Natural Language Engineering **21** (3): 477–495. © Cambridge University Press 2013
doi:10.1017/S1351324913000284

A survey of automatic Arabic diacritization techniques

AQIL M. AZMI and REHAM S. ALMAJED
Department of Computer Science, King Saud University, Riyadh 11543, Saudi Arabia
e-mails: aqil@ksu.edu.sa, reham.imamu@gmail.com

(*Received 16 October 2012; revised 8 September 2013; accepted 10 September 2013; first published online 10 October 2013*)

Abstract

In Modern Standard Arabic texts are typically written without diacritical markings. The diacritics are important to clarify the sense and meaning of words. Lack of these markings may lead to ambiguity even for the natives. Often the natives successfully disambiguate the meaning through the context; however, many Arabic applications, such as machine translation, text-to-speech, and information retrieval, are vulnerable due to lack of diacritics. The process of automatically restoring diacritical marks is called diacritization or diacritic restoration. In this paper we discuss the properties of the Arabic language and the issues that are related to the lack of the diacritical marking. It will be followed by a survey of the recent algorithms that were developed to solve the diacritization problem. We also look into the future trend for researchers working in this area.

1 Introduction

As a language, Arabic is both challenging and interesting. It is an old language that predates Islam. This is confirmed by the discovery of pre-Islamic Arabic inscriptions from second to fourth centuries CE (Al-Azami 2011). It is remarkable because the language has retained its orthography (spelling conventions) for the last fifteen centuries. A person with a slight knowledge of the language can easily read 1,400-year-old parchments. Of a pre-Islamic Arabic inscription at Jabal Ramm in Jordan that was discovered in 1931 and believed to be from the fourth century CE, Bellamy (1988) maintains that the language is closer to modern Arabic than Shakespeare's language to modern English. Arabic is a native language of over 330 million speakers (Central Intelligence Agency 2008), and over a billion and a half Muslims that regularly read the Holy Qur'an and perform the five daily prayers.

Arabic is a semitic language and can be classified as classical and modern. The Classical Arabic represents the pure language spoken by the Arabs, whereas Modern Standard Arabic (MSA) is an evolving variety of Arabic with constant borrowing and innovations to meet modern challenges (Farghaly and Shaalan 2009). In the Arabic orthographic system, small diacritical marks are used to represent three short vowels (a, i, u), while letters ا، و، ي، ى are used to indicate vocalic length. The diacritical marks are placed either above or below the letters to indicate the phonetic information associated with each letter to clarify the sense and meaning of the word.

https://doi.org/10.1017/S1351324913000284 Published online by Cambridge University Press

Table 1. *The basic Arabic diacritics. Nunation* (tanween) *can only be placed at the end of the word. The gemination* (shaddah) *is used to duplicate the letter it is placed on phonetically*

Diacritic set	Diacritic on the letter ن	Transliterated name	Arabic name	Place	Pronunciation
Short vowels	نَ	fatha	فتحة	above	/n//a/
	نُ	damma	ضمة	above	/n//u/
	نِ	kasra	كسرة	below	/n//i/
Nunation	نً	tanween fath	تنوين الفتح	above	/n//an/
	نٌ	tanween damm	تنوين الضم	above	/n//un/
	نٍ	tanween kasr	تنوين الكسر	below	/n//in/
Syllabification marks	نّ	shaddah	شدة	above	/n//n/
	نْ	sukon	سكون	above	/n/

There is an obvious discrepancy between the spoken and written forms of an Arabic word, as several different words may have the same orthography (Rashwan *et al.* 2011). For example, the undiacritized word (وهم) may mean (وَهَمَّ) *and he initiated* (وَهُمْ) *and they*, or (وَهْمٌ) *illusion*. The Arabic sound system consists of a total of thirteen different diacritics. The eight basic diacritics can be grouped into three sets as shown in Table 1. The rest are a combination of the diacritical mark *shaddah* and either one of the diacritics from the other two sets. In real-word situation, MSA is typically written without diacritics in magazines, books, etc mainly to cut the typing cost. A typical Arabic typist's speed is reduced to about one-third if he/she is entering diacritical marks. Diacritics are important not only for children or someone learning Arabic as a second language but are also crucial for a number of Natural Language Processing (NLP) applications such as automatic speech recognition (ASR), text to speech (TTS), machine translation (MT), etc that may need diacritized texts as a source for learning (Ananthakrishnan, Narayanan and Bangalore 2005). The same can be said about information retrieval. The majority of Arabic documents on the World Wide Web are written without diacritical markings. Consider the undiacritized word (علم), there are several meanings associated with it and among the more common ones are (عِلْم) *science*, and (عَلَم) *flag*. As a test, we googled for the undiacritized form (علم) and the number of hits was 98,500,000; while for (عِلْم) and (عَلَم) it was 463,000 and 335,000 respectively. If we take this as a rude measure, it goes to show that roughly 99% of the Arabic content in the web is without diacritical markings. Now we can imagine the hard time the user has to undergo reviewing each document pulled by a search engine just to confirm whether it has the right form of the word he/she is looking for. There are also instances when a sentence is truly ambiguous that can only be clarified through a diacritical marking. See Figure 1 for one such example. Diacritization is one of the Arabic NLP applications in which

https://doi.org/10.1017/S1351324913000284 Published online by Cambridge University Press

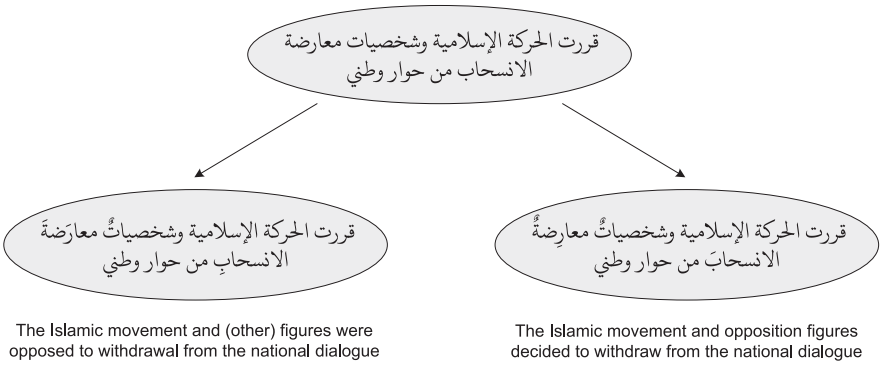


Fig. 1. A truly ambiguous Arabic sentence that recently appeared in aljazeera.net. Logically both interpretations make sense, but they both cannot be true. Only the diacritics can resolve the ambiguity. Here and in most cases a partial diacritization is sufficient to fully resolve the ambiguity (the lower two boxes). This is a practical solution for materials intended for printed form but not for text to speech, where full diacritization is necessary.

researchers develop tools that diacritize the texts automatically. In this paper we will survey some of the recent methods developed to tackle Arabic diacritization.

The rest of the paper is organized as follows. In Section 2 we look into the characteristics of the Arabic language, specifically those related to diacritization. A look at main techniques and means to evaluate the quality of diacritization is covered in Section 3. In Section 4 we do an in-depth survey of recent schemes to tackle the problem. We critically look at different techniques in Section 5. Finally, in Section 6 we conclude and summarize our survey.

2 Characteristics of the Arabic language

Arabic is a phonetic language with twenty-eight basic letters in its alphabet.[1] There are a number of challenges in the Arabic language and one of these is its high degree of ambiguity, morphologically and syntactically. In this section we briefly go over the morphology and syntax of the Arabic language and how it compares with that in other languages such as English.

2.1 Arabic morphology

Morphology deals with the internal structure of words and it is considered a base layer for other linguistic layers (Attia 2000). Arabic morphology is systematic but rather complex. There are two main properties that are used to build words: derivation and agglutination.

[1] Actually, there is some disagreement about the number of letters in Arabic. Most of the sources state twenty-eight basic letters. Others consider hamza (ء) a basic letter, making it twenty-nine letters (Habash 2010). These countings are based on the sound of the letter. But if we look at their basic shapes, we will spot thirty-six different shapes. In Arabic, both ت and ة have the same sound but are different letters, and replacing one with another changes the meaning of the word, e.g. (قضيت) *get rid of*, and (قضية) *case*. The same is true for letters ا and ى, the words (عصا) *stick*, and (عصى) *disobey*. This phenomena is known in other languages as well. The Danes did not officially recognize 'W' as a separate letter until 1980; before that it was just considered a variation of the letter 'V' (Wikipedia n.d.).

https://doi.org/10.1017/S1351324913000284 Published online by Cambridge University Press

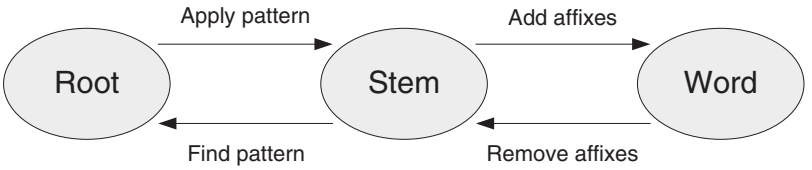


Fig. 2. Arabic word construction system.

The derivation property of Arabic is a stem generation mechanism in which stems are derived from the list of roots by applying some template patterns. The patterns are models that have specific semantics which are used to build the internal structure of words using three basic letters ف، ع، ل (in case of trilateral root) with some additional letters such as ا، س، ت، ي and different diacritics. The process of stem derivation yields a huge number of stems that gain their meaning from both the roots and the patterns (Attia 2000). For example, by using the root word ك ت ب (k t b) we can generate the stems: (كَتب) *write*, (كتاب) *book*, (كُتب) *books*, (كاتب) *writer*, (كتّاب) *writers*, (مكتوب) *written*, (مكاتبة) *correspondence*, (مكتبة) *library*, (مكتبات) *libraries* etc. Note that in the absence of a diacritical marking the first and the third words have the same form. Typically, the natives can decipher the word based on the context, but it is a challenge for someone new to the Arabic language. The famous Arabic lexicon, Lisan al-Arab (Ibn Mandhur 2009) lists 6,350 trilateral roots and 2,500 quadrilateral ones; and of these only about 1,200 roots are in MSA vocabulary (Habib 2008).

The agglutination property of the Arabic language glues stems that were generated using the derivation property with affixes (prefixes and suffixes). Figure 2 shows the Arabic word construction system using these two properties. The removal of prefixes in English often reverses or alters the meaning or grammatical function of the word, e.g. clean and unclean. This is not the case in Arabic, where removing the prefixes does not reverse the meaning of the word (Al-Sughaiyer and Al-Kharashi 2004).

Arabic has fewer affixes than English, which has seventy-five prefixes and about 250 suffixes (Salton 1989). However, the affixes in Arabic have the characteristic of concatenating with each other by predefined linguistic rules causing an increase in the overall number of affixes (Ali 1988). This characteristic allows one to articulate a complete sentence using a single string. For example, the word وسمعتهم (= و + سمع + ت + هم) is composed of a conjunction و, a verb سمع, a subject pronoun ت and an object pronoun هم. The English meaning of the word is, *and I heard them*.

Some examples of prefixes are as follows: the conjunctions و، ف, the prepositions ب، ل، ك، و، ت, the definite article ال, and some letters that are attached to the beginning of verbs to indicate tense, gender or number ن، ت، ي. Suffixes are fused onto the end of the stems such as possessive pronouns ي، هم، هن، ها، هـ. The suffixes ة، ت indicate the feminine of the word (either noun or verb). Then there are suffixes that are added to the word to indicate number such as ان، ين for dual masculine or feminine, and ون، ين for plural masculine and ات for plural feminine. Table 2 shows example of affixes added to the stem (معلم).

https://doi.org/10.1017/S1351324913000284 Published online by Cambridge University Press

Table 2. *Example of affixes added to the stem (معلم). The masculine and feminine are designated m and f respectively*

معلمة	teacher (*f*)	المعلم	the teacher	والمعلم	and the teacher
كالمعلم	as teacher	معلمان	two teachers (*m*)	معلمتان	two teachers (*f*)
معلمون	teachers (*m*)	معلمهم	their teacher	معلمه	his teacher (*m*)
معلمته	his teacher (*f*)	معلمها	her teacher (*m*)	معلمتها	her teacher (*f*)

2.2 Arabic syntax

A common view of the Arabic grammarian is that sentences in Arabic could either start with a noun (جملة إسمية) nominal sentence, or a verb (جملة فعلية) verbal sentence. A sentence starting with a noun may contain a verb within its structure but not in the first position. The diacritics are used with some nouns and verbs to indicate their case and mood respectively. The basic marks damma الضمة, fatha الفتحة, and kasra الكسرة are for nominative, accusative, and genitive nouns respectively, whereas damma, fatha, and sukon السكون are the basic marks for indicative, subjunctive, and jussive verbs respectively.

Morphology deals with the structure of words but it cannot determine how they could be integrated together. Syntax describes the construction of sentences and phrases (Habash 2010). It determines the syntactic role of words in the context of the sentence. However, morphology has a high impact on the correctness of syntax analysis since the former provides the latter with inflectional categories information such as number tense, person, voice, mood, gender, case, and definiteness (Ryding 2006; Attia 2008). So any ambiguity at the morphological level will have an effect on the syntactic level.

There are two main properties of the Arabic language which makes its syntax complex and ambiguous: it is a pro-drop and free order language. The pro-drop property means that the subject may not be explicitly present which leads to a high degree of complexity for automatic diacritization tools since the noun after the verb could be a nominative subject or an accusative object (Farghaly and Shaalan 2009). The basic order of Arabic words in a sentence is Verb–Subject-Object (VSO). However, other orders are possible: SVO and VOS (Farghaly and Shaalan 2009). This property of Arabic, the free-order property, is a crucial challenge for any automatic diacritization tool as the order of the sentence can only be clarified through the context. The diacritization is closely related to Arabic morphology and syntax in that the diacritics of internal letters are determined morphologically, while the case-ending diacritics are determined syntactically.

2.3 Ambiguity due to lack of diacritics

Ambiguity poses a significant challenge for any researcher working on Arabic NLP. The ambiguity exists on many levels. As a consequence of the lack of diacritics, Debili, Achour and Souissi (2002) calculated an average of 11.6 possible diacritizations for every non-diacritized word when analyzing an Arabic text with

https://doi.org/10.1017/S1351324913000284 Published online by Cambridge University Press

23,000 script forms. Maamouri and Bies (2010) reported twenty-one different analyses for the word (ثمن) as reported by Buckwalter Arabic Morphological Analyzer (BAMA) (Buckwalter 2004). According to SYSTRAN, a firm involved in developing MT systems for the last 40 years, there is an average of 2.3 ambiguities for a token in most languages, while for MSA it is 19.2 (Farghaly and Shaalan 2009).

While the undiacritized texts minimize the orthography representation of the spoken words, the accurate pronunciation of the written words demands a reader with good background knowledge. He/she needs to read and analyze the text morphologically and syntactically in order to pronounce it correctly. Consider the word read; it could be read as a past tense or a present tense depending on its context. However, in Arabic the ambiguity is more involved. Regrettably, MSA texts are usually written devoid of any diacritics. Words with similar spellings may have different pronunciations and meanings that can only be resolved by the context and proper knowledge of the grammar (Rashwan *et al.* 2011). The ambiguity caused by the absence of the diacritics fall into one of the following three cases:

- Ambiguity within the POS tag (Maamouri, Bies and Kulick 2006). This could be further subdivided into: (i) words with same spelling and POS tag but a different lexical sense. For example, the word (علم) with a noun POS may either mean (عِلْم) *science*, or (عَلَم) *flag*; and (ii) homograph ambiguity where words having the same spelling but different POS tags and lexical senses (Maamouri *et al.* 2006; Farghaly and Shaalan 2009). Again, consider the word (علم), it has different meaning with different POS tags. It could be (عَلِمَ) *he knew*, with a POS tag of third person masculine, singular, perfective verb; or it could be (عُلِمَ) *it was learned*, having a POS tag of third person, singular, passive verb.
- Ambiguity at grammatical level (syntactic ambiguity). Here sentences and phrases could be interpreted in more than one way. Here the diacritics are the only means to resolve the ambiguity (Maamouri *et al.* 2006). One such example is given in Figure 1.
- Internal word structure ambiguity (Van Gompel 2008; Farghaly and Shaalan 2009). The agglutination property of Arabic may produce a problem that can only be resolved through diacritics. For example, (كتاب) may either be (كِتَاب) *book*, or (كُتَّاب) *writers*.

3 Diacritization categories

Diacritization categories are the categories according to which letters should be diacritized. Attia (2000) classifies this into the following three categories: full, half, and partial. We would like to add minimalist diacritization to the category making it four. Going briefly over them are as follows:

- Full diacritization: All the letters are given appropriate diacritics.
- Half diacritization: Here we only diacritize the morphological-independent letters. The goal is to diacritize all the letters of a word except those which depend on the syntactic analysis of the word. Although often it is the last letter

https://doi.org/10.1017/S1351324913000284 Published online by Cambridge University Press

in the word that is syntactic-dependent, this is not always the case because of suffixes. Consider the word بيته (= بيت + ه), which means *his home*. With half diacritization it will be (بَيْته). Here the last two letters are syntactic-dependent, and in full diacritization it could be: *baituhu* (بَيْتُهُ), *baitahu* (بَيْتَهُ), or *baitihi* (بَيْتِهِ).

- Partial diacritization: Any other diacritization scheme (DS) where we diacritize a subset of letters. While studying the impact of diacritization on statistical machine translation (SMT), Diab, Ghoneim and Habash (2007) proposed further dividing partial diacritization into the following four different subschemes: an inflectional DS, which marks the verb passivization only, e.g. (سترمم الجدران); another inflectional DS that encodes both case and mood, e.g. (سترممُ الجدرانُ); a lexical DS, which marks only the words with shaddah (gemination) diacritic, e.g. (سترمّم الجدران); and another lexical DS that marks only the words having the *sukon* diacritic, e.g. (سترمم الجدْران).
- Minimalist diacritization: It uses the optimal number of diacritics to accomplish a certain desired effect. In the literature we came across two different desires: (a) find the optimal level of diacritization sufficient to yield the best ASR in Arabic, and (b) find the minimum number of diacritical markings needed to avoid text ambiguity for human readers. ASR is the act of transcribing speech information using a machine. The acoustic models of Arabic ASR systems are transcribed from Arabic scripts which are usually written without diacritics (Vergyri and Kirchhoff 2004). And this means loss of phonetic pronunciation information. Inserting the missing diacritics automatically will resolve this problem. Full diacritization has improved the state-of-the-art Arabic ASR (Kirchhoff and Vergyri 2005), but the optimal level of diacritization necessary to get the best results is yet to be investigated (Diab *et al.* 2007). For the second case our aim is to place the diacritics on certain letters only so that the ambiguity is fully resolved. One main reason for this is that it is more pleasant for the eyes than a fully diacritized text, which creates a much crowded space and puts heavy strain on eyes, especially when the vertical clearance between lines is small. Consider the following sentence: (درس محمد الدرس). It could either mean: *Mohammad studied the lesson*, or *Mohammad taught the lesson*. With minimal diacritics, it suffices to write, say (درّس محمد الدرس) and the reader understands that the second interpretation is the one intended. The same sentence when fully diacritized will look like (دَرَّسَ مُحَمَّدٌ الدَّرْسَ). Figure 1 presents another example. Typically, it is the religious texts in Arabic that are heavily diacritized. However, in recent years some publishers have opted for a minimal diacritized text. A full diacritized text slows down typists considerably, mostly to one-third (letter + SHIFT + diacritic) or even to one-fourth when shaddah is involved. In a minimally diacritized text only the critical letters will be diacritized, while the rest will be left without diacritization, alhough, in theory at least, we expect a greater through output from typists as fewer diacritics are to be typed. In reality typists struggle to decide where to apply and not apply the diacritics. This surely slows down typists with little or no improvement in typing speed

https://doi.org/10.1017/S1351324913000284 Published online by Cambridge University Press

when compared with a fully diacritized text. However, this is not an easy problem to solve, as Nelken and Shieber (2005) have noted: 'This raises an interesting optimization problem – what is the least number of vowel symbols that are required in order to ensure an unambiguous reading, and where should they be placed?' The authors continue, 'The optimization problem would then become one of finding the partially diacritized text with the minimal number of vowels that would be least ambiguous'. The fact that no work has been done in this area (to the best of our knowledge) shows that it indeed is a challenging problem. We believe the minimalist diacritization will be the next trend for researches working in this area.

4 Survey of Arabic diacritization techniques

Due to its importance, the problem of diacritization has been the focus of several studies. However, it is still an open area for researchers to improve accuracy and coverage of diacritization (Maamouri *et al.* 2006). The techniques used in diacritization can be classified into three main categories: rule-based, statistical approach, and hybrid approach. Before delving into further detail, we need some metric to measure system performance. There are two main measures that are used to report systems performance: Word Error Rate (WER) and Diacritization Error Rate (DER). WER is the percentage of incorrectly diacritized words in which at least one letter has a wrong diacritic. The percentage of incorrectly diacritized letters is denoted as DER.

4.1 Rule-based approach

Rule-based systems rely on the existence of well-formed rules to deduce a solution for a problem. They often exploit human knowledge to solve the problem intelligently and heuristically. Rule-based approaches have been successfully applied to many NLP problems with acceptable results once solid linguistic knowledge is provided. Adopting this approach is feasible for scarce-resource languages. However, the main drawback with this approach is the difficulty of maintaining the huge list of rules that cover all aspects of rich languages.

El-Imam (2003) used comprehensive Arabic-dependent rules supported by an exception words dictionary to assign each letter an appropriate sound. Shaalan (2010) introduced a morphological analyzer and a syntax analyzer for Arabic that was developed using rule-based approach. The morphological analyzer utilizes the Augmented Transition Network (ATN) to represent context-sensitive relation between stems and affixes. These two analyzers are the main components of diacritization systems, and thus these could be used to develop a diacritics restoration system that follows a rule-based approach.

4.2 Statistical approach

In this approach the model predicts the probability distribution of a sequence of words or characters depending on certain statistics such as their frequency in a

https://doi.org/10.1017/S1351324913000284 Published online by Cambridge University Press

data resource (Manning and Schtze 1999), that is they estimate the probability $P(w_1, w_2, \ldots, w_n)$, where $w_1, w_2, \ldots, w_n$ are sequence of words. One main advantage of adopting this scheme for the problem at hand is that there is no need for linguistic knowledge or tools such as POS taggers, morphological analyzers, etc. But then these approaches require a large corpus of fully diacritized texts. The corpus may suffer from data sparseness due to the inflection property of Arabic, which causes many words to be unseen in the corpora. There are several sub-models within this approach: Hidden Markov Model (HMM), n-gram, SMT, and finite state transducers (FST).

Gal (2002) proposed a statistical approach using HMM to restore diacritics of Arabic and Hebrew. The proposed system used the Holy Qur'an as a corpus and achieved a success rate of 86%. The method used in solving the diacritization problem did not involve any language-specific knowledge. The hidden states in the proposed model were the diacritized words of the training data. Each of these hidden states had a single outcome, formally, the observation that represented the undiacritized form of the hidden state word. Given a sequence of observations which are undiacritized words, $W_{1,n} = w_1, w_2, w_3, \ldots, w_n$, the Viterbi algorithm was used to find the most likely path through the model, which is defined as follows:

$$\begin{aligned} \text{argmax}_{T_{1,n}} P(T_{1,n}|W_{1,n}) &= \text{argmax}_{T_{1,n}} P(W_{1,n}, T_{1,n})/P(W_{1,n}) \\ &= \text{argmax}_{T_{1,n}} P(W_{1,n}, T_{1,n}) \end{aligned}$$

where $T_{1,n} = T_1, T_2, T_3, \ldots, T_n$ is the most probable diacritized words' sequence through the model.

Emam and Fisher (2004) invented a method that does not demand a skilled language expert. Instead the system tries to find the longest units, sentences, and phrases. In case the system fails to find such sentences, an n-gram model is applied for words and then for characters.

Nelken and Stuart (2005) proposed a system for Arabic diacritization in MSA using a probabilistic model expressed as a weighted finite state transducers. The model transduces fully the diacritized Arabic text into the undiacritized text and learns weights by means of statistical language model, namely a trigram word-based language model and a letter-based language model. The letter-based model generates diacritization schema that is only used for words that have not been seen in the training data. Their system used Linguistic Data Consortium (LDC) Arabic Treebank and achieved 7.33% word accuracy without case ending and 23.61% with case ending.

Ananthakrishnan *et al.* (2005) used word-level trigram model combined with four-gram character model. For the trigram word-level model, the diacritizer picks the diacritized form of an input word according to the following equation:

$$w_d^* = \text{argmax}_{w_1^d, w_2^d, \ldots, w_n^d} \prod_{i=2}^{n} P(w_i^d, w_i^u | w_{i-1}^d, w_{i-1}^u, w_{i-2}^d, w_{i-2}^u)$$

https://doi.org/10.1017/S1351324913000284 Published online by Cambridge University Press

For the unseen words, the system adopted the four-gram character model. To build the language model, the authors used LDC Arabic Treebank. The model was tested on LDC TDT4 Broadcast News data set. The reported word accuracy using this model was 80.21%.

Elshafei, Al-Muhtaseb and Alghamdi (2006) proposed a diacritic restoration system that is similar to Gals (2002). They used HMM for modeling and Viterbi algorithm to find the most probable diacritized form of a sentence. The training data had 18,623 diacritized words from different knowledge domains, while the test contained 995 words selected from fifty randomly picked verses from the Holy Qur'an. The result was 4.1% errors in the diacritical marking of letters. The authors argued that this error rate could be minimized to 2.5% by normalizing input text and applying some post-processing operations, such as syntax analyzer, to improve end cases of words.

KACST Arabic Diacritizer (KAD) is a simple and purely statistical-based diacritizer that relies on quad-gram letters (Alghamdi *et al.* 2006; Alghamdi and Muzaffar 2007; Alghamdi, Muzaffar and Alhakami 2010). The input to the system is a sequence of undiacritized letters. The proposed technique had two steps. The first step was to create a database with the most frequent diacritized quad-gram patterns along with their probabilities. The probability associated with the highest probable quad-gram is calculated as F_h/F_t, where F_h is the frequency of the highest quad-gram sequence, and F_t is the total frequency of that quad-gram. For the next step, the input letters are diacritized by retrieving from the database all the quad-grams constituting these letters. For each letter, all the occurrences, which are at most four quad-grams, are considered and the probabilities of the sequences that have the same diacritic for that letter are summed. The diacritic with the highest aggregated probability is chosen for that letter. The system used a dataset from LDC that contains 15,983 words with 83,897 letters. The authors reported an error rate of 13.83% once, including the last letter, and 9.25% if it was excluded.

Schlippe, Nguyen and Vogel (2008) treated the diacritization problem as a MT problem and sequence labeling problem. Their proposed translation system uses a pure SMT with several models. The translation model is built for a phrase-based system, where phrases were diacritized with a word level model. However, if the word is not in the training data, then character level is used. The authors used a training set of 23,000 sentences with 613,000 words extracted from LDC Arabic Treebank. Their system achieved a reported word error rate of 19.9%, which included the last letter.

Hifny (2012a, 2012b) proposed a diacritic restoration system that uses dynamic programming (DP), n-gram language model, and smoothing. The possible word sequences with diacritics are assigned scores using the n-gram language model. With the scores assigned, dynamic programming's task is to search for the most likely sequence. To solve the problem of unseen n-grams in the training data, the author uses smoothing. The aim of smoothing is to take some probability mass from the observed n-gram and distribute it to the unseen ones. The author experimented with different smoothing techniques that are commonly used in speech recognition and MT fields: Katz, modified Kneser–Ney, and absolute discounting. For training and

https://doi.org/10.1017/S1351324913000284 Published online by Cambridge University Press

testing, the author used Tashkeela (Zerrouki 2011), a corpus of 6.15 million words collected out of Islamic religious books using automatic web crawling. The size of the training data was 5.25 million words, while the test data had about 1.9 million words. Using quad-gram language model and absolute discounting gave the finest result. The best reported word error rate was 3.4% when end cases are ignored and 8.9% when including the last letter.

4.3 Hybrid techniques

It is argued that any Arabic NLP that does not take into account its rich properties is thought to be inappropriate (Farghaly and Shaalan 2009). Combining linguistic knowledge with other techniques, such as stochastic techniques or MT techniques, will mostly yield better results. This should not be unexpected as these techniques will be guided by language-dependent rules. For example, assume that the word (مِنْ), a preposition, was repeated in a data set one hundred times, while the word (مَنْ), a relative pronoun, was repeated ten times; then the word (من) in the sentence (المسلم من سلم المسلمون من لسانه) using purely stochastic approach will always be a preposition since it has a higher frequency. However, activating a linguistic rule such as a relative pronoun in Arabic is followed by a sentence or semi-sentence and a preposition is always followed by a noun increases the chance of correctly diacritizing the word (من) in both places.

Ananthakrishnan *et al.* (2005) did another experiment in which they took some linguistic features into account using BAMA (Buckwalter 2004). Here the authors used their trigram model to rank Buckwalter's word lists. The WER was slightly improved. Using POS tagger, the experiment results were further improved to 86.50%.

Habash and Rambow's (2007) proposed system achieves good results, especially when ignoring the last letter. This system has two main subsystems: Morphological Analysis and Disambiguation of Arabic (MADA), and the standard n-gram language model. The MADA subsystem exploits all fourteen dimensions of inflection in Arabic and has three main phases: (a) use BAMA (Buckwalter 2004) to get a list of all potential analysis of a word; (b) for each of the dimensions of inflection, a Support Vector Machine (SVM) classifier is applied to the word to predict the value of that dimension. The classifiers were trained on the annotated Arabic Treebank; and (c) the prediction of the classifier is used to choose from the analyses proposed by BAMA. The ones that match most of the dimensions win. The MADA subsystem narrows the potential morphological analysis of a word; however, it could still have more than one possible analysis. It is the function of the n-gram lexeme model, the second subsystem, to choose among the narrowed potential list. The system achieves a reported 5.5% word error rate when the last letter is ignored, and 14.9% word error rate when diacritizing includes the last letter.

Schlippe *et al.* (2008) treated the diacritization problem as a sequence labeling problem and proposed a solution using conditional random fields (CRFs) with integration of a POS feature of the previous, current, and the next word. This method performed worst with their MT-based technique (Section 4.2). The system

https://doi.org/10.1017/S1351324913000284 Published online by Cambridge University Press

achieved a reported word error rate of 21.9%, including the last letter. The authors anticipate that a better performance may be achieved if more features are used.

The statistical system proposed in Zitouni and Sarikaya (2009) uses maximum entropy approach for restoring short vowels and other diacritics. Their goal is to convert the knowledge used by native speakers into features and incorporate them into their models. These features range from lexical, segment-based, and POS tag. They used LDC Arabic Treebank and assume that the input text to be diacritized is free from diacritics. The system achieves a word error rate of 7.9% when the last letter is being ignored and 18% word error rate when diacritizing all the letters, including the last letter.

Shaalan, Abo Bakr and Ziedan (2009) proposed a system that has two main stages. The first stage is to diacritize internal characters, while the last letter is left for the second stage. The first stage uses two dictionary-based techniques (i.e. lookup process) to retrieve diacritized word/expression from a lexicon or bigram database and the SVM technique, which chooses the correct diacritized form of the word by means of POS. The second stage uses SVM to relate a case-ending of a word depending on its POS and its position within the sentence. The system achieves a reported word error rate of 12.16% taking into account the last letter.

Rashwan *et al.* (2011) introduced a hybrid system that combines the unfactorized system (dictionary-based system) and the factorized system (depending on morphological analyzer) to gain the advantages of both systems: coverage, speed, and accuracy. The proposed system consists of two phases: off-line and on-line phases. In the off-line phase, a dictionary model and a language model are built. In the on-line (run-time) phase, the input words are searched in the dictionary. If the word is found, then all its diacritized forms are retrieved, constituting a lattice. This lattice is disambiguated via the n-gram probability estimation and A* lattice search to infer the most likely sequence. The diacritized full-form words are then concatenated to unanalyzable segments and passed to the factorizing module. In the factorizing module, the words that were undiacritized in the first layer are factorized into its possible morphological components (prefix, root, pattern, and suffix) and again goes through the n-gram probability estimation and A* lattice search algorithm. The reported word error rate was 3.1% for inner letters and 12.5% when including the last letter.

The hybrid system by Hattab and Hussain (2012) combines SMT-based diacritizer and another diacritizer that is based on morpho-syntactical knowledge. Initially the sentences are diacritized using the statistical model, then forwarded to the morpho-syntactic diacritizer. The second diacritizer is based on the Arabic morphological analyzer, ElixirFM (Smrž 2007), and is especially helpful when faced with words out of vocabulary. To function, the system relies on a huge unique words' list that covers most of the common Arabic words. The list was created from a corpus of about eight million diacritized words. Three different sets of test data were created: a 50,000-word corpus randomly collected from the training set; a corpus of 50,000 words from sources outside the training set and mainly from the classical Arabic literature and Islamic domain; and a 10,000-word corpus from a diverse domain outside the training set. The reported error rate was 0.6%, 5.2%, and 14.1% (respectively), which includes the last letter.

https://doi.org/10.1017/S1351324913000284 Published online by Cambridge University Press

4.4 Summing up the reviewed systems

To complement the comparison of different systems, we will include the study in Bahanshal and Al-Khalifa (2012), which evaluated a commercial system Sakhr (`http://arabdiac.sakhr.com.eg/ArabDiac.aspx`); an open source diacritizer Mishkal (`http://tahadz.com/mishkal`); and KAD, a system that we have reviewed earlier. Neither of the two listed websites provide any technical details about their respected system. These three systems were tested to fully diacritize all the letters, including the last on a single chapter from the Holy Qur'an (Surah Al-Fatiha), and a short poem. For the Holy Qur'an, the authors reported a letter accuracy of 94.5%, 86.8%, and 71.4% for KAD, Sakhr, and Mishkal respectively. While for the poem they reported a letter accuracy of (in the same order) 66.6%, 77.7%, and 69.4% respectively. We can combine and express the results in terms of DER as follows. Let $|T_1|, Acc_1$ be the number of letters in the first sample text and its respective reported letter accuracy, and similarly is $|T_2|, Acc_2$. We can now calculate DER as

$$\text{DER} = \left(1 - \frac{|T_1| \cdot Acc_1 + |T_2| \cdot Acc_2}{|T_1| + |T_2|}\right) \times 100$$

The calculated DER for these systems is 12.6%, 15.5%, and 29.1% for KAD, Sakhr, and Mishkal respectively. In reporting the accuracy, the authors considered the syntactic-dependent letters as well, so the calculated DER includes the case ending. Although the DER for the KAD system is slightly different from the one reported by the authors themselves (Alghamdi *et al.* 2010), which is quite reasonable as the test data were different in both studies, nevertheless we can get a picture of relative performance of the other two systems, i.e. Sakhr and Mishkal.

Table 3 summarizes the systems surveyed. Although we compare the systems in terms of WER and DER, but as different systems use different datasets, we have to be careful when interpreting the results. The exception being Habash and Rambow (2007) and Zitouni and Sarikaya (2009), as these both used the same dataset. We feel that hybrid-based schemes are a better choice, for they combine human knowledge along with some intelligent techniques. None of the surveyed systems support partial diacritization.

Overlooking Table 3 we see some impressive schemes. Still we feel there is a large room for improvement. Consider the scheme used in Rashwan *et al.* (2011). This scheme has the best reported DER. It delivers a DER of 3.8% for all the diacritics, which means 96.2% correctly diacritized characters. Assuming a typical page having 450 Arabic words and with an average of five characters per word (Alotaiby, Alkharashi and Foda 2009), this translates to about 2,200 characters per page. Using the above system to fully diacritize the text in a typical page will result in a page with eighty-three wrongly diacritized characters. How much time will he/she spend trying to catch and correct these errors? Certainly it is more difficult to spot a wrongly diacritized character than, say, a wrongly spelled word out of, for example an optical character recognition (OCR) program.

https://doi.org/10.1017/S1351324913000284 Published online by Cambridge University Press

https://doi.org/10.1017/S1351324913000284 Published online by Cambridge University Press

Table 3. *Summary of the systems surveyed. The systems are given in chronological order of their appearance in the literature with newer ones at the top. WER is the word error rate, which is the percentage of incorrectly diacritized words, and DER is the diacritization error rate, which is the percentage of erroneously diacritized letters. The smaller the WER/DER, the better the performance. Blank entries mark information not available*

System	Technique	Dataset	Ignore last[a]		All diacritics[b]	
			WER	DER	WER	DER
Hifny	Statistical	1,902,145 words[c]	3·40		8·90	
Hattab *et al.*	Hybrid	50,000 words[d]				5·20
Sakhr	Commercial	<100 words[e]				15·50
Mishkal	Open source	<100 words[f]				29·10
Rashwan *et al.*	Hybrid	52,000 words (ATB3)	3·10	1·20	12·50	3·80
KAD[g]	Statistical	15,983 words (LDC)	26·03	9·25	46·83	13·83
Zitouni *et al.*	Hybrid	52,000 words (ATB3)	7·90	2·50	18·00	5·50
Shaalan *et al.*	SVM	38 ATB2.2 files	31·86	7·92	12·16	3·78
Habash *et al.*	SVM	52,000 words (ATB3)	5·50	2·20	14·90	4·80
Schippe *et al.*	SMT	613,000 words (LDC)	6·80	1·70	19·90	4·30
Schippe *et al.*	CRFs	613,000 words (LDC)	8·40	1·90	21·90	4·70
Elshafei *et al.*	HMM	995 words (Holy Qur'an)				4·10
Ananthakrishnan *et al.*	Statistical	LDC TDT4			19·69	
Ananthakrishnan *et al.*	Hybrid	LDC TDT4			19·73	
Nelken *et al.*	Weighted FST	Hayat corpus	7·33	6·35	23·61	12·79
Gal	HMM	Holy Qur'an	14·00			

[a] Ignore last letter since its diacritic is often determined syntactically (half diacritization).
[b] Taking into account the syntactic dependent letters (full diacritization).
[c] Tashkeela corpus (Zerrouki 2011).
[d] Proprietary corpus compiled from classical Arabic literature and Islamic domains.
[e] A single chapter from the Holy Qur'an and a short poem.
[f] Same as in Sakhr.
[g] Figures are taken from Alghamdi *et al.* (2010).

5 A critical look at different techniques

When comparing different techniques, we need to address the following:

(1) *Human expert availability*. Exploiting the knowledge of linguistic experts is the backbone of solving the problem of diacritization. Rule-based technique needs linguistic experts as a reference in writing the rules. Pure statistical approaches can do without linguistics. Hybrid approaches in many surveyed systems perform better as these techniques are guided by language-dependent rules, however they may cost more.

(2) *Error tolerance*. Some techniques may have the ability to present uncertain results, while others are strict in presenting the correct results. Pure rule-based approach cannot diacritize many sentences correctly due to lack of applied rules. However, the same sentences can actually be diacritized using statistical approaches, since the latter approach usually uses smoothing techniques to estimate probability of unseen data.

(3) *Availability of large training data resources*. A large corpus of fully diacritized texts is essential for techniques that need learning. Inflection property of Arabic may cause many words to be unseen in learning phase. Rule-based technique is feasible for Arabic diacritization in that Arabic could be considered as a scarce-resource language with high sparseness as it is morphologically rich. For hybrid approaches, a simple corpus is not sufficient. It has to be an expert-annotated corpus, e.g. Treebanks.

(4) *Generalization*. By this we mean the ability to diacritize any text either seen or unseen. Having an intensive list of rules makes the rule-based approach good for unseen texts. However, building such a list that covers all aspects of Arabic is truly a difficult task. Pure statistical approaches usually give unsatisfactory performance with unseen data, especially in complex languages that suffer from sparseness as is the case with Arabic, a highly inflected language. This sparseness may cause training data to be insufficient.

(5) *Ability to explain outcome to users*. This criterion is more applicable to rule-based techniques than to the statistical-based ones. The former are strict and their results can always be traced back to a certain rule. This is not the case with the latter technique as it depends on probability distribution.

As we have stated before, rule-based approach performs well once we have an extensive collection of well-formed rules with the help of expert linguists. Rule-based approach outperforms statistical approach in terms of resources' cost since it does not require a hefty memory for training model. However, due to lack of such rules, researchers tend to employ statistical techniques either purely or integrated with some linguistic properties. According to the surveyed systems, the statistical techniques employed in the diacritization problem were HMM, n-gram language models, Maximum Entropy model, conditional random fields and SMT.

For diacritization schemes, which are statistical-based, we feel that HMM and n-gram models are the most suitable ones since they follow the Markov rule, which states that the probability of diacritizing one word may depend on the words preceding it. Consider for example the verse from the Holy Qur'an, (حملته أمه وهنا على وهن).

https://doi.org/10.1017/S1351324913000284 Published online by Cambridge University Press

If we were to auto diacritize the word (وهن) then most probable it will be diacritized correctly as (وَهْن) *fainting* rather than (وَهُنّ) *and they are* (feminine, plural), which is the incorrect one. This is because these models keep track of word's history and the previous word, which is the preposition (على). However, adopting HMM for diacritization either at word level or at letter level, where the observation is a sequence of words/letters and the hidden states are the diacritized words/letters, suffers from the lack of integrating linguistic features, a scheme known to significantly improve error rates. Repeating the example from Section 4.3 (المسلم من سلم المسلمون من لسانه). Here the difficulty was in accurately diacritizing the twice occurring word (من), first time as a relative pronoun and the second time as a preposition. The correctly diacritized sentence should be (المسلم مَن سلم المسلمون مِن لسانه). A purely statistical scheme will erroneously diacritize both occurrences as (مِن), a preposition, which is a more prevalent form. This problem could have been resolved easily by enforcing some simple linguistic rules. In addition, HMM fails to give a reasonable estimation of words sequence when a word is not in the training set. This requires diacritizing these words based on letter statistics. One more important issue is that the HMM technique in its current use is not good for words' last letter or more specifically the syntactic-dependent letters. We suggest exploiting HMM at letter's level as a post-processing procedure to diacritize the syntactic-dependent letter in each word where the states represent these letters.

N-gram statistical language models can be used in diacritization problem as a stand-alone approach at either sentence, word, or letter level. But once it is incorporated with other approaches, such as rule-based or machine learning technique, the error rates decrease significantly. It is one of the most commonly employed approaches due to difficulty in extracting well-formed rules for Arabic morphology and syntax. The accuracy of an n-gram model of a corpus depends on two factors: the choice of n, and the smoothing technique used to estimate probabilities to infrequent or unseen words in the training data set. We gain accuracy with larger n values, but this means a much larger model, which will further constrain resources. Usually the model will grow exponentially in n. Besides, larger n values require a larger training data set as the grams tend to be more sparse (Liu *et al.* 2005).

The conditional random fields (CRFs) can be considered as an extension of Maximum Entropy (ME) model. While the CRF method is over the entire sequence, the ME model makes a decision for each state independent of other states. These both are conditional models that estimate the probabilities of possible label sequences given observation sequences without the need to expend the modeling effort on observations. As conditional modeling, CRF and ME classifiers can depend on observation sequence's arbitrary non-independent features, even though the model is not forced to account for the distribution of dependencies. These will be promised characteristics for diacritization problem as they can efficiently integrate and make effective use of diverse types of information, e.g. POS tags, lexical information, etc. The CRF approach requires fewer features than the ME model since the former's training is computationally expensive as it is a structured learning rather than a single-position learning (Klinger and Tomanek 2007).

https://doi.org/10.1017/S1351324913000284 Published online by Cambridge University Press

Support Vector Machine classifier uses linguistic features extracted from the input to predict its class depending on what it learned earlier from the training data. The robustness of this technique in NLP tasks lies in good generalization performance in the classification of data with overlapping features (Habash and Rambow 2007). For the diacritization problem, SVM classifier is used to classify undiacritized words for some features such as POS, gender, number, case ending, etc. And if the system is integrated with a morphological analyzer, then diacritics can be predicted. This technique reports acceptable error rates; however, it needs a comprehensive annotated corpus for all features that will be used for classification.

6 Summary and conclusion

Arabic is a sophisticated language. Its morphology is highly inflected and derivative, and its syntax is rich and vague even for the natives. In this paper we briefly discussed the properties of the Arabic language that are related to diacritization. We surveyed and critically reviewed many techniques that were used to automate the diacritization of text. The techniques can broadly be classified as rule-based, stochastic, and hybrid approaches. These approaches were compared from different points of view. It was observed that the hybrid schemes yield higher accuracy when compared with other two. With so many different techniques applied to the diacritization problem, we feel there is still a room for further research for suitable techniques in terms of accuracy and coverage. We also mentioned minimalist diacritization and lack of research in this area. The objective is to use the optimal number of diacritical marks, enough to avoid ambiguity in the written text or sufficient to yield the best ASR. This is one possible future research direction in this area.

Acknowledgments

We would like to thank all three anonymous reviewers whose comments greatly improved our presentation. This work was supported by a special fund of the Research Center of the College of Computer and Information Sciences (CCIS) at King Saud University.

References

Al-Azami, M. M. 2011. *The History of the Qur'anic Text: From Revelation to Compilation*, 2nd ed., pp. 123–9. Sherwood Park, Alberta, Canada: Al-Qalam Publishing.

Alghamdi, M., Khursheed, M., Elshafei, M., Alhargan, F., Alkanhal, M., Alshamsan, A., Alqahtani, S., Muzaffar, Z., Altowim, Y., Yusuf, A., and Almuhtasib, H. 2006. Automatic Arabic text diacritizer. Technical Report CI.25.02, King Abdulaziz City for Science and Technology (KACST), Riyadh, Saudi Arabia.

Alghamdi, M., and Muzaffar, Z. 2007. KACST Arabic diacritizer. In *Proceedings of the First International Symposium on Computers and Arabic Language*, Riyadh, Saudi Arabia.

Alghamdi, M., Muzaffar, Z., and Alhakami, H. 2010. Automatic restoration of Arabic diacritics: a simple, purely statistical approach. *Arabian Journal for Science and Engineering* **35**(2C): 125–35.

Ali, N. 1988. *Arabic Language and Computer* (in Arabic). Cairo, Egypt: Ta'reep.

https://doi.org/10.1017/S1351324913000284 Published online by Cambridge University Press

Alotaiby, F., Alkharashi, I., and Foda, S. 2009. Processing large Arabic text corpora: preliminary analysis and results. In *Proceedings of the 2nd International Conference on Arabic Language Resources and Tools*, Cairo, Egypt, pp. 78–82.

Al-Sughaiyer, I., and Al-Kharashi, I. 2004. Arabic morphological analysis techniques: a comprehensive survey. *Journal of American Society for Information Science and Technology* **55**(3): 189–213.

Ananthakrishnan, S., Narayanan, S., and Bangalore, S. 2005. Automatic diacritization of Arabic transcripts for automatic speech recognition. In *Proceedings of the International Conference on Natural Language Processing (ICON-05)*, Kanpur, India.

Attia, M. 2000. *A Large Scale Computational Processor of the Arabic Morphology and Applications* (unpublished Master's thesis), Cairo University, Cairo, Egypt.

Attia, M. 2008. *Handling Arabic Morphological and Syntactic Ambiguity within the LFG Framework with a View to Machine Translation* (unpublished D.Sc thesis), University of Manchester, Manchester, UK.

Bahanshal, A., and Al-Khalifa, H. 2012. A first approach to the evaluation of Arabic diacritization systems. In *Proceedings of the 7th International Conference on Digital Information Management (ICDIM 2012)*, Macau, China.

Bellamy, J. 1988. Two pre-Islamic Arabic inscriptions revised: Jabal Ramm and Umm Al-Jimal. *Journal of the American Oriental Society* **108**(3): 369–72.

Buckwalter, T. 2004. *Buckwalter Arabic Morphological Analyzer Version 2.0*. Philadelphia, PA: Linguistic Data Consortium (LDC).

Central Intelligence Agency. 2008. *World Factbook*. Washington DC: CIA.

Debili, F., Achour, H., and Souissi, E. 2002. De l'etiquetage grammatical a la voyellation automatique de l'arabe. Technical Report, Correspondances de l'Institut de Recherche sur le Maghreb Contemporain 17.

Diab, M., Ghoneim, M., and Habash, N. 2007. Arabic diacritization in the context of statistical machine translation. In *Proceedings of Machine Translation Summit XI(MT-Summit)*, Copenhagen, Denmark.

El-Imam, Y. 2003. Phonetization of Arabic: rules and algorithms. *Computer Speech and Language* **18**: 339–73.

Elshafei, E., Al-Muhtaseb, H., and Alghamdi, M. 2006. Statistical methods for automatic diacritization of Arabic text. In *Proceedings of Saudi 18th National Computer Conference (NCC18)*, Riyadh, Saudi Arabia, pp. 301–6.

Emam, O., and Fisher, V. 2004. A hierarchical approach for the statistical vowelization of Arabic text. Technical Report, IBM patent led DE9–2004–0006, US Patent Application US2005/0192809 A1.

Farghaly, A., and Shaalan, K. 2009. Arabic natural language processing: challenges and solutions. *ACM Transaction on Asian Language Information Processing* **8**(4): 1–22.

Gal, Y. 2002. An HMM approach to vowel restoration in Arabic and Hebrew. In *Proceedings of the ACL-02 Workshop on Computational Approaches to Semitic Languages (SEMITIC '02)*, Philadelphia, PA, pp. 27–33.

Habash, N. 2010. Introduction to Arabic language processing. In G. Hirst (ed.), *Synthesis Lectures on Human Language Technologies*. San Rafael, CA: Morgan & Claypool.

Habash, N., and Rambow, O. 2007. Arabic diacritization through full morphological tagging. In *Human Language Technologies: The Conference of the North American Chapter of the Association of Computational Linguistics (NAACL '07)*, Rochester, NY, pp. 53–6.

Habib, M. 2008. *An Intelligent System for Automated Arabic Text Categorization* (unpublished Master's thesis), Ain Shams University, Cairo, Egypt.

Hattab, A. M., and Hussain, A.K. 2012. Hybrid statistical and morpho-syntactical Arabic language diacritizing system. *International Journal of Academic Research (Part A)* **4**(4): 51–6.

Hifny, Y. 2012a. Smoothing techniques for Arabic diacritics restoration. In *Proceedings of the 12th Conference on Language Engineering (ESOLEC '12)*, Cairo, Egypt.

https://doi.org/10.1017/S1351324913000284 Published online by Cambridge University Press

Hifny, Y. 2012b. Higher order n-gram language models for Arabic diacritics restoration. In *Proceedings of the 12th Conference on Language Engineering (ESOLEC '12)*, Cairo, Egypt.

Ibn Mandhur. 2009. *Lisan al-Arab*, A. A. Haider and A. Ibrahim (eds.). Beirut, Lebanon: Dar al-Kutub al-Ilmiyah (in Arabic).

Kirchhoff, K., and Vergyri, D. 2005. Cross-dialectal data sharing for acoustic modeling in Arabic speech recognition. *Speech Communication* **46**: 37–51.

Klinger, R., and Tomanek, K. 2007. Classical probabilistic models and conditional random fields. Technical Report TR07-2-013, Department of Computer Science, Dortmund University of Technology, Dortmund, Germany.

Liu, Y., Shriberg, E., Stolcke, A., and Harper, M. 2005. Comparing HMM, maximum entropy, and conditional random fields for disfluency detection. In *Proceedings of European Conference on Speech Communication and Technology*, Lisbon, Portugal, pp. 3313–6.

Maamouri, M., and Bies, A. 2010. The Penn Arabic treebank. In A. Farghaly (ed.), *Arabic Computational Linguistics*. Stanford, CA: CSLI.

Maamouri, M., Bies, A., and Kulick, S. 2006. Diacritization: a challenge to Arabic treebank annotation and parsing. In *Proceedings of Arabic NLP/MT Conference*, The British Computer Society, Natural Language Translation Specialist Group, London.

Manning, C., and Schtze, C. 1999. *Foundations of Statistical Natural Language Processing*, 2nd ed. Cambridge, MA: MIT Press.

Nelken, R., and Shieber, S. 2005. Arabic diacritization using weighted finite-state transducers. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages (SEMITIC '05)*, Ann Arbor, MI, pp. 79–86.

Rashwan, M., Al Badrashiny, M., Attia, M., Abdou, S., and Rafea, A. 2011. A stochastic Arabic diacritizer based on a hybrid of factorized and unfactorized textual features. *IEEE Transaction on Audio, Speech, and Language Processing* **19**(1): 166–75.

Ryding, K. 2006. *A Reference Grammar of Modern Standard Arabic*. Cambridge, UK: Cambridge University Press.

Salton, G. 1989. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Reading, MA: Addison-Wesley.

Schlippe, T., Nguyen, T., and Vogel, S. 2008. Diacritization as a machine translation problem and as a sequence labeling problem. In *Proceedings of the 8th Conference of the Association for Machine Translation in Americas (AMTA-2008)*, Waikiki, HI, pp. 192–201.

Shaalan, K. 2010. Rule-based approach in Arabic natural language processing. *International Journal on Information and Communication Technologies (IJICT)*, Serial Publications **3**(3): 11–9 (Special Issue on Advances in Arabic Language Processing).

Shaalan, K., Abo Bakr, H., and Ziedan, I. 2009. A hybrid approach for building Arabic diacritizer. In *Proceedings of EACL 2009 Workshop on Computational Approaches to Semitic Language*, Morristown, NJ, pp. 27–35.

Smrž, O. 2007. *Functional Arabic Morphology: Formal System and Implementation* (unpublished PhD thesis), Charles University in Prague, Prague, Czech Republic.

Van Gompel, M. 2008. *Automatic Arabic Vocalisation*. Tilberg, Netherlands: University of Tilburg.

Vergyri, D., and Kirchhoff, K. 2004. Automatic diacritization of Arabic for acoustic modeling in speech recognition. In *Proceedings of the Workshop on Computational Approaches Arabic Script-Based Languages (SEMITIC '04)*, Stroudsburg, PA, pp. 66–73.

Wikipedia. n.d. Danish and Norwegian alphabet. Retreived March 17, 2013, from `http://en.wikipedia.org/wiki/Danish_and_Norwegian_alphabet`.

Zerrouki, T. 2011. Tashkeela: Arabic vocalized text corpus. Retreived June 9, 2013, from `http://aracorpus.e3rab.com/`.

Zitouni, I., and Sarikaya, R. 2009. Arabic diacritic restoration approach based on maximum entropy models. *Computer Speech and Language* **23**: 257–76.

https://doi.org/10.1017/S1351324913000284 Published online by Cambridge University Press