
64. Gopinath, R.A.: Maximum likelihood modeling with Gaussian distributions for classification. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 2, pp. 661–664 (1998)
65. Gales, M.J.F.: Semi-tied covariance matrices for hidden Markov models. IEEE Trans. Speech Audio Process. **7**(3), 272–281 (1999)
66. Povey, D.; Saon, G.: Feature and model space speaker adaptation with full covariance Gaussians. In: INTERSPEECH, pp. 1145–1148 (2006)
67. Gales, M.J.F.: Maximum likelihood linear transformations for HMM-based speech recognition. Comput. Speech Lang. **12**(2), 75–98 (1998)