# COMBINING SPEECH WITH TEXTUAL METHODS FOR ARABIC DIACRITIZATION

**AISHA SIDDIQA AZIM**

***B.Sc. (Hons), LUMS***

**A THESIS SUBMITTED FOR THE DEGREE OF MASTER OF SCIENCE**

**SCHOOL OF COMPUTING**

**NATIONAL UNIVERSITY OF SINGAPORE**

**2012**

# i. Acknowledgements

I am truly and humbly grateful to my supervisor, Dr Sim Khe Chai, for his immense patience with me as I slowly trudged through this work, his continual advices and suggestions, and all of his enormously valuable criticisms and encouragements. I have certainly learned a great deal from him!

A huge, warm thanks to Xiaoxuan Wang for putting up with all the endless subtleties of a completely new language, continuous changes in requirements and plain old hard work, and pulling it off so amicably well!

I would also like to thank Li Bo and Joey Wang for their help every time I randomly walked into problems using HTK.

This work would not be complete without the timely responsiveness and cooperation of Tim Schlippe, Nizar Habash and Owen Rambow.

It goes without saying, I owe everything to my family. And especially to my dear mother.

After God, everything in my life that I've managed to scrape around to getting done is because of her, and everything I haven't is because I wasn't listening!

## Contents

## ii. Summary

Arabic is one of the six most widely used languages in the world. As a Semitic language, it is an *abjad* system of writing, which means that it is written as a sequence of consonants without vowels and other pronunciation cues. This makes the language challenging for non-natives to read and for automated systems to process.

Predicting the vowels, or *diacritics*, in Arabic text is therefore a necessary step in most Arabic Natural Language Processing, Automatic Speech Recognition, Text-to-Speech systems, and other applications. In addition to the writing system, Arabic also possesses rich morphology and complex syntax. *Case endings*, the diacritics that relate to syntax, have particularly suffered from a higher prediction error rate than the rest of the text. Current research is *text-based*, that is, it focuses on solving the problem using textually inferred information alone. The state-of-the-art systems approach diacritization as a lattice search problem or classification problem, based on morphology. However, predicting the case endings remains a complex problem.

This thesis proposes a novel approach. It explores the effects of combining speech input with a text-based model, to allow the linguistically insensitive information from speech to correct and complement the errors generated by the text model's predictions. We describe an acoustic model based on Hidden Markov Models and a textual model based on Conditional Random Fields, and the combination of acoustic features with linguistic features.

We show that introducing speech to diacritization significantly reduces error rates across all metrics, especially case endings. Within our combined system, we incorporate and compare the use of one of the established SVM-based diacritization systems, MADA, against our own CRF-based model, demonstrating the strengths of our model. We also make an important comparison between the use of two popular

tools in the industry, BAMA and MADA, in our system. In improving the underlying text-based diacritizer, we briefly study the effects of linguistic features at three different levels that have not previously been explored: phrase-, word- and morpheme-level.

The results reported in this thesis are the most accurate reported to date in the literature. The diacritic and word error rates are 1.6 and 5.2 respectively, inclusive of case endings, and 1.0 and 3.0 without them.

## iii. List of Tables

## iv. List of Figures

## v. List of Abbreviations

| | |
|---|---|
| **ASR** | Automatic Speech Recognition |
| **ATB3** | Penn Arabic Treebank, Part 3, version 1.0 |
| **BAMA** | Buckwalter Arabic Morphological Analyzer |
| **CRF** | Conditional Random Field |
| **DER** | Diacritic Error Rate |
| **$\mathbf{DER}_{\mathbf{abs}}$** | Diacritic Error Rate (absolute) |
| **FST** | Finite State Transducer |
| **GALE** | Global Autonomous Language Exploitation |
| **HMM** | Hidden Markov Model |
| **HTK** | Hidden Markov Model Toolkit |
| **LDC** | Linguistic Data Consortium |
| **LM** | Language Model |
| **MADA** | Morphological Analysis and Disambiguation for Arabic |
| **MFCC** | Mel Frequence Cepstral Coefficient |
| **MLE** | Maximum Likelihood Estimation |
| **MSA** | Modern Standard Arabic |
| **NLP** | Natural Language Processing |
| **POS** | Part of Speech |

| | |
|---|---|
| **SMT** | Statistical Machine Translation |
| **STT** | Speech-to-text |
| **SVM** | Support Vector Machine |
| **TTS** | Text-to-speech |
| **WER** | Word Error Rate |

# 1. Introduction

*Arab language is not merely the richest language in the world. Rather, those who excelled in … it are quite innumerable.*

- Aswan Ashblinger

Arabic is one of the six most widely spoken languages in the world[1], and the vehicle of a rich cultural and religious tradition that finds its roots in 6th Century AD and continues to be an important influence in the world today. While it has evolved subtly over time and space and is expressed colloquially in a number of dialectical forms, the lingua franca of the Arab world remains Modern Standard Arabic (MSA), and it is this standardized form that will be dealt with in this thesis. Increased automation in daily life pulled Arabic into the field of computational linguistics in the nineteen-eighties, but only in the past decade have widely-recognized research efforts been made as part of the internationalization process. One of the most fundamental aspects of automating processes in any language is *disambiguating words in the script*.

## 1.1 Arabic Diacritization

The Arabic alphabet consists of 28 consonants. The vast majority of nouns, adjectives and verbs in Arabic are generated from roots that comprise a combination of only three core consonants. Given the language's highly inflective nature and morphological complexity, a single sequence of three consonants could easily represent over 100 valid words. To disambiguate the different words that could be represented by a single set of consonants, short vowels and other phonetic symbols are used. However, Arabic is an *abjad* system of writing, so the script is written as a sequence of consonants. Short vowels are included only as optional *diacritics*.

[1] http://en.wikipedia.org/wiki/List_of_languages_by_number_of_native_speakers

This does not pose serious problems to native readers, who are familiar enough with the language to contextually infer the correct pronunciation of the script; the vast majority of Arabic literature therefore rarely includes diacritics. However, this lack of diacritics does pose serious problems for learners of the language as well as most automated systems such as Automatic Speech Recognition (ASR), Text-to-speech (TTS) systems, and various Natural Language Processing (NLP) applications. Hence the diacritization of raw Arabic text becomes a necessary step for most applications.

Table 1.1 lists the diacritics in Arabic. The three short vowels may appear on any consonant of the word. The three *tanweens*, or *nunation* diacritics are an extended, post-nasalized form of the short vowels, and may appear only on the final letter of a word. The *shadda*, or *gemination* diacritic, may appear in combination with any of the above diacritics. *Gemination* happens when a letter is pronounced longer than usual. This is not the same as *stress*, which is the relative emphasis given to a syllable.

Finally there is the *sukoon*, which indicates that no vowel sound is to be vocalized on the consonant in question, although the sound of the consonant is vocalized.

| **Short vowels** | ◌َ | ◌ِ | ◌ُ |
|---|---|---|---|
| *<Pronunciation>* | */a/* | */i/* | */u/* |
| **Nunation** | ◌ً | ◌ٍ | ◌ٌ |
| *<Pronunciation>* | */an/* | */in/* | */un/* |
| **Syllabifacation** | ◌ّ | ◌ْ | |
| *<Name>* | *Shadda* | *Sukoon* | |

Table 1.1. List of Modern Standard Arabic (MSA) diacritics.

### 1.1.1 Two Sub-Problems: Lexemic and Inflectional

Restoring diacritics to text (*diacritization*) can be divided into two sub-problems. **Lexemic diacritization** disambiguates the various lexemes that may arise when a single sequence of consonants is marked with different combinations of diacritics. An example of lexemic diacritization is presented in Table 1.2. **Inflectional diacritization** disambiguates the different syntactic roles that a specific given lexeme may assume in a sentence, and is typically expressed on the final consonant of a word. Inflectional diacritics are also known as *case endings*.

| Word | Diacritized | Pronunciation | Meaning |
|---|---|---|---|
| كتب<br>consonants<br>***k t b***<br>without diacritics | كَتَبَ | */kataba/* | he wrote |
| | كَتَّبَ | */kattaba/* | he made someone write |
| | كُتُبٌ | */kutubun/* | books |

Table 1.2. Three of several valid diacritizations of the Arabic consonants that represent /k/, /t/ and /b/

Considering the last meaning above ("books"), different inflectional diacritics applied on the final consonant of the word could represent different syntactic roles of the "books" in the sentence, such as whether they are the subject of a verb, or an object, and so on. Inflectional diacritization is a complex grammatical problem that requires deeper syntactic and linguistic information of Arabic [1]. The literature on Arabic diacritization therefore reports two different sets of experimental results: error rates that include the error of predicting the case endings, and error rates that do not. Error rates that do not include them are naturally lower.

However, case endings are important for accurate interpretation of texts and for serious learners of the language. They are particularly necessary for scholarly texts that employ more linguistic knowledge than the average colloquial verbiage. The significance of inflectional diacritization is highlighted in Figure 1.1 below. A single wrongly predicted case ending has the capacity to completely reverse semantics. This risk is increased by the flexibility of Arabic syntax. For example, while a valid verbal phrase in English in arranged in the following order: SVO (Subject-Verb-Object), any of the following arrangements would be valid in Arabic: VSO, SVO, VOS, SOV.

| | |
|---|---|
| The boy ate the lamb | أَكَلَ الوَلَدُ الخُرُوفَ |
| The lamb ate the boy | أَكَلَ الوَلَدَ الخُرُوفُ |

Figure 1.1. The Arabic sentences corresponding to the English are both identical except for the diacritics on the underlined letters. In the first sentence, the arrangment is VSO, in the second it is VOS. This has been done by simply switching the inflectional diacritics on the subject and the object.

Existing studies use a variety of approaches to deal with the problem of diacritization. The problem has been approached as an end in itself, as a sub-task of ASR [2], or as a by-product of another NLP task such as morphological analysis or Part-of-Speech (POS) tagging [3]. It has been tackled using Support Vector Machines (SVMs) [4], Conditional Random Fields (CRFs) [5], a Maximum Entropy framework [6], Hidden Markov Models (HMMs) [7, 8] and weighted finite state transducers [9]. However, inflectional diacritization has remained less accurate than the rest of the text, as Habash [1] asserts that it is a complex problem.

## 1.2 Research Objectives and Contributions

Automated methods that use textually-extracted features have not yet solved the inflectional diacritization problem, but the human mind is certainly capable of inferring the right pronunciation. This thesis employs human intuition via speech data in an attempt to improve Arabic diacritization in general and inflectional diacritization in particular.

The only previous coverage of speech in the field of diacritization has been in the context of other objectives, such as Automatic Speech Recognition. This thesis explores speech based on its own merits for diacritization. The claim is that acoustic information should be able to complement and correct existing textual methods. This thesis will investigate this claim and attempt to explore the extent to which acoustic information aids the textual process.

The claim uses the fact that diacritization using textually-extracted linguistic features, such as POS and gender, generates linguistically-informed errors, especially in inflection; while diacritization using acoustic information generates a different class of errors that are based on features extracted from speech, such as energy and pitch. The errors generated using acoustic information should be more consistent across both lexemic and inflectional diacritization, since acoustic features do not differentiate between regular diacritics and case endings.

To explore the above claim, we use a novel approach to diacritization that combines linguistically-insensitive speech data with text-based NLP methods. Our results demonstrate that speech could in fact be used as an important component in the process.

We cover four main areas in this thesis.

Firstly, two independent diacritization systems are built – one model based on acoustic information and the other on textually extracted linguistic information. The acoustic system models speech as HMMs. The text-based system is modelled using CRFs.

Secondly, weighted interpolations of the above systems' results are explored, to arrive at an optimal combination of speech and text in a single diacritization system. We describe the process of combining the two mediums to predict diacritics.

Thirdly, the combined system will be used to evaluate two established tools. Some of the most accurate research work in the field has relied on the following tools for diacritization and text-based feature extraction: MADA (Morphological Analysis and Disambiguation for Arabic) and BAMA (Buckwalter Arabic Morphological Analyzer). These two resources will be compared in light of the combined diacritization method presented.

Finally, we focus on text-based diacritization. Within the framework of our combined system, we investigate the effects of varying the underlying text-based models: a model that casts the diacritization as a sequence labelling problem using CRFs, and one that uses a classification approach based on SVMs. Aside from the combined system, we then work with text-based diacritization to study the effects on case endings of textually-extracted features at the phrase-, word- and morpheme-levels.

Our proposed system could be useful in various multimodal applications in Arabic, particularly for language learners, such as the simultaneous production of audio-books and fully diacritized text books. The current publication of Arabic books is either without diacritics or with often incorrectly diacritized texts. The long term objective is to bridge the gap between non-natives and complex written Arabic in an educational environment, and this work is a step towards that objective.

## 1.3 Outline

The rest of this thesis is organized as follows.

Chapter 2 reviews existing work done on Arabic diacritization. The work is divided into those studies related to purely text-based diacritization and those that include acoustic information.

Chapter 3 briefly visits the subject of Arabic orthography, as it relates to the process of diacritization in this thesis.

Chapter 4 covers the theoretical framework of text-based diacritization. Special attention is given to SVMs and CRFs, as the two text models focused on in this thesis. The underlying functionality of BAMA and MADA are also described, as they relate to the system proposed in this thesis and are widely used by other studies mentioned in the literature review.

Chapter 5 gives an overview of speech-based diacritization: the features of speech, HMMs, and the components and algorithm of the diacritizer proposed in this thesis.

Chapter 6 proposes diacritization as a weighted combination of speech- and text-based methods.

Chapter 7 describes the datasets and the data processing used in building and experimenting with the combined diacritizer. The orthographic rules in Chapter 3 are applied as they relate to each mode of diacritization. The individual steps and components of the text-based and speech-based diacritizers are covered in detail, followed by the processing required in the combination of speech and text.

In Chapter 8, the speech-based system's results are combined with the optimal text-based system's results. Different weighted interpolations are evaluated. Two features are varied and compared in the combination: (1) The base solutions that are used to constrain the system’s predictions; and (2) the text model’s framework is switched from CRFs to SVMs. N-best lists are also processed and evaluated.

Chapter 9 describes the Text-based diacritizer and related experiments. It covers the extraction of text-based features at the morpheme-, word- and phrase-levels, and evaluates their effectiveness in light of diacritization.

Finally, Chapter 10 concludes the thesis and proposes future directions.

## 2. State-of-the-art Diacritization Systems

Automatic diacritization appears under a few different names in the literature – it is sometimes known as *vocalization*, *romanization*, *vowelization*, or the *restoration* or *prediction* of *vowels* or *diacritics*. For consistency in this thesis, we will refer to the subject as *diacritization*, or the *prediction of diacritics*. The diacritization problem has traditionally been approached from an exclusively text-based point of view. This is understandable, since speech technologies, especially in Arabic, have not yet reached the same level of maturity as text-related fields. However, the advent of the multimodal user interfaces (MUI) industry and projects such as *Global Autonomous Language Exploitation* (GALE)[2] have pushed the development of Arabic speech recognition systems into the limelight. Since diacritics are necessary to disambiguate and realize the pronunciation of text, most work that incorporates acoustic information into the process has been primarily geared toward ASR. There has been little to no work in diacritization on studying the effects of acoustic information for its own merits.

We begin the literature review with an overview of text-based systems, and then cover studies that include the use of speech.

### 2.1 Text-based

Automatic diacritization was initially taken as a machine translation (MT) problem, and solved using rule-based approaches, such as in the work of El-Sadany and Hashish [10], at IBM Egypt. Their system comprised a dictionary, a grammar module and an analyzer/generator. The grammar module consisted of several rules including morphophonemic and morphographemic rules. While rule-based systems have their advantages, such as simple testing and debugging, they are limited in their ability to

[2] http://www.darpa.mil/Our_Work/I2O/Programs/Global_Autonomous_Language_Exploitation_%28GALE%29.aspx

deal with inconsistencies, noise or modification, without considerable human effort. Moreover, Arabic is both a highly *agglutinative* as well as a *generative* language. Its morphological structure allows for new words to be coined whenever needed, which instantly give birth to a towering scale of additional new words, formed from numerous valid combinations and permutations of existing roots and morphemes. Therefore, rule-based methods are only sustainable for a limited domain.

More recent approaches have used statistical methods instead.

In the statistical machine translation (SMT) [11] approach, a source language is translated into a target language with the help of statistical models such as language models and word counts. A Language Model (LM) is a statistical method of describing a language, and is used to predict the most likely word after a given string or sequence of strings, using conditional probability. An example-based machine-translation (EBMT) approach was proposed by Eman and Fisher [12], in their development of a TTS system. The diacritization module was based on a hierarchical search at four different levels: sentence, phrase, word and character. Beginning at the sentence level, it searched for diacritized examples in the training data that could fit the given sentence. If not found, it broke the sentence down into phrases and searched for fitting diacritized phrases from the example set. If not found, it broke the phrases down into words, and then if needed into letters. It used n-grams (explained in Section 4.2) to statistically select an appropriate example for the given input. Schlippe [5] used a similar approach, operating first at the level of entire phrases, then at word-level and then at character-level, and finally using a combination of word and character-level translation, so that the system could derive benefit from both levels.

As opposed to the above methods, the majority of approaches to diacritization have viewed the problem as a sequence labelling task.

Gal [7] and El-Shafei [8] modelled the problem using HMMs (described in detail in Section 5.2). In this approach, un-diacritized words were taken to be *observations*, while their diacritized solutions were taken to be the *hidden states* that produced those observations. Viterbi, a probabilistic dynamic programming algorithm, was used to find the best hidden states. Gal achieved a Word Error Rate (WER) of 14%. El-Shafei achieved a Diacritic Error Rate (DER) of 4.1%, which was reduced to 2.5% by including a pre-processing stage that used trigrams to predict the most frequent words. Both studies were evaluated with the inclusion of case endings. But they were trained and tested on the text of the Quran alone, which is finite and unchanging. For Gale [7], this is understandable, since the Quran is the most accessible fully diacritized text, and very few other annotated resources were available at the time.

However from 2003, the Linguistic Data Consortium (LDC) of the University of Pennsylvania began to publish large corpora of MSA text[3], including Arabic Gigaword[4] and the Penn Arabic Treebank. These corpora are fully annotated with diacritics, POS tags and other features, and have addressed the problem of the scarcity of training material for supervised learning approaches.

Nelken and Shieber [9] made use of these corpora in their approach to diacritization. They built three LMs: word, character and simplistic morphemes (or *clitics*). Nelken and Shieber employed these LMs in a cascade of finite state transducers (FSTs) [13] – machines that transition input into output using a transition function. The FSTs relied on the three LMs for making transitions. The first FST used the word LM to convert a given un-diacritized text into the most likely sequence of diacritized words that must have produced it. Words that could not be diacritized by the word-based LM were then decomposed by the second FST, which used the letter LM to break

[3] http://www.ldc.upenn.edu/

[4] http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2009T30

words down into letters. The rest of the FSTs were used for decomposing spellings and clitics. A simple illustration of their system is presented in Figure 2.

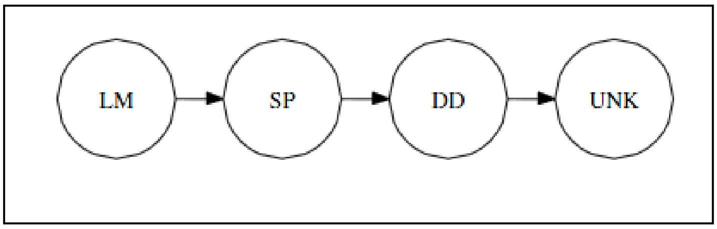


Figure 2.1. Cascading weighted FSTs

One shortcoming of this method was the independence of the FSTs. For example, the Clitic FST could not refer to the Word FST. This created problems for clitic diacritics that depended on the preceding letter in the word. However, the approach using weighted FST was one of the first that decomposed words into morphemes and diacritized at that level. This is distinct from decomposing words into letters, which are not meaningful sub-units of words. The Weighted FSTs system generated a WER of 23.61% and a DER of 12.79% when case endings were included. Without case endings, the results were 7.33% and 6.35% respectively.

Zitouni et al. [6] achieved a higher accuracy when they viewed each word as being a sequence of characters *X,* whose each character was classified by a diacritic as its label. The objective was to restore the sequence of labels, *Y*, to the sequence of consonants, *X*. The Maximum Entropy framework (or MaxEnt) [14] was used to solve this problem. The *Principle of Maximum Entropy* states that, without external information, the probability distribution that best explains the current classification of data should be that with the maximum entropy, or uniformity [15]. Given a supervised training environment, where each training example (as a sequence of consonants) was provided with a vector of features, the MaxEnt framework associated a weight with each feature to maximize the likelihood of the data. The classification features that Zitnoui et al used were lexical, segment-based (or morpheme-based) and POS tags. The system performed at 18% WER and 5.5% DER with case endings. Without case endings, the WER and DER were 7.9% and 2.5%. In their paper, Zitouni et al clearly defined their evaluation metrics and suggested a division of training and testing data from the LDC Arabic Treebank. These suggestions have been adopted by subsequent researchers, and will be used throughout the system in this thesis as well.

With a slightly different take on the problem, Nizar and Habash [5] dealt with diacritization via MADA[5], their multi-faceted software for Arabic morphological tagging, diacritization and lemmatization. MADA performs diacritization with the help of the BAMA, since BAMA provides fully diacritized solutions in additional to morphological analyses. The MADA software adds an additional feature layer to enhance BAMA. It employs trigrams trained on the Penn Arabic Treebank, and 14 different taggers based on SVMs (see Section 4.5). Each SVM is trained to classify words according to a different linguistic feature, such as POS, gender, case or number. For a given word, the 14 different taggers' classification decisions are combined into a collective score. Five other features, such as the trigram probabilities from the LM, are added into the calculation. These scores from a total of 19 features are then used to rank the analyses provided by BAMA, and the highest ranking analysis is chosen as the final diacritization solution. MADA has an accuracy of 14.9% WER and 4.8% DER with case endings. Without case endings, it achieves 5.5% WER and 2.2%. The software is regularly updated and has been used by several research groups, including those at MIT, Cambridge University, RWTH in Germany, National Research Center of Canada and others. It is an immensely valuable tool in the field, but its word-based approach to diacritization means it cannot capture inter-word dependencies or syntax, hence its inflectional diacritization suffers.  We will compare the use of the diacritized solutions from MADA versus those from BAMA in Section 8.1.

[5] http://www1.ccls.columbia.edu/MADA/

In 2008, Tim Schlippe [5] compared the traditional sequence labelling approach with Statistical Machine Translation (SMT). His SMT approach is mentioned towards the beginning of this literature review, in which he achieved a 21.5% WER and 4.7% DER with case endings, and 7.4% WER and 1.8% DER without case endings. In his sequence labelling approach, Conditional Random Fields (CRFs) were used to predict the correct sequence of diacritics for a given sequence of un-diacritized consonants. The CRFs were trained on consonants, words and POS tags, and were used to predict the diacritic on each consonant. The best results achieved in this approach were WER 22% and DER 4.7%, inclusive of case endings, and WER 8.3% and DER 1.9% without case endings. Although the CRFs did slightly worse that the SMT approach, Schlippe concluded that additional training data and context would improve the performance of the sequence labelling approach based on CRFs. This thesis is in agreement with that suggestion and further explores diacritization using CRFs.

The most accurate system reported in the literature to-date is the dual-mode Stochastic Hybrid Diacritizer of Rashwan et al. [16]. Like Eman and Fisher [12], Nelken and Shieber [9], and Schlippe's SMT approach, they work on raw Arabic text using a *combination* of different levels of diacritization. They use two levels: fully formed words and morphemes. In the first mode, they diacritize fully formed words using the most likely solution from an A* lattice of diacritized words. When the search returns no results, the system switches to the second mode where the word is *factorized* into its morphemes, and the most likely solution is selected from a lattice of diacritized morphemes. They add a *morphemes syntactic diacritics* LM to assist in this mode – this adds a layer of sophistication to their work that deals with inflectional diacritics. Another important aspect of their work is their unique morphological word structure. Words are built from morphemes in a quadruple structure, where each structure is a composition of its *prefix*, *root*, *pattern*, and *suffix*. This hybrid system produces a 12.5% WER and 3.8% DER with case endings, and 3.1% WER and 1.2% DER without case endings.

While each of the above studies has been a valuable contribution to the subject, at 12.5% word-error rate, this thesis asserts that there is still room for improvement. Also, despite being a distinct problem, inflectional diacritization has typically not been given distinct attention from lexemic diacritization. It is instead covered by methods that deal with the general diacritization of all Arabic text. This is a natural consequence of the fact that predicting case endings in Arabic is a complex problem, as asserted by Habash [1]. The only notable exception to this trend is the work of Rashwan [16].

We now turn our attention to approaches that have taken acoustic information into account.

## 2.2 Speech-based

Kirchoff and Vergyri [2] covered automatic diacritization as a subtask of ASR. The study began with an acoustic model that incorporated morphological and contextual information, and ended with a purely acoustically based system that did not use additional linguistic information. Like Habash and Rambow [3], a tagger was used to score the diacritized solutions of a word from a list of solutions provided by BAMA, in order to incorporate textually-extracted linguistic features such POS. However, the tagger was trained in an unsupervised way, using Expectation Maximization (EM) [17], to converge at the most likely solutions from BAMA. The probability scores assigned by the tagger were used as transition weights for a word-pronunciation network. Acoustic models trained on CallHome[6] data were then constrained by these pronunciation networks to ensure that they selected only valid sequences of diacritized words. The WER and character error rate achieved in this way were 27.3% and 11.5%, with case endings. It was demonstrated that textually-extracted linguistic information was required to achieve reasonable diacritization accuracy, since acoustic models without linguistically constrained word-pronunciation networks generated less than satisfactory results: word and character error rates in that case were as high as 50% and 23.08%. Kirchoff and Vergyri did not model the gemination diacritic (*shadda*).

Ananthakrishnan et al. [18] selected solutions generated by BAMA with the help of maximum likelihood estimated over word and character n-grams. They reported a WER of 13.5%, with case endings included. As future work for ASR, they proposed an iterative process where the solution generated by the text-based diacritizer above is used to constrain the pronunciation network for an incoming acoustic signal. The

[6] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC97S45

recognizer's output from this process may be used to iteratively re-train the existing acoustic model for ASR.

While there are a large number of studies in Arabic speech technologies, to our knowledge the above two studies are the only notable published contributions that discuss *diacritization* in some depth. Since their objective is ASR, they are not concerned with the accuracy of inflectional diacritization. Outside of ASR, Habash and Rambow [3] only proposed the question of introducing acoustic information into their method.

To briefly conclude the speech-based section of this literature review, we reiterate the results of Kirchoff and Vergyri [2], which demonstrated that textually-extracted linguistic information is required to improve speech-based diacritization. Subsequent work in the industry focused on purely text-based methods that achieved significantly higher accuracy, which we covered in 2.1.

This thesis will depart from the above trend, taking essentially the opposite view from Kirchoff and Vergyri. It will explore the use of speech-based diacritization to complement and correct the results generated by a text-based system, with special attention to case endings.

# 3. Arabic Orthography

This chapter briefly covers the spelling system, or *orthography*, of Arabic. While there are numerous rules and classes of rules [19], we will cover only the major ones as they relate to both text-based and speech-based diacritization in this thesis.

The rules are listed below. We use the Buckwalter encoding to spell Arabic words. The Buckwalter code is a one-to-one mapping from Arabic graphemes to 7-bit ASCII characters. (See Appendix A for the mapping.)

## 3.1 Definite Article "Al"

Definiteness is determined by the *Al* article, which appears before a word, similar to how the word "the" in English precedes a word to mark its definiteness. However, *Al* is a proclitic which becomes a part of the orthographic form of the word it defines. The Arabic consonants are divided into two classes: *solar letters* and *lunar letters*. The following rules apply when a word is prefixed with *Al*:

1. If the word begins with a *solar letter*, then the phonetic "*l*" sound of *Al* is eliminated from pronunciation, although still written, and the solar letter is geminated. For e.g. the consonant *$* (pronounced "*sh*") begins the word *$ams*, (pronounced "*shams*"), which means "sun". To write "the sun", we prefix the word the definite article. Taking the gemination diacritic to be "~" according to the Buckwalter encoding, we also geminate the solar letter:

   ***Al + $ams = Al$~ams***

   The correct pronunciation would be "a***sh***ams".[7]

[7] The underlining has been used to represent gemination, corresponding the ~

2. If the word begins with a lunar letter, then affixation of *Al* is trivial, and the lunar letter retains its phonetic sound during pronunciation.
3. If the word and its definite article are both preceded by a clitic diacritized with *i*, then the *A* of the *Al* is eliminated, both in orthography as well as phonetic pronunciation. e.g.:

***li + Al + kitAb = lilkitAb***

## 3.2 Alif Maksoorah

*Alif* is the first letter of the Arabic alphabet, transliterated as *A*, and is also one of the long vowels. While there are 28 consonants and three short vowels in Arabic, three of the consonants (*A*, *y*, *w*) also function phonetically as long vowels ("*aa*", "*ee*" and "*oo*"). This usually happens when they are diacritized by a *sukoon* or no diacritic, and are preceded by their shorter counterpart. For example, in the word *fiy*, (which means "in"), *y* is preceded by the short vowel *i*, and itself has no diacritic, hence the word is pronounced "*fee*". This is a phonetic realization that does not create challenges in orthography.

However, in the case that *y* is the third letter of a verbal root and is preceded by the short vowel *a*, it is called *Alif Maksoorah* – an *alif* that is represented by *y*. There is a slight different in orthographic form of the regular *y* and *Alif Maksoorah*, which is reflected in Buckwalter encoding:

| Arabic | | Buckwalter | |
|---|---|---|---|
| **Y** | **Alif Maksoorah** | **Y** | **Alif Maksoorah** |
| ي | ى | Y | Y |

Table 3.1. Orthographic difference between *y* and *Alif Maksoorah*.

The only difference between the two forms in Arabic script is the two dots. These dots are sometimes eliminated in writing, by typographical error or simple inconsistency. Perhaps to reflect this, BAMA does not distinguish between words ending with *Y* or *y*. Therefore, given a word *ElY*, the morphological analyses that BAMA produces will include all morphological analyses of the word *ElY*, as well as of the word *ElY*. This adds a challenge to data processing and valid diacritization, while using BAMA.

## 3.3 Taa Marbootah

The *Taa Marbootah* is simply the consonant *t* (known as *Taa*) in Arabic, except in a new orthographic form when it is affixed as a feminine marker at the end of a word. This change creates two problems in diacritization.

1. Like y and *Alif Maksoorah* above, it is confusable with another character, *h*.
2. If a suffix is attached to a word ending in *Taa Marbootah*, the *Taa Marbootah*'s orthographic form transitions back to its original "Taa" form.

| **Taa** | **Taa Marbootah** | **H** |
|---|---|---|
| Arabic | | |
| **ت** | **ة** | **ه** |
| Buckwalter | | |
| **t** | **p** | **h** |

Table 3.2. Orthographic difference between *Taa*, *Taa Marbootah* as a feminine marker, and *h*.

## 3.4 Hamza

*Hamza* is the glottal stop, and has several orthographic forms in written script, depending on the context and on its preceding letter. The following are different representations of *hamza*:

| Buckwalter | > | < | & | ‘ | } |
|---|---|---|---|---|---|
| Arabic | أ | إ | ؤ | ء | ئ |

Table 3.3. Different orthographic forms of *hamza*.

Note that the fifth form on the right is confusable with *Alif Maksoorah* in Table 3.1. This adds to the complexity of analyses that are generated by BAMA. The first two forms are confusable with *alif* (which is *A* in Buckwalter encoding, and ا in Arabic).

While there is no *hamza* in words beginning with *alif*, a glottal stop is often phonetically realized, if the word is pronounced independently from the previous word.

## 3.5 Alif

1. The phonetic pronunciation of the long vowel *alif* may sometimes be varied or elongated. There are different orthographic forms of *alif*, or extended diacritics to represent this form of *A*. They are:

| Buckwalter | ` | \| |
|---|---|---|
| Arabic | ◌ٰ | آ |

Table 3.4. Elongated *Alif* and short vowel variants.

2. In written script, *alif* is added at the end of certain verbs, such as 3rd person male plural verbs in the past tense, which end in *uw*. In this case, the

additional *alif* has no phonetic realization and no diacritic, but is only added in the script.

## 3.6 Diphthongs

There are two diphthongs in MSA, which are produced when a potential long vowel is preceded not by its shorter counterpart, but by a different class of short vowel. For e.g. if the consonant *w* is preceded by its own shorter counterpart *u*, it is transformed into the long vowel of the sound "*oo*". However if it is preceded by the vowel *a*, it produces a diphthong. The two diphthongs are created as follows:

1. *a* + **w** = "*aw*"
2. *a* + **y** = "*ay*"

If the consonants w or y have a short vowel diacritic instead of a *sukoon* (no vowel sound), or are followed by *alif* (i.e. "*A*" in Buckwalter encoding), they do *not* form diphthongs, but are pronounced as simply as the consonants *w* and *y* in English.

## 3.7 Tatweel: the Text Elongating Glyph

The *Tatweel* has no semantic use but is used for presentation purposes, such as text justification, to elongate characters.

The table below shows an example of a word with and without *tatweel*. *Tatweel* is represented in Buckwalter transliteration with an underscore.

| | **Without Tatweel** | **With Tatweel** |
|---|---|---|
| Buckwalter | ktAb | k_tAb |
| Arabic | كتاب | كـــــــتاب |

Table 3.5. *Tatweel*

## 3.8 Consonant-vowel Combinations

1. Two consonants with the *sukoon* (no-vowel) diacritic may not be appear adjacent to each other.
2. A vowel may not appear without a consonant.
3. Even in script that is often considered “fully diacritized”, not all consonants receive a diacritic. There are two cases for this:
    a. A consonant is included in the script but not pronounced. An example of this case is the *alif* in Section 3.5 above, which is added at the end of certain word formations.
    b. It is simply missed out by convention in the spelling of some words, and is not necessary for disambiguation.

    Having no diacritic is different from having the *no-vowel*, where the consonant is still pronounced.

This concludes our brief overview of the main orthographic rules handled in this thesis.

# 4. Text-Based Diacritization

We begin this chapter by covering the basic concepts and groundwork required to understand text-based diacritization as used in this thesis.

## 4.1 Text-based Linguistic Features

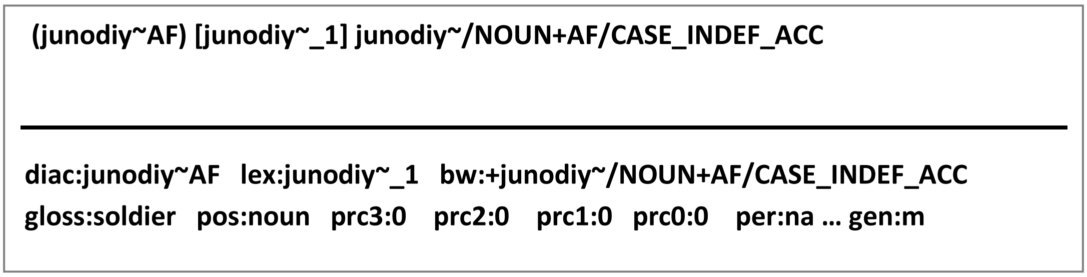
Various linguistic features may be extracted from an annotated text corpus, or using tools such as stemmers, or morphological analyzers such as BAMA. The features include, but are not limited to:

- **POS**. For examples: *noun*, *proper noun*, *verb*, *adjective*, *adverb*, *conjunction*.
- **Gender**. Arabic words have one of two genders: *male* or *female*.
- **Number**. Arabic words have one of three number values: *singular*, *dual* or *plural*.
- **Voice**. *Active* or *Passive*, such as in "he did", or "it was done".
- **Clitic**. This may be used to specify whether a word has a clitic[8] concatenated to it.
- **BPC**. Base Phrase Chunk (BPC) features determine the *base phrase* that a word belongs to in a sentence, for examples a verbal phrase (VP), a nominal phrase (NP), an adverbial phrase (ADVP), and so on.
- **Case**. This feature applies to nouns. It can be *genitive*, *nominative*, or *accusative*.
- **Mood**. This feature applies to verbs. It may be *subjunctive*, *jussive* or *indicative* or *imperative*.
- **Lemma**. The morphological root of a word, excluding affixes.
- **Pattern**. Arabic morphology consists of *patterns*, each pattern carrying its own semantics. Patterns are templates of operations that are applied to a root to transform it to a different form that implies different meanings related to the essence of the root meaning. Operations include adding a specific character at

[8] http://www.sil.org/linguistics/GlossaryOfLinguisticTerms/WhatIsACliticGrammar.htm

the beginning of the root, substituting a vowel, eliminating a character from the final position and so on. There are ten patterns. For example, the word *kataba* means, “he wrote”. Using Pattern II which geminates the middle letter results in what is pronounced as *ka<u>t</u>taba*, which means “he forced someone to write”.

Figure 4.1 illustrates an example of the kind of features generated in the morphological analysis of the same word, using two different tools.



```
(junodiy~AF) [junodiy~_1] junodiy~/NOUN+AF/CASE_INDEF_ACC
```

```
diac:junodiy~AF   lex:junodiy~_1   bw:+junodiy~/NOUN+AF/CASE_INDEF_ACC
gloss:soldier   pos:noun   prc3:0   prc2:0   prc1:0   prc0:0   per:na … gen:m
```

Figure 4.1. Features extracted for the word *jndyAF*. The first analysis shows a lexeme and detailed POS tag. The second shows lexeme, Buckwalter analysis, glossary, simple POS, third, second, first, zeroth proclitic, person and gender.

Other textually extracted features that are used in various NLP tasks, including diacritization, are statistical features such as word counts or probabilities. Such features are often combined to create a language model.

## 4.2 Language Model

A language model is used to describe the statistical characteristics and probabilities of a language for each word within it. An *n-gram* [20] is a type of language model that computes the conditional probability of a word given a sequence of previous words. We briefly review a few concepts in probability theory to explain n-grams.

The simple probability of an event A is the frequency of its occurrence, or its count (denoted by the function *C*), divided by the number of total events, say N:

$$P\,(A) = C\,(A)\,/\,N \tag{1}$$

So if the word “cat” were to appear 5 times in a dataset of 70 words, then P(cat) = 5/70.

*Conditional Probability* defines the probability of an event *A*, on the condition that event *B* has already occurred, as P (A | B):

$$P (A \mid B) = P (A \cap B) / P (B) \tag{2}$$

In finding the probability of a word given a sequence of previous words, the expression is extended to P ($w_i$ | $w_{i-l}$, $w_{i-2}$, … $w_1$). In the case of languages, it is usually impossible to compute the probabilities of all words given all possible sequences, since not every one of them can appear in a language. Therefore an underlying concept of *independence* for many probabilistic models is the *Markov principle*, which states that the probability of an event depends only on the previous event. In this case, rather than finding the probability of a word given sequences of all the possible previous words, we approximate this probability by relying on only one or two previous words. *N-grams* are used to predict the occurrence of a word, say $x_i$, given n-1 preceding words, i.e. P ($w_i$ | $w_{i-(n-1)}$ ,…, $w_{i-1}$). Unigrams (where n=1) assume that the probability of every word is completely independent. The most common n-grams that are used are bigrams and trigrams. Bigrams are n-grams where n=2, so the conditional probability of each word depends on the probability of one previous word (2-1=1). For trigrams, n=3, and the conditional probability of a word depends on two previous words (3-1=2).

Since language models predict only the most likely words, they also function as constraints, keeping the generation or decoding of a language within the confines of valid possibilities.

## 4.3 BAMA

Tim Buckwalter's BAMA [21] was released in 2002, and is used widely for Arabic NLP tasks. It is used for several NLP applications, including diacritization, since the morphological analyses that it generates for an input word are fully diacritized.

BAMA has three components, described below [23]:

1. **Lexicon**. An Arabic word is stored as a concatenation of its stem, and its prefix and suffix, if any. Lexicon entries include all possible stems, prefixes, and suffixes. Each entry is given a *morphological compatibility* category, which ensures that only morphologically cross-compatible combinations may be made. Entries are also encoded with their fully diacritized forms and their *lemma*. (While a *stem* is used to segment a word, such as "unchanging", into "un", "chang", and "ing", a *lemma* works at a more semantic level, so the lemma of "unchanging" is the word "change".)
2. **Compatibility tables**. Three tables are used to list all the categories that may be concatenated with a given category. An example of a compatibility category may be *Pref-wa*, which specifies the prefix conjunction *wa*, (i.e. "and" in English). All morphological categories that may be concatenated with this conjunction will be listed under this category. In the *Pref-wa* category, all noun stem categories are listed.

3. **Analysis engine**. The engine segments an input word into all of the possible combinations of its prefix, stem and suffix. (There may be several valid segmentations possible since the lack of diacritics causes ambiguity). It then fetches all possible matches of the segments in the lexicon, and checks against the compatibility tables to construct valid combinations as output. The output is then a list of *solutions*, where each solution is provided with its diacritized form, its lemma, its morphological analysis and its POS tag. If an

input word is not three-way-compatible with the tables, BAMA returns “NO SOLUTION”.

This thesis uses BAMA version 2.0. Version 3.1 is known as SAMA, and gives a slightly higher accuracy in terms of diacritization, but was not available to us at the time of this research.

## 4.4 ALMORGEANA

MADA is a software suite that morphologically analyzes, disambiguates, lemmatizes and diacritizes Arabic text, in one fell swoop. [22] Almorgeana (ALMOR) is MADA’s morphological analysis and generation system. It is built on top of the databases provided by BAMA [23]. While BAMA operates on the surface form of words, ALMOR adds another layer that deals with the lexeme and feature form, so BAMA entries are extended with two more keys, *lexeme* and *feature*. The analysis engine of ALMOR does not break down words into their prefix, stem and suffix segments, but into their lexeme and feature keys. Compatibility is then matched across the three tables using these keys, and all compositions that match are converted back into their prefix-stem-suffix forms and returned as output.

Since most complexity is hard-coded into the tables, BAMA uses a simple analysis engine to provide all possible solutions to a word. MADA goes a step further and processes its solutions to narrow down the list to the most likely solution. This is done using SVMs.

## 4.5 Support Vector Machines

A basic SVM is a linear, binary, non-probabilistic classifier that maximizes the margin between two classes, i.e. it finds a line (or *decision boundary*) that leaves the greatest distance between the points in each class. Given an input data point, it then assigns the point to one of the two classes, depending on which side of the boundary the data point falls.

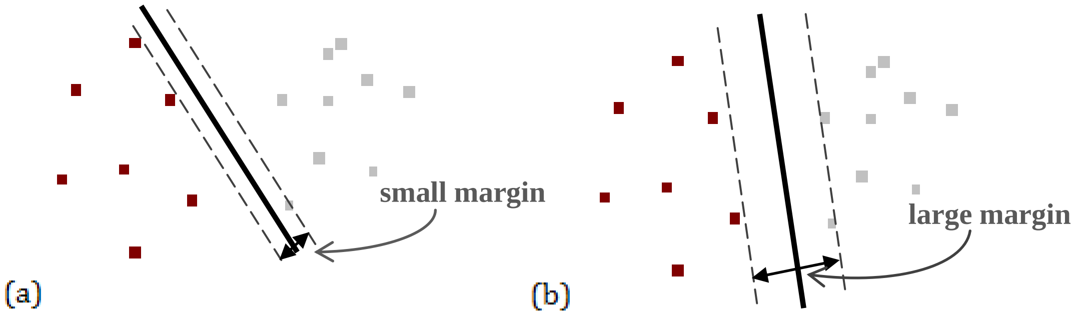


Figure 4.2. Decision boundary lines separating data points with the greatest margin in SVMs.

It is often difficult to find a single line that separates the training data points cleanly, so the input points are mapped onto higher dimensional space *feature space*. Then instead of a simple two-dimensional line, (p-1)-dimensional *hyperplanes* are found that maximize the margin between the classes. Data points are input as p-dimensional vectors, and since training is supervised, the class of each data point is known.

Formally, given training data T, a set of *n* data points, where each point $x_i$ is described by a p-dimensional coordinate vector and a binary variable (-1,1), classifying which class $x_i$ belongs to, T can be described as:

$$T = \{ (x_i, y_i) \mid x_i \in \mathbf{R}^p, y_i \in \{-1,1\}\}^{n}_{i=1} \qquad (3)$$

A hyperplane that divides points with y = 1 from those points with y = -1 can be described as the set of points **x** that satisfy the following equation:

$$\mathbf{w} \, . \, \mathbf{x} - b = 0 \qquad (4)$$

where **w** is the normal vector to the hyperplane, *b* is constant, and **.** denotes the dot product.

**w** and *b* are chosen such that the margin between the two classes is maximized. In other words, the distance between two hyper-planes that mark the boundaries of the classes is maximized. This is done using constraints and quadratic programming, and becomes an optimization problem based on a *kernel function*. Explaining these techniques is outside the scope of this thesis, but they can be referred to in [24].

In the context of diacritization, MADA uses several different SVM classifiers, each trained on a different morphological feature extracted from the Penn Arabic Treebank (ATB3) dataset. [25] Each SVM is trained to classify and predict, for each word, a specific morphological feature. The 14 features are listed below in Table 4.1 [26].

| **Feature Name** | **Explanation** |
|---|---|
| POS | Simple part-of-speech |
| CNJ | Presence of a conjunction clitic |
| PRT | Presence of a particle clitic |
| PRO | Presence of a pronominal clitic |
| DET | Presence of the definite determiner |
| GEN | Gender |
| NUM | Number |
| PER | Person |
| VOX | Voice |
| ASP | Aspect |
| MOD | Mood |
| NUN | Presence of nunation (indefiniteness marker) |
| CON | Construct state (head of a genitive construction) |
| CAS | Case |

Table 4.1. The fourteen features used by MADA to classify words.

Each SVM classifier is assigned a weight via a tuning dataset. For a given word, the classifiers' predictions are collected and compared against the potential morphological analyses provided by ALMOR, and the comparisons are scored. The more the taggers that classify a specific analysis of a word, the higher the score of that analysis. In addition to the 14 SVMs above, five other weighted features are included in the scores, listed in Table 4.2.

| Feature Name | Explanation |
|---|---|
| Spellmatch | Confirms whether the generated analysis, with its diacritics stripped, matches with the undiacritized word that was given as input. |
| Isdefault | Determines whether the generated analysis is what is labelled as "default" by BAMA, i.e. the same as the input. This is less likely to be the correct analysis. |
| UnigramDiac | A unigram language model of the diacritics of full form words. |
| UnigramLex | A unigram of lexemes. |
| N-gram | A 4-gram lexeme model. |

Table 4.2. Five features used to score MADA analyses in addition to the 14 SVM features.

The analysis that has the highest collective score of these 19 features is selected as the final solution.

## 4.6 Conditional Random Fields

As opposed to the binary classification of SVMs, CRFs are a probabilistic framework oriented towards segmenting and labelling sequence data. While some models, such as HMMs (see Section 5.2) model the joint probability between random variables, CRFs have the ability to define *conditional* relationships between observations and labels. This relaxes the independence assumption inherent in many models, such as HMMs, that operate on the Markov principle. Operating outside the Markov principle means that CRFs are powerful at capturing long range dependencies across characters, words and word features. The training is discriminative, during which the conditional likelihood of the classes are maximized.

If we take $X$ to be a sequence of observations, and $Y$ to be its corresponding sequence of labels, then the conditional distribution $p\ (X|Y)$ can be modelled as an undirected graph, where $X$ and $Y$ are disjoint sets of nodes. Formally, Lafferty et al. [28] defined CRFs as follows:

Let $G = (V,E)$ be a graph such that $Y = (Y_v)_{v \in V}$, so that $Y$ is indexed by the vertices of $G$. Then $(X,Y)$ is a conditional random field in case, when conditioned on $X$, the random variables $Y_v$ obey the Markov property with respect to the graph:

$$p\ (Y_v\ \mid X, Y_w, w \neq v\ ) = p\ (Y_v\ \mid X, Y_w, w \sim v\ ) \qquad (5)$$

where $w \sim v$ means that $w$ and $v$ are neighbors in $G$.

An example of a simple chain-structured CRF is illustrated in Figure 4.3 below, where $Y$ is conditioned on $X$.

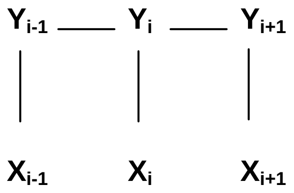


Figure 4.3. Disjoint sets Y and X

In the context of Arabic diacritization, the observed sequence $X$ is the set of consonants given as textual input, and $Y$ is the sequence of diacritics applied to it. Given a feature function $f_i$ that helps determine the likelihood of $Y$, and $X_j$ and $Y_j$ are sub-sequences of $X$ and $Y$, then:

$$\log p\ (X|Y,\ \theta)\ = \sum_{i} \theta_i f_i\ (X_j, Y_j) \qquad (6)$$

where $\theta_i$ is a numerical weight that is assigned to $f_i$. A function $f_i$ over $X_i$ and $Y_i$ could be a fixed Boolean function, which returns *true* if, for example, $X$ was a word-final consonant and $Y$ was a *nunation* case ending. The weights $\theta_i$, or parameters, need to be estimated during training, so they are maximized over the training data $T$, using the following equation:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{(X,Y)\in T} \log p\ (Y \mid X, \theta) \qquad (7)$$

Testing consists of decoding the sequence of observations X into the most likely sequence of labels, $Y^*$, given the parameters $\theta^*$ found above:

$$Y^* = \underset{Y}{\operatorname{argmax}}\ p\ (X \mid Y, \theta^*) \qquad (8)$$

Figure 6 shows the CRFs for an Arabic word.

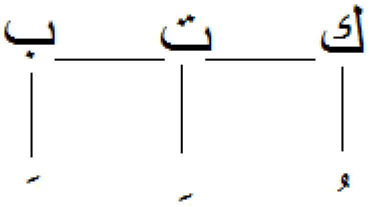


Figure 4.4. CRFs for the sequence of consonants and the sequence of diacritics

CRFs can learn various features and varying contexts to model the training data and predict the set of diacritics, *Y**. Features may include POS, case, and clitic information. Contexts refer to current, previous and next consonants, POS tags, diacritics or other features.

The marginal probability of a single label may also be found, using the formula:

$$p_\theta(Y_j = y \mid X\ ) = \frac{\alpha_j(y|X)\beta_j(y|X)}{Z_\theta(X)} \tag{9}$$

where $Z_\theta(X)$ is a normalization function of the observance of *X*, and α and β are the forward and backward probabilities used in *forward-backward* algorithms. *Forward-backward* algorithms are a class of dynamic programming inference algorithms that compute a set of probabilities going forward in one direction from one set of states to another, and another set of probabilities while going backward. To refer to the forward-backward inference used by CRFs, refer to [27].

With the necessary background covered, we now discuss the text-based diacritizer built in this thesis.

## 4.7 Components of a Text-based Diacritizer

The text-based diacritizer consists of the following five components:

1. **Text Normalizer**. The input stream consists of raw text, sometimes partially diacritized. If the input is in Arabic, it needs to be converted to Buckwalter encoding. Once it is in Buckwalter, a number of normalizing steps are needed to ensure orthographic consistency (See Chapter 3). Text normalization is discussed in Data Processing (Chapter 7), where spellings are converted to a string of consonants that can each receive a diacritic. An example of input text ready for diacritization is shown in Figure 4.5.

```
w*krt An Aby rA$d lA yntmy AlY Ay tnZym Aw Hzb.
wr>t An nql syArth AlY brytAl fy AlbqAE kAn bqSd Altmwyh.
wfhm Anh yjry AlEml ElY Edd mn AlAtSAlAt AlAxyrp…
…
```

Figure 4.5. Raw text in Buckwalter encoding.

2. **Feature Extractor**. Depending on the features used to predict diacritics, the feature extraction layer may employ different tools. In this thesis, the following features were required prior to predicting the diacritics on the test data:

    1. Consonants. Each consonant from a word is taken as a feature.

    2. Words. Words are taken as independent features.

    3. POS tags. These could be extracted using the BAMA engine and the MADA SVM-based analyzer.

    4. Base Phrase Chunk (BPC) tags. These were extracted using non-recursive SVM-based base phrase chunking.

    For the main diacritizer in this thesis, only POS were required. The BPC tags were used in experiments, described in Chapter 8.

After feature extraction, the annotated text is converted to input for the CRFs model. This is described in Section 7.3 Processing: Text-Based Diacritizer.

3. **Text Model**. The CRFs that model the training data use a character bigram for the labels, *V*. The features used in this thesis are:

   1. current consonant

   2-4. The current, previous and next word

   5-7. The POS of the current, previous and next word.

   In addition, the *context* of learning is +-4, i.e. it spans a window of 9 consonants. In other words, if the current consonant is $c_i$, then the following 9 consonants are also learnt:

   $$c_{i-4}, c_{i-3}, c_{i-2}, c_{i-1}, c_i, c_{i+1}, c_{i+2}, c_{i+3}, c_{i+4}$$

4. **Language Model**. Although the text-based diacritizer does not use a regular statistical language model, it relies on morphological analyses (generated by BAMA) as a look-up dictionary to restrict the predictions of the diacritizer to valid words only.

5. **Lexical Scorer**. Each of the candidate solutions is assigned a score, and the highest scoring analysis is selected as the diacritized solution. The scoring is explained in the next section.

## 4.8 Algorithm for Text-based Diacritization

Given a set $D_w$ that consists of $n$ possible diacritized solutions for a word $w$, we are interested in selecting the best solution. We may do this by scoring each solution $d_{w,I} \in D_w$, and choosing the highest scorer.

The text model will be used to assign the scores, producing a set of n tuples, $T_{w:}$

$$T_w = \{ x \mid x = (d_{w,i}, t_{w,i}),\ d_{w,i} \in D_w,\ w \in W \} \tag{10}$$

We will reference the score of a selected tuple in $T_w$ as $T_w[d_{w,i}]$. The scores are assigned through the following process:

For each input word $w$, the diacritizer takes it in as a sequence of raw Arabic consonants, $C$. Each consonant $c$ in this sequence may be assigned fifteen possible labels: the three individual short vowels, the three *tanweens*, the *sukoon*, combinations of the gemination diacritic with the short vowels and *tanweens*, and *no diacritic* (represented as epsilon). Let $V$ be this set of fifteen labels:

V={ ◌َ, ◌ِ, ◌ُ, ◌ْ, ◌ّ, ◌ً, ◌ٍ, ◌ٌ, ◌َّ, ◌ِّ, ◌ُّ, ◌ًّ, ◌ٍّ, ◌ٌّ, ε }

The dotted circles are placeholders for consonants.

The diacritizer then operates as follows:

1. Using formula (9) in (Section 4.6) above, the CRF model is used to compute the marginal probability $p_v$ of each diacritic possibility $v \in V$, for each consonant $c \in C$. In this case, $C$ is the sequence of observations, and $V$ is the sequence of labels:

$$p_\theta(v \mid C) = \frac{\alpha_j(v|C)\beta_j(v|C)}{Z_\theta(C)} \tag{11}$$

2. Given our set $D_w$ of possible solutions for $w$, we calculate the score of a single solution $d_{w,i} \in D_w$ as the sum of the log of $p_v$:

$$T_w\ [d_{w,i}] = \sum_{v \in V} log\ x\ ,\ x = \begin{cases} p_v, & p_v > 0 \\ 0.00001, & p_v = 0 \end{cases} \tag{12}$$

   We choose a low value, *0.00001* for $p_v = 0$ in place of *log 0*.

3. The best diacritization solution for this word, $d^t{}_w*$, is then simply the one that maximizes this sum.

$$d^t{}_w* = \underset{d_{w,i} \in D_w}{\text{argmax}}\ T_w[d_{w,i}] \tag{13}$$

If the input string *W* consists of *M* words, then the diacritizer produces *M* sets of scored tuples.

# 5. Speech-Based Diacritization

## 5.1 Speech-based Acoustic Features

Speech attributes – such as energy and pitch – are used to compute and extract features from speech waveforms. A number of useful features may be extracted, such as Linear Prediction Coefficients (LPC), Perceptual Linear Prediction Coefficients (PLP), or Mel Frequency Cepstral Coefficients (MFCC). MFCCs are used widely in speech technologies and have been chosen for the acoustic model in this thesis. MFCCs are derived from the *Mel scale*. This is a scale of pitches perceived by humans to be equal in distance from one another, although they may not be equidistant in the true frequency scale of sound. Mel frequency can be derived from true frequency *f* using the following calculation [28]:

$$f_{mel} = 2595 \text{ x } \log_{10}(1 + f/700) \quad (14)$$

*Filter bank coefficients* are used to represent the spectral magnitude of waveforms in a compact way, where each coefficient represents the sum of data points that pass a *band-pass filter* (a wave filter that accepts frequencies within a given range).

MFCCs are computed by performing truncated Discrete Cosine Transforms (DCT) on the logs of the Mel filter bank coefficients [29].

Speech waveforms can then be represented using feature vectors of MFCCs.

In addition to the MFCCs, the following are features that may be extracted and that are used in this thesis:

- **Energy term**. The energy is computed as the log of the speech signal energy:

$$E = log \sum_{n=1}^{N} s_n^2 \quad (15)$$

where $s_n$ is one out of $N$ speech signals [30].

- **Zero mean**. Utterance-based normalization is applied on the spectral mean.
- **Time derivatives**. Describing speech in relation to its correlations with time greatly enhances speech recognition. The *delta* coefficients of time derivatives are computed as follows:

$$d_t = \frac{\sum_{\theta=1}^{\Theta} \theta(c_{t+\theta} - c_{t-\theta})}{2\sum_{\theta=1}^{\Theta} \theta^2} \tag{16}$$

where $c_{t\text{-}\Theta}$ to $c_{t+\Theta}$.are corresponding static coefficients, and Θ is the parameter window length. [30]

The above features may form the feature vectors that are needed to build an acoustic model for the representation and characterization of speech. The most widely used statistical models used for this purpose are Hidden Markov Models (HMMs).

## 5.2 Hidden Markov Models

HMMs are sequences of finite-state transducers that accept feature vectors of speech and transduce them into sequences of states that model units of sound. Units of sound may be phones, words, or syllables, or whatever is suitable for the application. The basic underlying model of HMMs is a statistical Markov model, where all states and *transition probabilities* are visible, or *observed* [31]. A HMM adds an additional *hidden* layer of states, which is said to cause the *observed* states, based on *emission probabilities*. In an acoustic model for speech, the observed states represent the speech feature vectors, while the hidden states represent the units of sound, such as individual phone utterances, which caused the observed states. Figure 5.1 illustrates an example of HMMs.

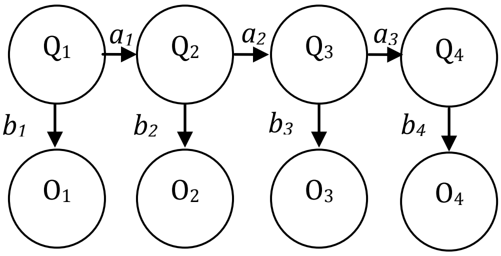


Figure 5.1. HMMs. Transition and emission probabilities are denoted by $a$ and $b$ respectively.

A HMM $\theta$ can be described as follows:

- $Q$ = the set of HMM states

- $\mathrm{O}$ = the set of observed states

- $A$ = set of transition probabilities

- $B(\mathrm{o})$ = set of emission probabilities

We refer to a state $o \in O$ at time *T*, as $o_t \in O^T$, and likewise $q_t \in Q^T$.

The Baum-Welch, Viterbi and Forward Recursion algorithms are commonly used in ASR operations on HMMs. In this thesis, we are concerned with the following operations:

1. **Estimation**. Baum-Welch [32] [43], an Expectation Maximization algorithm, is used to estimate the parameters of the HMMs, given the observed state sequences. This is the training of the acoustic model. The maximum likelihood estimation of the parameters is achieved by maximizing the log likelihood function [33]:

$$L(\theta) = \log p\ (O^T{}_1 \mid \theta\ ) = \log \sum_{Q_1{}^{T+1}} p\ (O^T{}_1, Q_0{}^{T+1} \mid \theta) \tag{17}$$

2. **Evaluation:** In this process, the best alignment between the HMMs and the speech feature vectors is computed. This is done using the Viterbi dynamic programming algorithm. Evaluation is expressed as follows [34]:

$$p\ (O^T{}_1|Q^T{}_1, \theta) = \prod\nolimits^T{}_{t=1} p\ (o_t, q_t, \theta) \tag{18}$$

When alignment is unknown, the *expected* likelihood is computed instead, which makes use of the transition probabilities $A$, and the forward-backward algorithm. (For details, refer to [31], [35]). *Forced alignment* may also be used, where output labels constrain the evaluation algorithm so that it simply computes the likelihood of a given sequence of labels.

## 5.3 Components of a Speech-Based Diacritizer

The speech-based diacritizer in this thesis comprises the following components:

1 **Feature Extractor**. The features extracted from speech in this system are MFCCs energy, utterance-based mean normalization, and the first, second and third time derivatives.

2 **Acoustic model**. The acoustic model is built on HMMs. Each HMM models the pronunciation of a single phone, which may be either a consonant or a vowel. However, we know that most Arabic transcriptions do not include vowels, although they are necessary for realizing word pronunciations. The question then arises: how are they modelled? There are two types of systems that solve this problem in ASR [36]:

   a. *Phonetic system*. In this case, diacritics are explicitly modelled in the dictionary. This requires a lexicon to be ready before speech recognition. However, the generative power of Arabic morphology, its agglutinative nature and the ambiguity produced by an *abjad* system makes such an option problematic – the dictionary grows very large very quickly. Therefore a phone-level dictionary is preferred over a word-level dictionary.

   b. *Graphemic system*. Diacritics are implicitly learned in the acoustic model. This is the system employed in this thesis. Graphemes in the orthographic representation of a word are mapped to phonetic transcriptions via a grapheme-to-phoneme (G2P) processing layer.

The acoustic model then uses a phone-level multi-pronunciation dictionary to predict implicitly modelled diacritics such as short-vowels.

3 **Grapheme-to-phoneme (G2P) layer**. Arabic orthographic rules are applied to convert word graphemes into phonetic transcriptions. The G2P layer used in this thesis is rule-based. The underlying rules that this G2P layer will follow have been covered in Chapter 3.

4 **Language model**. In ASR systems, a language model is often represented by a *grammar structure* or *word network*, which is used to constrain the recognizer to generate sequences of words within the valid confines of a language. The language model assigns probabilities to sequences of words, with the help of a probability distribution of the n-grams found in the training data. Hence a word that appears often will have a higher probability of being predicted than other words. In this speech-based diacritizer, the language model comprises the solutions proposed by BAMA, i.e. the set $D_w$ for each word $w$. The language model scores are not assigned using n-grams but using an acoustic scorer.

5 **Acoustic Scorer**. Using forced alignment, the scorer assigns acoustic log-likelihoods to the elements of set of the diacritized possibilities, $D_w$, in the language model.

The implementation of the above components is described in detail in Chapter 7, Data Processing.

## 5.4 Algorithm for Speech-based diacritization

Given speech input with its features extracted, then for each un-diacritized input word $w$, speech-based diacritization is a four-step process:

(1) **Obtain the possible diacritized solutions.** The set $D_w$ is obtained by taking the unique solutions provided by BAMA.

(2) **Convert solutions to phonetic transcriptions.** Each of the diacritized solutions $d_{w,i} \in D_w$ provided by BAMA are converted through the G2P layer to their phonetic transcriptions.

(3) **Force alignment to extraction word boundaries from speech data**. The acoustic model is used to apply a forced alignment (employing the Viterbi algorithm) between the speech input and the phonetic transcriptions obtained from the G2P layer above. This is done in order to identify *w* in the speech input. Given an un-diacritized word *w*, its corresponding utterance is identified in the speech input by extracting its word boundaries in time, generated during alignment.

(4) **Scoring: Estimating maximum log likelihood**. The word boundaries from Step (3) are used to identify words in the speech input and force align them with each phonetic transcription obtained in Step (2). The acoustic model assigns the maximum log likelihood score to each solution during forced alignment, generating tuples of the form $(d_{w,i}, s_{w,i})$. The acoustic log likelihood is:

$$p\,(O^{T}{}_{1} \mid \theta) = max\ \ p\,(O^{T}{}_{1} \mid Q^{T}{}_{1}\,,\theta) \qquad (19)$$

The acoustic scores are normalized for the utterance, and the most likely solution, $d^{s}{}_{w}*$, selected from $D_w$ is then simply the solution with the maximum log likelihood:

$$d^{s}{}_{w}* = \underset{d_{w,i} \in D}{\operatorname{argmax}}\,[S_w] \qquad (20)$$

We now have an acoustically-scored set of diacritized solutions, $S_w$:

$$S_w = \{\, x \mid x = (d_{w,i}, s_{w,i}),\ d_{w,i} \in D_w\,,\ w \in W\,\} \qquad (21)$$

The process is repeated for each word $w \in W$. Speech-based diacritization for each word *w* is summarized in Figure 5.2 below.

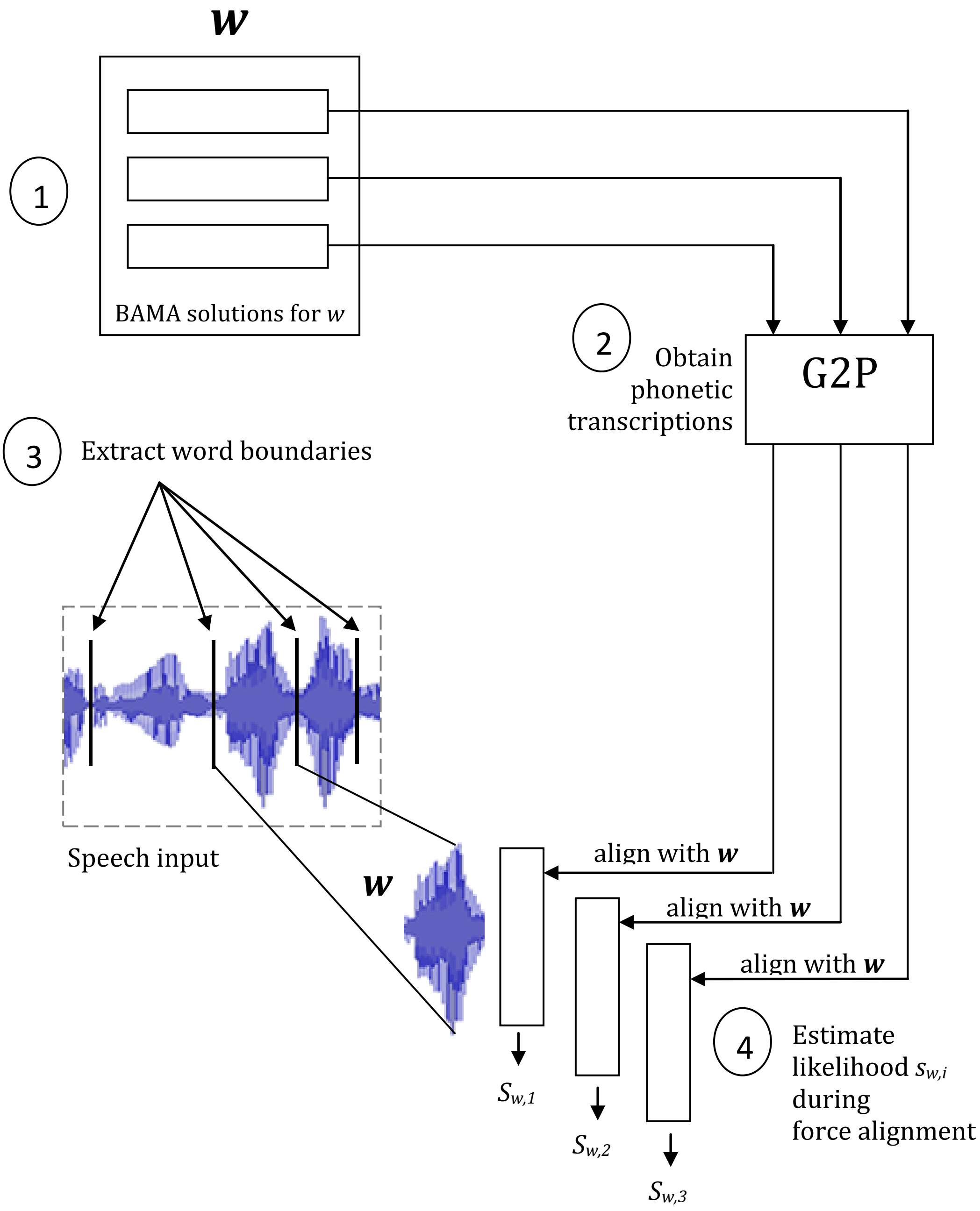


Figure 5.2. Obtaining acoustic scores for combined diacritization using the speech-based diacritizer.

# 6. Combined Diacritization

This thesis takes the human ability to infer the correct pronunciation of a text, and interpolates it via acoustic data with a text-based diacritizer. Having understood the individual diacritizers, we now turn to the question of their interpolation. We are interested in how well their strengths combine to produce a multi-modal diacritizer. We begin with a theoretical overview of the combination.

## 6.1 Overview

The proposed system accepts two streams of data as input: (1) raw Arabic text, and (2) an acoustic signal of correctly vocalized speech corresponding to that text. The objective is to predict the correct diacritics for each consonant in the text.

The text is encoded in Buckwalter transliteration [37]. An example of Buckwalter transliterated text is shown in Figure 6.1, along with its correctly diacritized version below. The diacritics have been printed in bold. (See Appendix A for a complete mapping of Arabic graphemes to Buckwalter).

ATlE AlnA}b AlEAm Altmyyzy EdnAn EDwm

A**i**T~al**a**E**a** Aln**~**A}**i**b**u** AlEAm**~u** Alt**~a**m**o**y**i**yz**i**y**~u** E**a**donAn E**a**D**~u**wm

Figure 6.1. Diacritized and undiacritized Buckwalter transliterated text.

The correct pronunciation of the text is inferred by a human and recorded as an acoustic signal.

Two independent diacritizers are employed: (1) one that is based on text input and modeled by Conditional Random Fields (CRFs); and (2) one that is based on speech input and modeled by Hidden Markov Models (HMMs).

Figure 6.2 summarizes the system's architecture.

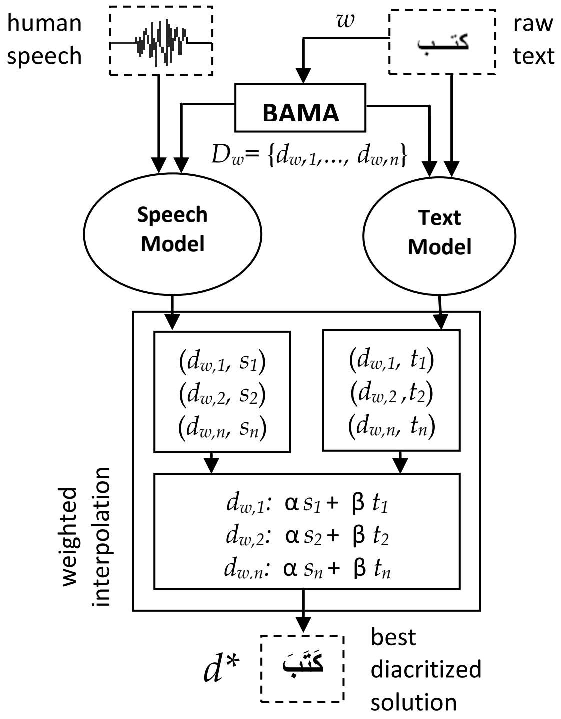


Figure 6.2. Combined diacritization architecture.

We consult BAMA to produce a list of morphological analyses with fully diacritized solutions for a given word.

Let *W* be the sequence of input words. Then for each word $w \in W$, BAMA generates a set of potential diacritized solutions, $D_w$. Since our task is diacritization, we are only concerned with the unique sequence of diacritics on a string of consonants, and not with its morphological analyses. Therefore if BAMA produces several different morphological possibilities for a word, all with the same sequence of diacritics, then the word is counted as having a single solution. Figure 6.3 illustrates a list of solutions proposed by BAMA for a given input word.

```
INPUT STRING: ال يوم
LOOK-UP WORD: Alywm
   Comment:
     INDEX: P1W7
  SOLUTION 1: (Alyawom) [yawom_1] Al/DET+yawom/NOUN
     (GLOSS): the + today
  SOLUTION 2: (Alyawomu) [yawom_1]
Al/DET+yawom/NOUN+u/CASE_DEF_NOM
     (GLOSS): the + today + [def.nom.]
  SOLUTION 3: (Alyawoma) [yawom_1]
Al/DET+yawom/NOUN+a/CASE_DEF_ACC
     (GLOSS): the + today + [def.acc.]
* SOLUTION 4: (Alyawomi) [yawom_1]
Al/DET+yawom/NOUN+i/CASE_DEF_GEN
     (GLOSS): the + today + [def.gen.]
  SOLUTION 5: (Alyawom) [yawom_1] Al/DET+yawom/NOUN
```

Figure 6.3. Buckwalter-generated solutions for the word “Alywm”. The diacritized solutions that we are interested have been printed in bold.

If a particular word *w* has *n* unique solutions, then $D_w$ can be expressed as:

$$D_w = \{\, d_{w,1},\, d_{w,2},\, \ldots,\, d_{w,n} \,\} \qquad (22)$$

These solutions may be ranked by their likelihood, so that we have two sets of scored potential solutions, for $w \in W$. Revisiting the scored output of the text-based and speech-based diacritizers (4.8 and 5.4), we have two sets for *w*:

$$T_w = \{\, t \mid t = (d_{w,i}, t_{w,i}),\, d_{w,i} \in D_{w,}\, w \in W \,\} \qquad (10)$$

$$S_w = \{\, t \mid t = (d_{w,i}, s_{w,i}),\, d_{w,i} \in D_{w,}\, w \in W \,\} \qquad (21)$$

Where $S_w$ is the set of scored solutions from the acoustic model, and $T_w$ is from the text model.

As before, we reference the score of a selected tuple in these sets by $S_w[d_{w,i}]$ and $T_w[d_{w,i}]$. A weighted combination is then applied to the tuples in each set, and the final diacritized solution $d_w^*$ is that which maximizes the combination. This is expressed in the formula below:

$$d_w{}^* = \underset{d_{w,i} \in D_w}{\operatorname{argmax}} \left(\alpha . S_w[d_{w,i}] \ + \ \beta . T_w[d_{w,i}]\right) \tag{23}$$

We now discuss the method of combination in detail.

## 6.2 Algorithm for Weighted Interpolations

Text and speech cannot be directly interpolated. They are different classes of information, represented in different mediums.

The stream that the text-based diacritizer operates on is in the form of Buckwalter-encoded text. The text-based diacritizer predicts a vowel for each consonant. The text can therefore be said to comprise *consonant-vowel* pairs. This is pairing is necessary in order to with CRFs, which operate on a one-to-one mapping between observations and labels.

An important challenge that arises in combining the input text with the phonetic transcriptions of the speech is the fact that orthographic phonetic transcriptions cannot be converted into consonant-vowel pairs.

To illustrate this, we take an English word as example. The word “phone” in English consists of five letters, and it can be split into the following consonant-vowel pairs: (p, no-vowel), (h, o), (n, e). However, its phonetic transcription consists of only three phones with no direct way to map this into recognizable consonant-vowel pairs:

**[ f oh n ]**

Therefore, while the scores assigned by the text-diacritizer in Section 4.8 are summed over each character, the acoustic model’s scores are computed over phones.

Since we cannot map phonetic transcriptions to orthography, we use the acoustic log likelihood of the entire utterance *w* to derive the speech-based diacritizer’s scores.

Given that the length of the input text, || $W$ || = $M$, we now have two sets of tuples, $\{T_{w,}\ldots,T_M\}$, which we denote by $T$, and $\{S_w\ldots S_M\}$, denoted by $S$.

We are interested in interpolating them and selecting the highest scoring solution.

We follow the five steps below:

1. List BAMA solutions. Solutions must conform to the consonant-vowel consistency mentioned above.
2. Gather $T$ and $S$ together and sort tuples first by ID and then by the text-based scores in $T$. This is necessary in order to manage n-best experiments (in Section 0) and to extract solutions.
3. Apply a weighted interpolation, using Equation (23).

   This is done as follows:

```
foreach $w \in W$ do
    foreach $d_{w,i} \in D_w$ do
        $A_w = \alpha.S_w[d_{w,i}] + \beta.T_w[d_{w,i}]$
    end
end
```

Figure 6.4. Algorithm for weighted interpolations.

   If we are interested in selecting the n-best and computing oracle error rates, we interpolate only the top $n$.

4. Select $A_w$* that has the maximum score, from $A_w$, for each $w \in W$.
5. Given the IDs of the best solutions, we refer back to our list of BAMA's solutions and find the best analysis $A_w$*. This is the combined diacritizer's solution.

Figure 6.5 summarizes these steps.

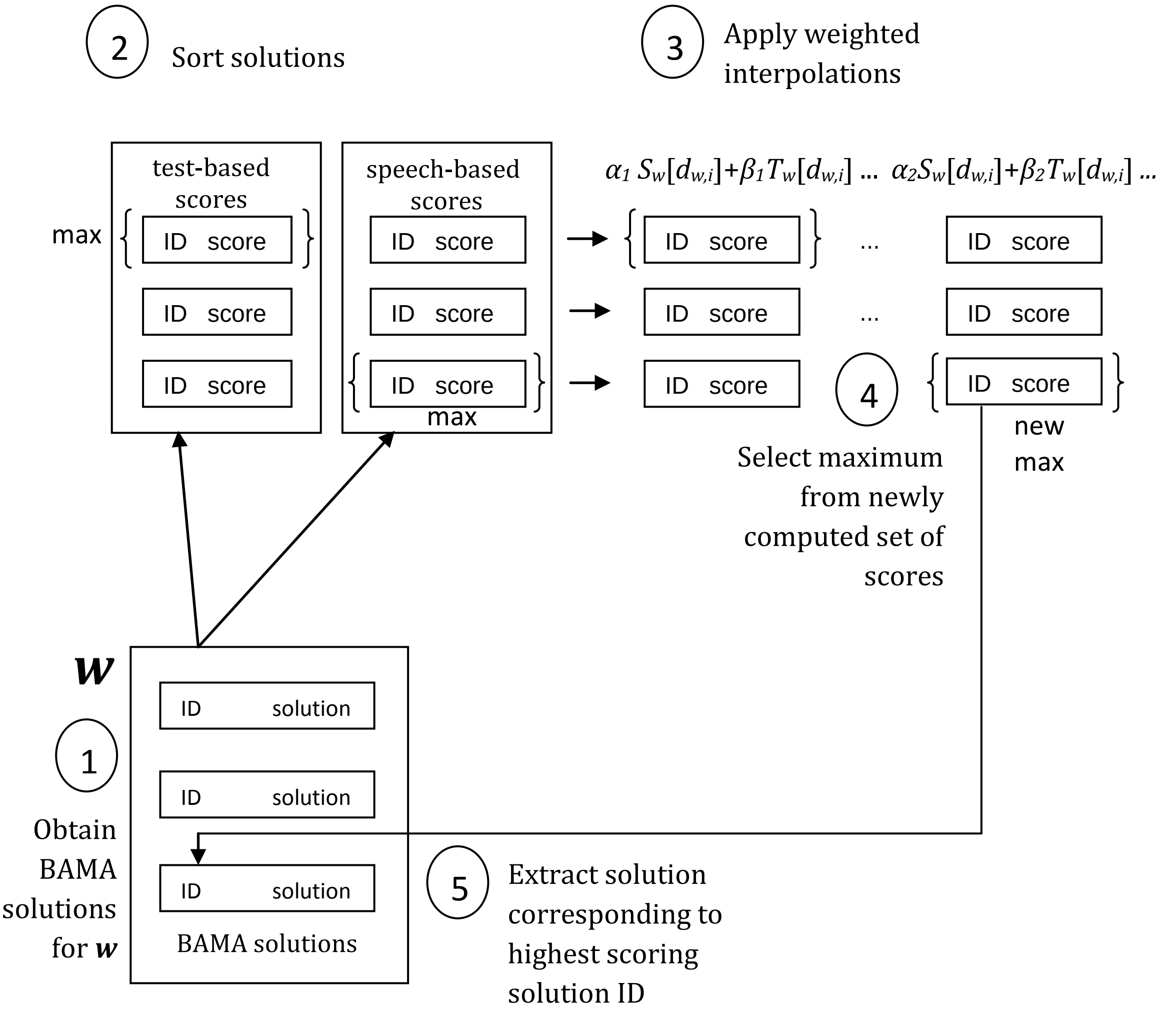

2
Sort solutions
3
Apply weighted interpolations
test-based scores
speech-based scores
$\alpha_1 S_w[d_{w,i}]+\beta_1 T_w[d_{w,i}]$ ... $\alpha_2 S_w[d_{w,i}]+\beta_2 T_w[d_{w,i}]$ ...
max
ID score
ID score
ID score
ID score
ID score
ID score
max
ID score
ID score
ID score
...
...
ID score
ID score
ID score
4
new max
Select maximum from newly computed set of scores
w
1
Obtain BAMA solutions for w
ID solution
ID solution
ID solution
BAMA solutions
5
Extract solution corresponding to highest scoring solution ID

Figure 6.5. Combining speech-based and text-based scores brings out the best of both. The solution with the highest score (indicated by max {}) may vary according to interpolation weights.

# 7. Data Processing

This thesis makes use of corpora developed by LDC [38], which provides large resources for researchers in many languages.

## 7.1 Training Data

For Arabic diacritization, Zitouni et al. [6] suggested a division of datasets from the Penn Arabic Treebank corpus, Part 3, v1.0: catalog number LDC2004T11 and ISBN 1-58563-298-8 [25]. This corpus is commonly known as ATB3. It consists of fully diacritized news stories from An-Nahar News[9], a leading Arabic daily newspaper in Lebanon. The corpus consists of Arabic articles and corresponding documents where each word in the source text is accompanied with its BAMA-provided solutions. Each of these solutions is annotated with all of its possible diacritics and morphological analyses. The corpus consists of 600 documents.

The division suggested by Zitouni et al. [6] is 85% for training and 15% for testing, where training data consists of approximately 288,000 tokens, and testing data of approximately 52,000 tokens.

### Speech

The speech-based diacritizer was trained on selections of the above training dataset suggested by Zitouni et al. [6]. Approximately 3 hours of human speech was recorded, based on native knowledge of diacritization, from documents 1 through 49 of January 2002.

### Text

The An-Nahar Arabic source articles in ATB3 are also present in another corpus provided by LDC, Arabic Gigaword Fourth Edition, LDC Catalog No. LDC2009T30, ISBN: 1-58563-532-4, from the GALE project [39] [40]. For training CRFs and

[9] www.annahar.com

replicating the results of Schlippe [5], we needed more data than the division suggested by Zitouni et al [6], so we used the An-Nahar documents from Arabic Gigaword Fourth Edition, for a total of approximately 471000 words.

## 7.2 Testing Data

The same testing data was taken from the division suggested by Zitouni et al. [6] was used throughout all of the experiments used in this thesis, for both speech and text. This dataset consists of 1886 sentences. However, 282 sentences were eliminated from this thesis, because they contained one or more words that were annotated with "NO SOLUTION" in the corpus. The final dataset was 1604 sentences. This made about 35350 words (excluding non-word tokens such as digits and punctuation).

## 7.3 Processing: Text-Based Diacritizer

Referring to the components of a text-based diacritizer (Section 4.7), the following steps are needed to build our diacritizer before we can diacritize an input text:

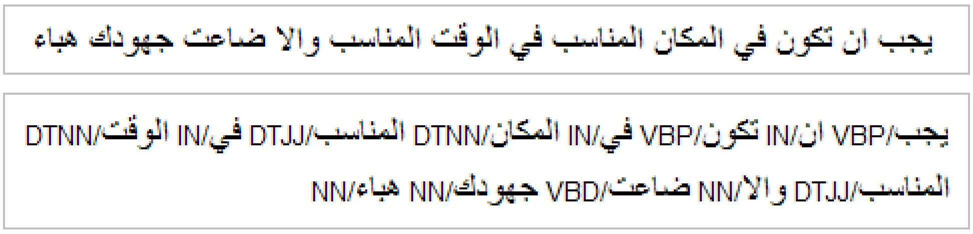
1. Extract the POS features from the training data.
2. Normalize the text.
3. Ensure consonant-vowel consistency.
4. Prepare data for the training of the CRFs.
5. Train the CRF model.

We now discuss the procedure in detail.

### Step 1. Feature Extraction.

We use CRF++ to work with our model. In reproducing the work of Schlippe [5], more data required was required than the training dataset suggested by Zitouni [6], so the POS tags presented in the ATB3 corpus were not used. The Stanford POS tagger was instead used to tag all the training data [41]. The 31 tags used by the Stanford POS tagger are listed in Appendix D.

The Arabic text was tagged by the POS tagger per sentence. The figure below shows the text before and after tagging:



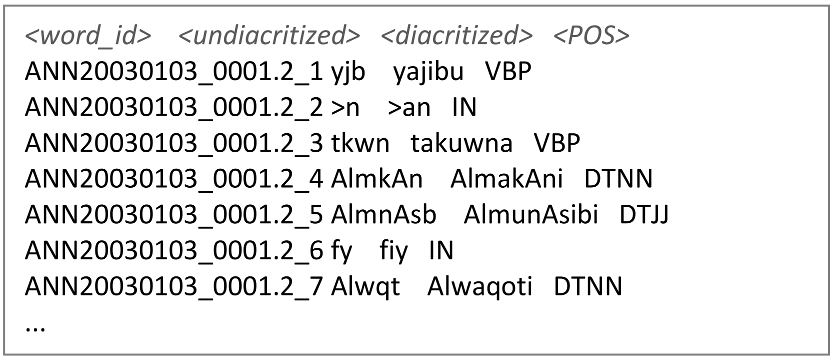
يجب ان تكون في المكان المناسب في الوقت المناسب والا ضاعت جهودك هباء

يجب/VBP ان/IN تكون/VBP في/IN المكان/DTNN المناسب/DTJJ في/IN الوقت/DTNN
المناسب/DTJJ والا/NN ضاعت/VBD جهودك/NN هباء/NN

Figure 7.1. POS-tagging on Arabic text.

Before training the CRFs, we also need the labels (or diacritics) of the given text, so that the CRFs can learn the dependencies. However, the Arabic Gigaword corpus [39] consists only of raw Arabic documents; it is not annotated with diacritized solutions. To generate the diacritics, we used MADA 3.1 (Section 4.4), which can take in UTF-8 Arabic and reproduce it in Buckwalter encoding, with diacritics included. This means that our own system incorporates an element of the diacritization error generated by MADA. Nonetheless, it was the best available option. MADA also tags text with its own set of POS tags, but since we intended to work with Schlippe's system we chose the Stanford tag set.

The training data is now tagged and diacritized in Buckwalter transliteration. Figure 17 below shows a sample of our data at this stage:



```
<word_id>   <undiacritized>   <diacritized>   <POS>
ANN20030103_0001.2_1 yjb    yajibu   VBP
ANN20030103_0001.2_2 >n    >an   IN
ANN20030103_0001.2_3 tkwn   takuwna   VBP
ANN20030103_0001.2_4 AlmkAn    AlmakAni   DTNN
ANN20030103_0001.2_5 AlmnAsb    AlmunAsibi   DTJJ
ANN20030103_0001.2_6 fy    fiy   IN
ANN20030103_0001.2_7 Alwqt    Alwaqoti   DTNN
...
```

Figure 7.2. Training data sorted and tagged.

### Step 2. Text Normalization.

Text normalization relies partly on some of the orthographic rules mentioned in Chapter 3. We apply the following rules:

- All the representations of hamza were mapped onto one character, ( ' ).
- The variant forms of *alif* ($A$ and | ) are mapped onto the single character $A$.
- The *Taa marbootah*, denoted by $p$ in Buckwalter transliteration, was converted to the consonant $t$ when it was diacritized, and to "h" otherwise.
- The glyph ` is taken as the short vowel a, (*fatha*).
- For conflicts between the diacritized and un-diacritized versions of text, such as between $h$ and $p$, and between $y$ and $Y$ (*alif maksoorah*), the un-diacritized reference was given preference. This is because the un-diacritized Buckwalter mapped directly from the Arabic text is more reliable as being closer to the intended source than the processed output of a diacritizer.
- Some conflicts consisted of additional consonants that were not found in the original text. Again, within the scope of this thesis, we assume that the input text is reliable and eliminate such conflicting options. This is reasonable since they we found that they comprise less than 1% of the data.

### Step 3. Consonant-Vowel consistency

The diacritized solutions must be arranged into consonant-vowel pairs, so that the length of each sequence of vowels $v_{w,i}$ for solution $d_{w,i}$ is equal to the that of its corresponding sequence of consonants, which we denote $c_{w,i}$. This is in keeping with the CRF model of two disjoint sets, X and Y, which are a corresponding mapping of observations to labels.

This means that each consonant must be labelled by a diacritic.

This raises two questions, what is the best to deal with consonants that have no diacritic, or those that have more than one?

Even in fully diacritized text, not every consonant is given a diacritic, some consonants conventionally have none, such as the *alif* in the definite article "*al*". This is distinct from the *sukoon* diacritic, which requires that no vowel sound be pronounced on the consonant in question, but the consonant is still vocalized. In most cases where there is no diacritic, the letter is either a conventional typographic practice, understood to always have a particular diacritic, or silent. Therefore in such a situation, if a consonant is not found to have a vowel after it, its diacritic was taken as <*empty*> [5].

The gemination diacritic (*shadda*) may appear in combination with other diacritics. In this case, as per the evaluation metrics of the research community, a geminated consonant in with another diacritic is taken to be a single, combined diacritic [6], [16].

The total number of diacritics is then fifteen labels. The Arabic diacritics in Section 4.8 are mapped onto:

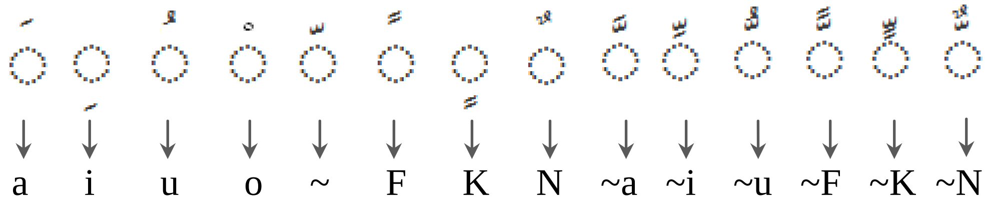


Figure 7.3. Arabic diacritics in Buckwalter Transliteration.

The table below shows consistency being maintained in mapping from un-diacritized and diacritized words.

| | |
|---|---|
| **Un-diacritized** | Alwaqt |
| **Diacritized** | Alwaqoti |
| **Consent-Vowel pairs** | (A,<empty>) (l,<empty>) (w,a) (q,o) (t,i) |

Table 7.1. Consonant-Vowel consistency.

Named Entity Recognition is beyond the scope of this thesis. Hence where there were conflicts in proper names, i.e. where the diacritized solution introduced new vowels or consonants into the word, no changes were allowed that were in conflict with the un-diacritized reference.

### Step 4. Prepare data for the training of the CRFs.

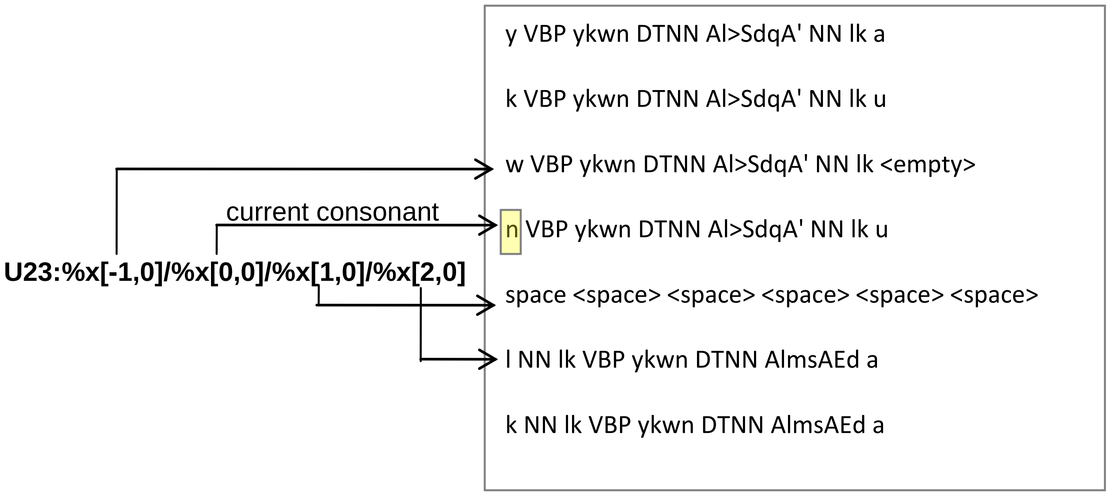
From our training data, we build feature files that conform to the input format required for CRF++. The feature files arrange our data and POS tags in columns, with a single observation, its features, and its label on each line. Figure 19 gives an example of training data converted into feature files for CRF++ [53]. We enforce word boundaries with the inclusion of the symbol <*space*>.

```
<consonant> <current POS> <previous word> <previous POS> <next word> <next POS> <label>
y VBP ykwn DTNN Al>SdqA' NN lk a
k VBP ykwn DTNN Al>SdqA' NN lk u
w VBP ykwn DTNN Al>SdqA' NN lk <empty>
n VBP ykwn DTNN Al>SdqA' NN lk u
space <space> <space> <space> <space> <space> <space> <empty>
l NN lk VBP ykwn DTNN AlmsAEd a
k NN lk VBP ykwn DTNN AlmsAEd a
space <space> <space> <space> <space> <space> <space> <empty>
A DTNN AlmsAEd NN lk DTJJ AlfEAl <empty>
l DTNN AlmsAEd NN lk DTJJ AlfEAl <empty>
m DTNN AlmsAEd NN lk DTJJ AlfEAl u
s DTNN AlmsAEd NN lk DTJJ AlfEAl <empty>
A DTNN AlmsAEd NN lk DTJJ AlfEAl <empty>
E DTNN AlmsAEd NN lk DTJJ AlfEAl i
d DTNN AlmsAEd NN lk DTJJ AlfEAl a
…
```

Figure 7.4. Training data prepared for the training of CRFs.

### Step 5. Train the CRF model.

CRF++ uses *templates* to further process the input training data. The template we built incorporates a bigram of vowels and a context of ±4, so that if the current consonant is $c_i$, then the above seven features are learnt for each of the 9 consonants in {$c_{i-4}$,…, $c_i$, …, $c_{i+4}$}. As an example, Figure 7.5 illustrates the context that is captured by the template command of *U23:%x[-1,0]/%x[0,0]/%x[1,0]/%x[2,0]*.



```
y VBP ykwn DTNN Al>SdqA' NN lk a
k VBP ykwn DTNN Al>SdqA' NN lk u
w VBP ykwn DTNN Al>SdqA' NN lk <empty>
n VBP ykwn DTNN Al>SdqA' NN lk u
space <space> <space> <space> <space> <space>
l NN lk VBP ykwn DTNN AlmsAEd a
k NN lk VBP ykwn DTNN AlmsAEd a
```

Figure 7.5. Sample template for CRF++.

The complete template used is listed in Appendix B.

CRF++ converts the input feature file according to the template, and the *crf_test* command is used to decode it. The figure below shows a sample of the output generated by CRF++ after testing.

```
# 0.087053
A...  ATIE....  i    i/0.913122    a/0.085395    i/0.913122    o/0.000000  u/0.001355...
T...  ATIE.....~a   ~a/0.993586    a/0.000815    i/0.000144    o/0.004988  u/0.000003 ...
I ... ATIE....   a    a/0.994231    a/0.994231    i/0.004455    o/0.000284    u/0.000306
E ... ATIE....   a    a/0.958163    a/0.958163    i/0.016923    o/0.008431    u/0.015059
space    .... <empty>/1. a/0.000000    i/0.000000    o/0.000000    u/0.000000    ~/0.000000
A ...AlnA}b    <empty>    <empty>/0.999972    <empty>/0.999972    /0.000015    i/0.000008...
l ... AlnA}b   <empty>    <empty>/0.999727    <empty>/0.999727    a/0.000168    i/0.000011....
n....AlnA}b   ~    ~/0.999754    a/0.000000    i/0.000000    o/0.000000    u/0.000000    ..
A... AlnA}b    <empty>    <empty>/0.999974    <empty>/0.999974    a/0.000000    i/0.000000...
} ... AlnA}b      i    i/0.999950    a/0.000019    i/0.999950    o/0.000002    u/0.000028    ~/0.000000...
b... AlnA}b      u   u/0.927505    a/0.022317    i/0.050107    o/0.000005    u/0.927505    ~/0.000...
```

Figure 7.6. CRF++ output. Each diacritic is listed with its marginal probability.

There are several columns, of which a few are shown. The first column shown represents the consonant *c* whose diacritic we need to predict. The current word is then shown, followed by the diacritic that the model predicted. After this, the marginal probability for each of the fifteen possible diacritics is given, of which a few are shown. The lexical scorer uses these probabilities to calculate the score of the current word, as given in formula (12) in Section 4.8. We store this score with $d_{w,i}$ to build tuples $s_{w,i}$ and the set $S_w$. A sample of the final output of the lexical scorer is presented in the figure below.

```
ANN20021015_0101_XML_1_1     ATIE      a o a a   -7.809711053
ANN20021015_0101_XML_1_2     ATIE      i ~a a a  -0.145843514
ANN20021015_0101_XML_1_3     ATIE      a o u u   -20.04889289
ANN20021015_0101_XML_1_4     ATIE      a o u a   -15.8958508
….
ANN20021015_0101_XML_2_1     ATIE      <empty> <empty> <empty> <empty>  -35.72480241
ANN20021015_0101_XML_2_2     AlnA}b    <empty> <empty> ~ <empty> i <empty>  -10.15257198
...
```

Figure 7.7. Different diacritization solutions with their scores, $t_{w,i}$.

The first column represents the ID of the solution (which is a concatenation of the word ID and its solution number *i*), the second column is the un-diacritized Buckwalter transliteration of the word, and the third represents the sequence of vowels of this solution. The last column gives the text-based diacritized score.

## 7.4 Processing: Speech-Based Diacritizer

The speech required for building the speech model was personally recorded in a clean environment for this thesis. Three hours were recorded by a human using native knowledge of Arabic diacritization from selections of the training data mentioned in 7.1, to build a single-speaker model. A sampling rate of 48 kHz was used, at 16 bits per sample.

To build the speech model prior to speech-based diacritization, we explain the processing required in the components discussed in 5.3.

### 1. Feature Extractor

We use HTK [42] as our toolkit for speech processing and recognition. The HCopy command is our feature extractor. Extracting the features is accomplished by the HCopy command as follows:

*HCopy -T 1 -C lib/cfgs/wave2mfcc.cfg -S lib/flists/train3hr_dataset.mfcc.scp*

where wave2mfcc.cfg converts the .wav sound files into feature vectors, using the configuration parameters below:

```
SOURCEFORMAT=WAV
TARGETKIND=MFCC_E_D_A_T_Z
TARGETRATE=100000.00
SAVECOMPRESSED=T
SAVEWITHCRC=T
WINDOWSIZE=256000
USEHAMMING=T
PREEMCOEF=0.97
NUMCHANS=26
CEPLIFTER=22
NUMCEPS=12
ENORMALISE=T
```

Figure 7.8. Configuration file for the acoustic model.

The main features extracted are listed in *TARGETKIND*, as MFCC_E_D_A_T_Z, i.e. MFCCs; energy; first, second and third time differentials; and the zero mean. For other features, refer to the HTK Book. [42]

Likewise, features are extracted from the speech files corresponding to the test data. The test data was read out and recorded personally for this thesis, producing 11 hours and 40 minutes of speech.

## 2. Acoustic Model

To model speech using HMMs, we begin by using the *HCompV* tool of HTK to create prototype definitions for each phone to be modelled. We create prototypes for 3-state left-to-right HMMs, where each state is represented by one Gaussian distribution, such that each distribution has the same mean and variance of the feature vectors of the training data. We duplicate the prototype for each phone that we plan to model for our speech.

In this thesis, we train the following 62 non-silent phones:

- The 28 Arabic consonants.
- 28 geminated variations of the above.
- The three short vowels (/**a**/, /**i**/, /**u**/).
- The two diphthongs (/**ay**/ and /**aw**/).
- An additional phone for the emphasized "*L*". This is a pharyngealized form of the alveodental alveolar ل (/**l**/), and is used uniquely in the pronunciation of the Arabic name for God, *Allah*. It is modelled as a separate phone since its pronunciation is notably different from the regular vocalization of ل (/**l**/).

To train the model, the HERest tool is then used to run the models through Baum-Welch re-estimation [32], [43].

Silence (**sil**) was also modelled and then tied with the short pause (**sp**) to make the model more robust [44].

The phones were converted into tied-state context-dependent triphones. The *HLEd* tool, which edits transcription files, was used to convert the monophone transcriptions to triphone transcriptions. An example of this conversion is shown in the figure below.

| | |
|---|---|
| e | e+i+tt2 |
| i | i+tt2+a |
| tt2 | tt2+a+l |
| a | a+l+a |
| l | l+a+E |
| E | a+E+a |
| a | E+a |

Figure 7.9. Monophone to triphone transcriptions.

Modeling speech as triphones improves robustness since more context is included in the training. The models were then cloned and the transition matrices tied (i.e. the parameters shared) in each triphone set, by the *HHED* tool.

Decision tree clustering is then applied to tie states together. Decision tree clustering [46] begins with the assumption that all states are equal in a single node, and by testing different classification questions, it uses decision tree algorithms to split the node such that the likelihood of diagonal covariance Gaussian of each child node is maximized [47]. The decision questions were based on Elshefai's classification of Arabic consonants [45].

However, a single Gaussian distribution is often not enough to represent a state, so to more precisely model the data, *HHED* was used to iteratively increase the number of Gaussian components per state. This process is called "mixing up". It is done by splitting the Gaussian distribution (with the highest prior) into two, and then perturbing the identical mean vectors by adding 0.2 standard deviations to one and subtracting them from the other [48]. The final number of components per state that yielded the highest accuracy was 24 components.

Four Baum-Welch iterations were carried out in between each iterative increase.

The accuracy was found by evaluating the test data using the *HVite* command to run the Viterbi algorithm for finding the best alignment between the transcript and the states of the model. In this step, the model was aligned against the reference transcript, *ref_cns_sp_am.mlf*, which was processed using the G2P layer (discussed below) to represent the test data's phonetic pronunciation, and the vowels alone were allowed in it, as displayed in the figure below.

```
#!MLF!#
"test_dataset_prompts_ANN20021015_0101_XML_1.lab"
i
a
a
.
"test_dataset_prompts_ANN20021015_0101_XML_2.lab"
a
a
i
u
.
```

Figure 7.10. Vowel reference transcript.

During this step we also retrieved the word boundaries of each word, as covered in Section 5.4.

The accuracy results of the evaluation (produced by the *HResults* tool) were:

```
,-------------------------------------------------------------.
| HTK Results Analysis at Tue Nov 15 00:57:57 2011              |
| Ref: lib/mlabs/shadd/ref_vwl_sp_am.mlf                        |
| Rec: res_24.mlf                                               |
|=============================================================|
|          # Snt |  Corr    Sub    Del    Ins    Err  S. Err |
|-------------------------------------------------------------|
| Sum/Avg | 1604  |  93.68   6.12   0.20   0.20   6.52  94.01 |
`-------------------------------------------------------------'
```

Figure 7.11. Accuracy results of the acoustic model.

6.52 was the best error rate obtained, down from 10.09 before mixing up the Gaussian components to 24.

### 3. G2P Layer

Similar to text normalization required for text processing, the G2P layer relies on some of the rules mentioned in Chapter 3. Given a fully diacritized word, the G2P rules that convert Buckwalter transliteration to phonetic transcriptions based on the 62 modelled phones are as follows [45]:

- Gemination of solar letters following the definite article ال (/**al**/), and elimination of the ل consonant from the phonetic pronunciation.
- Appending the consonant ن (/**n**/) for nunation.
- Conversion of an undiacritized ة (/**t**/), or the *taa' marbootah*, to ه (/**h**/).
- Normalization of *alif maqsoorah* (ى) to the long vowel *alif* (/**aa**/).
- Normalization of the glottal stop, or *hamza* (ء); variant orthographic forms of the glottal stop were mapped to a single phone.
- Eliminating *alif* of the definite article ال (/**al**/) if it is preceded by a prefixed consonant.
- Eliminating *alif* if it is the last letter of a word and is preceded by an undiacritized و.
- Words beginning with *alif* had the *alif* replaced by a glottal stop, since this more closely resembles the phonemic pronunciation of the text.

Additionally, the following constraints were added:

- Two consonants diacritized with a *sukoon* may not follow one another.
- No diacritic can appear independently, without a consonant.
- The two letters و (/**w**/) and ي (/**y**/) generally function as consonants, but are converted into diphthongs when preceded by the short vowel (/**a**/), while themselves being diacritized with a *sukoon*, or with no diacritic, and not followed by *alif*. However, since diphthongs are represented by the symbols (/**aw**/) and (/**ay**/) respectively, it was necessary to distinguish between words where *a* and *w*

were naturally juxtaposed and pronounced as consonants and when they represented a diphthong. To solve this problem, every phone in the phonetic transcription is separated by a space. Therefore $l\ a\ w$ becomes distinct from $l\ aw$.

## 4. Language Model

The solutions provided by BAMA are phonetically transcribed and used as a language model to constrain the speech recognizer's final recognition results from the acoustic model.

## 5. Scorer

The acoustic scorer is the final component of the speech-based diacritized, which actually scores the solutions provided by BAMA.  We cover the four-step processing of the scorer below.

**Step 1. Obtain all possible diacritized solutions.** The solutions provided by BAMA can be obtained from the ATB3 corpus. All *unique* solutions have to be extracted.

**Step 2. Convert solutions into their phonetic transcriptions.** This is done using the G2P layer mentioned above. Each solution $d_{w,i} \in D_w$ from BAMA is run through the G2P layer to obtain a phonetically transcribed version for alignment with the speech input. Figure 27 shows a few words after being phonetically transcribed.

| | | |
|---|---|---|
| Al$amsi | → | a sh2 a m s i |
| AlmunAsib | → | a l m u n a s i b i |
| fiy | → | f i |
| Altarobiy~api | → | a t2 a r b i y2 a t i |

Figure 7.12. Diacritized words before and after G2P processing

**Step 3. Force alignment from speech data to identify word boundaries.**

Incoming text is diacritized with the help of incoming speech. We need to identify the words in the speech data so that they may be aligned with the text. Using the *HVite* command, the speech data is aligned with the reference file, as used above in building the acoustic model. This time however, the segment times of alignment are also displayed the resulting output file. The times appear as in the figure below.

```
#!MLF!#
"test_dataset_prompts_ANN20021015_0101_XML_1.rec"
0 5500000 i
5500000 8600000 2a
8600000 9800000 a
9800000 11900000 a
11900000 12000000 sp
12000000 14000000 a
14000000 17700000 2a
17700000 19000000 i
19000000 21200000 u
21200000 21300000 sp
...
```

Figure 7.13. HVite output transcriptions, vowels predicted with time boundaries.

The first two columns in the output file represent the beginning and ending time of the pronunciation of the vowel. The second column consists of the vowel pronounced (the consonant is implicitly included). We deduce from the given figure that the time window for the first word (i.e. from the beginning to the first "sp") is 0 – 11900000 milliseconds. Since we exclude short pauses, the next word is pronounced from 12000000 to 2120000 milliseconds. We do this for each word *w* in the reference file, until we have all the time boundaries for each word. We generate an ID for each word from the filenames provided, and we store the word boundaries.

**Step 4 Scoring: Estimating maximum log likelihood**. Given the word boundaries extracted in step 1, we may identify the specific time frame against which we want to align each word solution *w* to obtain an acoustic score. The figure below shows a sample of the alignment script for the three solutions to $D_{ANN20021115\_0034\_XML\_2}$.

```
…ANN20021115_0034_XML_2_20_1.mfcc=…/ANN20021115_0034_XML_2.mfcc[1482,1551]
…ANN20021115_0034_XML_2_20_1.mfcc=…/ANN20021115_0034_XML_2.mfcc[1482,1551]
…ANN20021115_0034_XML_2_20_1.mfcc=…/ANN20021115_0034_XML_2.mfcc[1482,1551]
…
```

Figure 7.14. Each solution in the MFCC feature file aligned against its word boundaries.

The phonetic transcriptions of $d_{\text{ANN20021115\_0034\_XML\_2,1}}$,$d_{\text{ANN20021115\_0034\_XML\_2,2}}$ and $d_{\text{ANN20021115\_0034\_XML\_2,3}}$ are prepared as follows (only two are shown):

```
"*/test_dataset_prompts_ANN20021115_0034_XML_2_20_1.lab"
e
a
l2
a
t
i
y
.
"*/test_dataset_prompts_ANN20021115_0034_XML_2_20_2.lab"
e
a
l
a
t
a
y
```

Figure 7.15. Phonetic transcriptions of solutions ready to be aligned.

Scoring is done using the *HVite* command of the HTK.

The acoustic scores are normalized for the utterance, and the most likely solution selected by the acoustic model is then simply the solution with the maximum log likelihood. Note that the alignment of valid diacritization sequences is constrained by the dictionary.

The final output format of the acoustic scorer is identical to that in of the lexical scorer.

## 6. Dictionary

In addition to the above components, we also mention the building of the dictionary as part of the data processing.

We built two dictionaries, one for deriving acoustic scores, $s_{w,i}$, and one for the first the alignment that extracted word boundaries. Deriving the scores was based on a simple one-to-one mapping between phones, since we were given the labels against which we wanted to align the model.

However the initial alignment in extracting word boundaries required more processing. For this, our dictionary was a monophone, multi-pronunciation dictionary built to model regular consonants and case endings separately. Case endings were distinguished with an underscore. Given 28 consonants, 28 geminated versions of the same consonants, and the additional emphasized (/**l**/) consonant, this meant that the number of entries in the lexicon was doubled from 57 to 114.

Since this is a multi-pronunciation dictionary, each entry is repeated with its different pronunciations. We take the output of vowels as the different pronunciations. There are three short vowels, and a *sukoon*, for a total of 4 pronunciations for each entry. Therefore from 114 entries we have 114 x 4 = 456. However, 57 of the consonants are case endings, to which an additional three pronunciations, (the three *tanweens*, or nunation diacritics) may apply. Therefore, we add 57x3 = 171 more entries.

Diphthongs and long vowels also have to be accommodated. As mentioned above, the two letters و (/**w**/) and ي (/**y**/) function as diphthongs in some cases if preceded by (/**a**/), but they also function as long vowels when preceded by the short vowels (**/u/**) and (/**i**/) respectively. To deal with this, for every consonant, two more entries were added, one that was concatenated to (/**w**/) and one to (/**y**/). Each of these can have two possible pronunciations, as shown in the image below. We therefore now add these additional four entries for each of the initial 114 entries.

Finally, we add the silence and short pause entries, *sp* and *sil*, for a total of 1085 entries. Figure 7.16 lists samples of dictionary entries representing the different phones modelled in the system.

| | | |
|---|---|---|
| b | [o] | b |
| b | [a] | b a |
| b | [i] | b i |
| b | [u] | b u |
| t | [o] | t |
| t | [a] | t a |
| t | [i] | t i |
| t | [u] | t u |

| | | |
|---|---|---|
| b2 | [2o] | b2 |
| b2 | [2a] | b2 a |
| b2 | [2i] | b2 i |
| b2 | [2u] | b2 u |
| t2 | [2o] | t2 |
| t2 | [2a] | t2 a |
| t2 | [2i] | t2 i |
| t2 | [2u] | t2 u |

| | | |
|---|---|---|
| b2w | [2u] | b2 u w |
| b2w | [2aw] | b2 a w |
| b2y | [2i] | b2 i y |
| b2y | [2ay] | b2 a y |
| t2w | [2u] | t2 u w |
| t2w | [2aw] | t2 a w |
| t2y | [2i] | t2 i y |
| t2y | [2ay] | t2 a y |

| | | |
|---|---|---|
| _bw | [u] | b u w |
| _bw | [aw] | b a w |
| _by | [i] | b i y |
| _bw | [ay] | b a y |
| _tw | [u] | t u w |
| _tw | [aw] | t a w |
| _ty | [i] | t i y |
| _ty | [ay] | t a y |

Figure 7.16. Regular consonants, geminated consonants, diphthongs, and case endings.

This dictionary ensures that case endings are the only ones allowed to be diacritized by the *tanweens*.

## 7.5 Processing: Weighted Interpolation

Given $S_w$ and $T_w$, we are interested in interpolating them, and finding the optimal values of the interpolation weights.

We begin by associating both sets of scores, $S_w$ and $T_w$, with each solution, as listed below:

```
ANN20021015_0101_XML_1.1_1   ATlE  a o a a  -7.809711053    -63.3859
ANN20021015_0101_XML_1.1_2   ATlE  i ~a a a -0.145843514    -65.3133
ANN20021015_0101_XML_1.1_3   ATlE  a o u u  -20.04889289    -64.5849
ANN20021015_0101_XML_1.1_4   ATlE  a o u a  -15.8958508     -63.9714
ANN20021015_0101_XML_1.1_5   ATlE  a o u o  -20.62895333    -64.4178
ANN20021015_0101_XML_1.1_6   ATlE  a ~a i u -12.07641006    -67.6933
...
ANN20021015_0101_XML_1.2_1   AlnA}b <empty> <empty> ~ <empty> i <empty> ...
...
```

Figure 7.17. $S_w$ and $T_w$

All entries were sorted by solution ID and then by the scores in $T_w$. Interpolation can now be applied using equation (23).

From the complete set of interpolated results, we select for each word *w*, the solution producing the maximum interpolated score. This is the best solution for *w*. We take this solution and refer back to the list of BAMA solutions, $D_w$, to extract its vowel transcriptions, which we can evaluate against our reference.

In the case of selecting the n-best solutions and finding oracle error, we keep the solutions sorted by $T_w$ scores, as we are interested in seeing the effects of speech on text-based diacritization, and not vice versa.

## 8. Experiments: Weighted Combinations

All experiments in this chapter are tested using the same dataset proposed by Zitouni et al. [6]. The evaluation metrics are also the same, with two points to note.

1. Firstly, current literature on diacritization uses all tokens from the established datasets in calculating error, but we exclude numbers and punctuation. This is because there is no variation in learning these tokens' diacritics; they all have "*no diacritic*". Hence including them in evaluation portrays a slightly optimistic measure of the true diacritization accuracy of Arabic words.
2. In the case of computing error *without* case endings, calculations are usually made after removing the diacritics on the final consonant of a word, but letting the final consonant remain. For the same reason as above, we suggest a new metric: ***DER$_{abs}$*** *no CE*– this is the *absolute* diacritization error rate that *excludes* the final consonant

The results of the interpolation are listed in the table below.

| | **T** | **9:1** | **8:2** | **7:3** | **6:4** | **5:5** | **4:6** | **3:7** | **2:8** | **1:9** | **S** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **DER CE** | 4.4 | 3.9 | 3.5 | 3.1 | 2.6 | 2.2 | 1.9 | **1.6** | 1.6 | 2.3 | 4.7 |
| **DER$_{abs}$ no CE** | 2.1 | 2.0 | 1.8 | 1.7 | 1.6 | 1.5 | 1.4 | **1.3** | 1.4 | 2.0 | 4.4 |
| **DER no CE** | 1.7 | 1.6 | 1.5 | 1.4 | 1.3 | 1.2 | 1.1 | **1.0** | 1.1 | 1.6 | 3.5 |
| **WER CE** | 17.1 | 14.8 | 13.1 | 11.2 | 9.4 | 7.5 | 6.2 | **5.2** | 5.2 | 7.0 | 12.0 |
| **WER no CE** | 5.2 | 4.8 | 4.5 | 4.2 | 3.9 | 3.5 | 3.2 | **3.0** | 3.4 | 5.0 | 9.4 |

Table 8.1. Weighted interpolations of text and speech, using TEXT:SPEECH ratios. "CE" and "no CE" refer to Error Rates with and without Case Endings. (T: Text, S: Speech.)

The error rates of the individual text-based and speech-based diacritizers are listed in the *TEXT* and *SPEECH* columns in Table 8.1 above. For both columns, DER CE is comparable. However, within the *TEXT* column, DER$_{abs}$ differs from DER CE by 47.7%, while there is no significant reduction to DER$_{abs}$ in *SPEECH*. This confirms the hypothesis that text features contribute to systematic inflectional errors, while acoustic features generate regular errors throughout the text. For the same reason, we note that WER CE for *SPEECH* is significantly better than it is for *TEXT*, although in other evaluation metrics, *SPEECH* is worse than *TEXT*.

We further found that while the text-based diacritizer incorrectly predicted 13.6% of all case endings, the speech-based diacritizer only did so for 6.1%. This can be seen in Table 8.2, where we include the prediction accuracy of case endings only.

| | **Text** | **Speech** | **Best** |
|---|---|---|---|
| **CE only** | 13.6 | 6.1 | 2.8 |
| **Non-CE** | 2.1 | 4.6 | 1.3 |
| **Overall** | 4.4 | 4.7 | 1.6 |

Table 8.2 Prediction error of “CE only”: case endings alone; “Non-CE”: all other characters; “Overall”: both of the above. “Best” refers to the best TEXT:SPEECH ratio; i.e. 3:7.

We experimentally derived the optimal values of the interpolation weights, *α* for *TEXT* and *β* for *SPEECH*. The results in Table 8.1 show that the interpolation improves diacritization across all metrics consistently, until it peaks at $\alpha = 0.3$ and $\beta = 0.7$.

Most importantly, the “Best” column in Table 8.2 indicates that acoustic information plays a significant role in lowering the error rates of inflectional diacritics.

We found instances where the speech-based results were corrected by the text-based results, and vice versa. We illustrate an example of the latter. Figure 8.1 presents two tuples from sets $T_{109839}$ and $S_{109839}$. The set of solutions for word *109839* is $D_{109839}$, which consists of 17 solutions, each with an ID, the sequence of consonants of word *w* (*whm* in this case), and a sequence of vowels. The vowel sequence for Solution 6 is *aoi*, shown in the example. A solution is stored with its text-based score to form the top tuple, $(d_{109839,6}, t_{109839,6}) \in T_{109839}$, and with its acoustic score to form the bottom tuple, $(d_{109839,6}, s_{109839,6}) \in S_{109839}$.

| | | | |
|---|---|---|---|
| 109839_6 | w h m | a o i | -19.86843 |
| 109839_6 | w h m | a o i | -52.45680 |

Figure 8.1. Sample tuples from the scored sets.

The scores are combined as in Equation (22) to produce a new score, $a_{109839,6}$. Likewise, the scores from other solutions are combined, producing a set of 17 scores, $a_{109839,1}$, …, $a_{109839,17}$, from which the best solution may be selected.

Figure 8.2 shows the different solutions of the word *whm* chosen by the two diacritizers:

| | | | |
|---|---|---|---|
| (a) | 109839_6 | w h m | a o i |
| (b) | 109839_9 | w h m | a u <empty> |

Figure 8.2. Textually scored solution corrected by combination with acoustic score.

The text-diacritizer initially selected (a), which is the sixth solution, $d_{109839,6} \in D_{109839}$. Later, combined with the acoustic score, the speech-based diacritizer's choice, (b) $d_{109839,9}$, was finally selected. The correctly diacritized word is in fact *wahum*. (<*empty*> corresponds to "*no diacritic*").

Overall, we can conclude from our experiments that while text and speech do complement each other, acoustic information is a crucial factor in "canceling out" the text-based errors.

## 8.1 Varying base solutions

BAMA generates morphological analyses for words using linguistic rules hard-coded in three tables: *prefixes*, *stems* and *suffixes*. Words are segmented into prefix-stem-affix triples before they are analyzed against the corresponding tables.

As described in Section 4.4 above, in contrast to the **surface form** of words used above, MADA [4] operates on their **functional** form, extending BAMA with a lexicon of lexeme and feature keys. When a word is input, it is analyzed not by its triples, but by its lexeme and feature keys [23].

On average, MADA produces 2 less solutions for a word than BAMA[10]. As a diacritizer, it produces close to the most accurate results in the literature. We therefore investigate its use in our combined system.

We are interested in the effect of varying base solutions in our combined diacritizer.

We evaluate our diacritizer on three different sets of base solutions from which we select $d_w^*$: *BWT*, *MADA*, *ALL*.

*BWT* is the set of solutions generated by BAMA, *MADA* is the set generated by MADA (version 3.1) [49], and *ALL* is the union of *BWT* and *MADA*. The results are shown in Figure 8.3.

Despite the superiority of MADA for diacritization and morphological analysis, using its analyses as the base solutions in a combined diacritizer causes a reduction in accuracy. This suggests that BAMA's less rigorous analysis produces solutions that more closely resemble the linguistic insensitivity of speech.

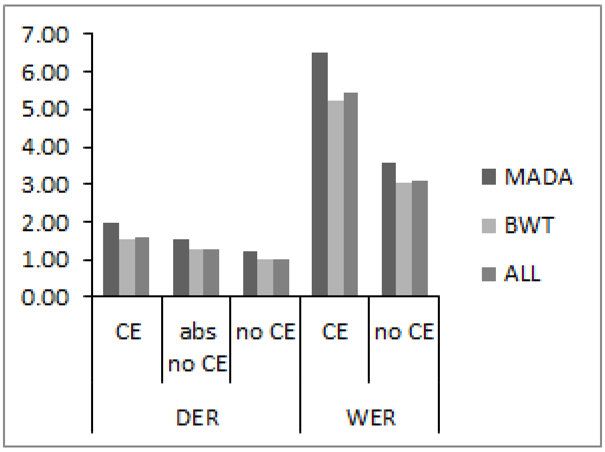


Figure 8.3. Comparing error from three sets of base analyses in combined diacritization.

This gives the speech-based diacritizer the flexibility to choose solutions differently from the text-based diacritizer. This disparity in the choice of each diacritizer is what combines the advantages of the text model's morphological sensitivity with the

[10] Calculated from the solutions provided by BAMA and MADA for the test dataset.

syntactic accuracy of the speech model. Note that inclusion of MADA's solutions in *ALL* slightly confuses the diacritizer's choices as well.

## 8.2 N-Best

We have concluded that the solutions generated by BAMA produce the best results in our system. From an efficiency and implementation point of view, we are interested to know how many solutions are actually necessary for this. We generate N-best lists and find the oracle error.

Using the existing setup, we select the *N* best solutions, scored by the text-based diacritizer, and interpolate them with speech. We vary *N* from 1 to 20, and plot the results in the graph below.

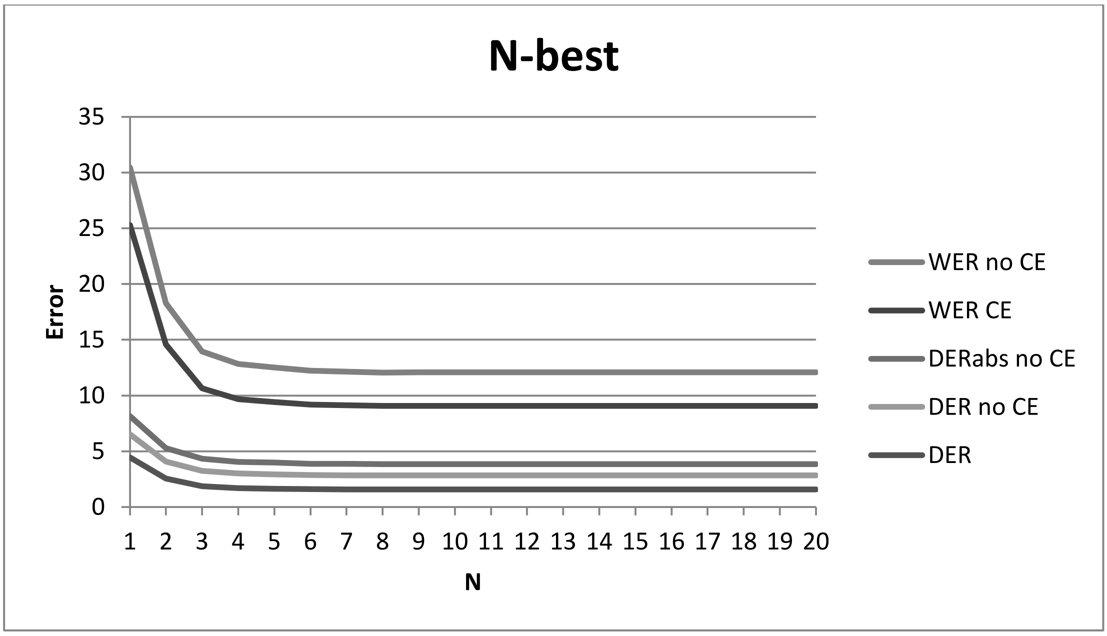


Figure 8.4. N-best solutions' error.

The error improves quickly until approximately N=5, after which there is little change in any metric, suggesting that the top 5 textually diacritized solutions are sufficient to achieve the best error rates.

## 8.3 Varying the Text-based model

As opposed to the probabilistic framework of CRFs, Support Vector Machines (SVMs) are non-probabilistic linear classifiers, explained above in Section 4.5. We compare the accuracy of SVM-based diacritization with our CRF-based method. We keep in mind that the SVMs of MADA, which are trained on 14 different features linguistic features, are used to tag possible solutions, while the CRFs are trained on one linguistic feature (POS), but a combination of other features extracted from the consonants alone. (Section 4.7)

To compare the performance of the SVM-based MADA approach with the CRF-based approach in our system, we used two models, $T^{crf}$ and $T^{svm}$, both of which selected solutions provided by MADA (in place of BAMA). We choose MADA solutions for compatibility, since MADA's model operates on the solutions produced by MADA's analysis engine, ALMORGEANA. We trained CRFs to build $T^{crf}$, with the same configurations in Chapter 7.3 (Step 5. Train the CRF model.), on the training data described by Zitouni et al. [6]. $T^{svm}$ is already available in MADA. The models were used to score the solutions for each *w* to produce the sets $T_w^{crf}$ and $T_w^{svm}$. The best diacritized solutions were selected according to formulas (7) and (8). Table 8.3 lists the results.

| | SVMs | CRFs |
|---|---|---|
| **DER CE** | 5.4 | 4.8 |
| **DER$_{abs}$ no CE** | 2.6 | 2.2 |
| **DER no CE** | 2.1 | 1.7 |
| **WER CE** | 20.1 | 18.4 |
| **WER no CE** | 6.4 | 5.3 |

Table 8.3. Text-based diacritization using CRFs vs. SVMs, before combining speech.

The individual scored sets, $T_w^{crf}$ and $T_w^{svm}$, were then interpolated with speech for each word *w*. The optimal combined values are reported in Table 8.4.

|  | SVMs | CRFs |
|---|---|---|
| **DER CE** | 2.8 | 2.0 |
| **DER$_{abs}$ no CE** | 2.2 | 1.5 |
| **DER no CE** | 1.8 | 1.2 |
| **WER CE** | 8.7 | 6.5 |
| **WER no CE** | 5.5 | 3.6 |

Table 8.4. Text-based diacritization using CRFs vs. SVMs, after combining speech.

Both Tables 8.3 and 8.4 above suggest that CRFs are superior to SVMs at modeling Arabic diacritic patterns. This is especially so given that SVMs were trained on several linguistic features, while the only linguistic feature explicitly learned by the CRFs is POS. This can be understood through the long context and consonant-vowel conditional dependencies across words that are captured by CRFs. We further explored the power of CRFs by testing a model that did not explicitly learn any linguistic features. Using a reduced subset of the training data (~200K words), two models were built, *With_pos* and *Without_pos*, and tested them on the same test dataset used throughout this thesis.

The configuration of *With_pos* and *Without_pos* was identical to our text model in Chapter 7 (Step 5. Train the CRF model.). The one difference was that the training of *Without_pos* excluded morphological information; the current, previous and next POS features were removed. The two models were allowed to freely diacritize text, without being constrained by BAMA's solutions.

Surprisingly, the results in Table 8.5 show that with the exception of WER CE, POS features offer little to no significant reduction in error, confirming the strength of CRFs in modelling diacritics without excessive information. However, the 20% reduction in WER CE from 21.4 to 17.1 shows that POS does offer some useful cues in predicting case endings.

|  | Without_POS | With_POS |
|---|---|---|
| **DER CE** | 5.4 | 5.4 |
| **DER no CE** | 3.5 | 3.4 |
| **WER CE** | 21.4 | 17.1 |
| **WER no CE** | 7.5 | 7.4 |

Table 8.5. CRF-based diacritization with and without learning linguistic features.

This invites the question, what other linguistic features are useful specifically for predicting case endings? We experiment with some options in the next chapter.

# 9. Experiments: Text-based Diacritization

## 9.1 Linguistic Features at Three Different Levels

Following the results from the previous chapter, we are interested in finding those features that contribute the most to improving diacritization, particularly case endings. Previous studies such as Rashwan et al. [16], Nelken and Scheiber [9] have combined diacritization techniques at various levels, such as character and word, or morpheme and word. Schlippe [5] compared diacritization at three different levels: phrase, word and character. However, Schlippe's study was in SMT rather than CRFs, which he concluded were likely to perform better than SMT approaches. Furthermore, these different levels that *have* been compared have been in terms of content, but not in terms of explicit features representative of the underlying syntax of the said level.

We propose here to study the diacritizing potential of *features* at different *structural* levels. That is, rather than only using training text in chunks of phrases, words and characters, we employ features that capture information from those levels of structure.

We will use BPC tags to capture phrase-level features, POS tags to capture word-level features, and proclitic information to capture morpheme-level features.

### BPC

We obtain the BPC tags from AMIRA 2.0 [50] [51], a non-recursive base-phrase chunker, which tags words based on their base phrase chunks. The BPC tags produced by AMIRA 2.0 are listed in Appendix D.

Base Phrase Chunking has not been previously explored for diacritization purposes. Revisiting the flexible syntax of Arabic mentioned in the Chapter 1, we recall that all of the following arrangements of **V**erb **S**ubject and **O**bject are valid in an Arabic sentence: ***VSO***, ***SVO***, ***VOS***, ***SOV***.

Recall also that the function of an Arabic word, such as whether it is the subject or object of a word, directly affects its case. This effect may often be extended from words to phrases, such as from a verb to an entire verbal phrase. The previous two facts, coupled with a flexible phrasal structure, mean that phrase-level tags may carry important information that spans across multiple words and helps in predicting the diacritics of individual words. The question is: whether or not the captured information is the same as that which is captured by other tags, such as POS. The hypothesis is that to some level, BPC must be able to improve inflectional diacritization.

### POS

The POS feature is first discussed in detail in Chapter 4, Section 4.1. The tags for the experiments in this section are obtained from MADA 3.1, and are listed in Appendix D.

### PRC1

The proclitic feature, *PRC1*, indicates what the first proclitic of a word is, and whether one is attached to the word or not. The values that PRC1 may assume are listed in Appendix E. Among the values are the different prepositional clitics in Arabic, which directly affect the inflectional diacritics of a majority class of nouns.

To illustrate a case of PRC1, consider the word *AlkitAbu* (meaning “the book”) in Arabic. If *AlkitAb**u*** (الكِتَابُ) is prefixed by the proclitic *bi* (بِ), meaning “by”, then the word becomes *bilkitAb**i*** (بِالكِتَابِ). (For the orthographic rules that eliminated the letter *A*, see Chapter 3.) For this word, the PRC1 feature is “bi_prep”. If the word had no proclitic, then PRC1 = 0.

The tags from each of the above three levels may be used to tag each word *w* from the test set.

Our experimental setup is as follows.

The training and testing data is the same subset of approximately 200K words used in Section 8.3, in our *With_POS* and *Without_POS* text models. We carry forward those two models, and for ease of comparison in the following experiments, we rename them to **consonants** and **consonants+POS**.

We train models on joint features and on multiple independent features. We denote joint features by concatenating them with **&**, and independent features with **+**. For example, a model that is trained on POS and PRC1 as two different features will be referred to using POS+PRC1. A model trained on POS and PRC1 as a single, joint feature will be referred to as POS&PRC1. For each feature added, the models are trained after incorporating the same features for the previous, current and next words.

Our results are as follows:

| **Features** | **DER** | **$\text{DER}_{\text{abs}}$ no CE** | **WER** | **WER no CE** |
|---|---|---|---|---|
| Consonants | 5.4 | 3.51 | 21.41 | 7.53 |
| | | | | |
| consonants+POS | 5.39 | 3.42 | **17.06** | 7.39 |
| consonants+PRC1 | 5.94 | 3.54 | **18.93** | 7.73 |
| consonants+POS+PRC1 | 5.8 | 3.38 | **19.05** | 7.42 |
| consonants+POS&PRC1 | *7.47* | *3.34* | *26.28* | *7.29* |
| consonants+POS+PRC1+POS&PRC1 | 6.21 | 3.38 | **20.74** | 7.38 |
| | | | | |
| consonants+BPC | 6.26 | 3.66 | **19.99** | 7.96 |
| consonants+POS+BPC | 6.77 | 3.56 | 22.76 | 7.67 |
| consonants+POS&BPC | 6.53 | 3.5 | 21.83 | 7.56 |
| consonants+POS+BPC+POS&BPC | 6.81 | 3.53 | 22.97 | 7.59 |

Table 9.1 Comparing features at three levels: CRFs.

The first section of the table focuses on POS and PRC1; the second section focuses on POS and BPC. Consistent with the results in Section 8.3, consonants alone are sufficient to diacritize with reasonable accuracy. However, comparing the *WER* and *WER no CE* columns, we see that the other two features do improve inflectional diacritization. While BPC mostly reduces accuracy, our hypothesis of BPC being able to diacritize case endings is not incorrect, since it still gives 6.6% improvement of *WER no CE* over consonants. However, comparing the performance of **consonants+BPC** and **consonants+PRC1**, BPC is less effective than the morpheme-level PRC1. Also, the word-level POS feature is still the highest contributor to inflectional diacritization accuracy.

A possible justification for the phrase-level results may be that there are fewer BPC tags in comparison to the number of POS tags; therefore the information provided by BPC is not discriminative enough. It would be interesting to see if the errors decrease with more training data. However, given these findings, phrase-levels features do not appear to be useful in comparison to word-level features.

POS and BPC (word and phrase level) features, when used together as independent features, increase consonants' *WER* by 6.3%, but relating them as a joint feature brings them closer to the performance of consonants, with only 2.0% increase. These results are indicative of the dependency between word and phrase level features. After all, a phrase's "verbal phrase" tag requires the presence of a word that has "verb" as its POS tag. The drop in accuracy when POS and BPC appear together may also be the result of giving unique treatment to two features that are inherently dependent – to the point that one of them is almost subsumed by the other.

The POS and PRC1 (word and morpheme level) features show the opposite behaviour. PRC1 performs next to POS in predicting case endings, and better than phrase-level BPC when used as an independent feature with POS. But when connected with POS as a joint feature, it brings the greatest *increase* (22.7%) in the *WER* from that of consonants. This can be further seen in the fact that POS&PRC1

creates the highest increase in *DER* as well (38.3%). On the other hand, it generates the highest decrease in error *without* case endings (*DER no CE* and *WER no CE*). This appears to be in confirmation with the results of Rashwan et al. [16], who show that diacritization at the level of morphemes may have advantages over word-level.

When word and morpheme-level features are used together independently as well as in joint features (consonants+POS+PRC1+POS&PRC1), they offer small improvement across all metrics except DER.

When the same is done with word and phrase-level features, they appear only to add noise.

Overall, the findings show that the word-level POS feature remains the most effective for diacritization, and in particular for case endings, followed by PRC1.

In order to examine how strongly these conclusions correspond to the nature of the features chosen versus how much they are influenced by the learning model, we perform one more experiment. We attempt to examine similar features while using SVMs in the learning model instead of CRFs.

MADA is employed once again, to take advantage of its SVM-based model. However, incorporating features that are not already available in MADA's implementation is complex, so our experiment excludes BPC and joint features, and offers only a partial comparison to the CRF-based results of this section.

The baseline model of MADA consists of 19 features, as discussed in Section 4.5 (Tables 4.1 and 4.2). For this experiment, unit weights were assigned to all feature classifiers. In this way, the effect of individual features could easily be "turned on or off" by assigning them weights of 1 or 0.

The results obtained on the test data are listed in Table 9.2 below

| Features | DER | DER$_{abs}$ no CE | WER | WER no CE |
|---|---|---|---|---|
| Baseline | 5.25 | 1.93 | 19.44 | 5.84 |
| Baseline+POS | 5.23 | 1.90 | 19.34 | 5.74 |
| Baseline+PRC1 | 5.20 | 1.93 | 19.35 | 5.82 |
| Baseline+POS+PRC1 | 5.19 | 1.90 | 19.28 | 5.74 |

Table 9.2 Comparing POS, PRC1 and POS+PRC1 using SVMs

In general, these results appear more accurate than those of our CRF-based model in Table 9.1. This does not mean that SVMs provide a better model for diacritization than CRFs, as we recall that the results in Table 9.1 were derived from a sub-optimal CRF-model that was stripped of its next and previous words' context.

The results in Table 9.2 do suggest that some of the feature behaviour we observed in Table 9.1 may be partially attributed to the learning model. In particular, we note that the baseline is improved by the addition of other features in almost all cases, and that the best case is always that with the most features turned on. In contrast, the baseline CRF model was already performing at an almost optimal level, before the inclusion of additional features. We also note that the differences in error rates of Table 9.2 are not as varied as they were in the CRF-based model.

While we do not have the results for joint features, we can see that the independent combination of POS and PRC1 does not increase error. We also see that similar to the results in the CRF model, the POS feature improves the results across all metrics. With the exception of DER, it offers greater improvement than PRC1. In fact, when POS and PRC1 are combined, we note that the improvement of Baseline+POS+PRC1 over Baseline+POS is either minimal, and where case endings are not included, (in the *DER*$_{abs}$ *no CE* and *WER no CE* columns) it is non-existent. This also means that the PRC1 feature offers some insight into case endings.

From the feature trends of this short SVM-based experiment, we arrive at the same conclusion regarding POS and PRC1 that we did while using CRFs. That is, word-level features are the most useful for improving diacritization accuracy. The only exception is in DER, where POS decreases the baseline from 5.25 to 5.23, while PRC1 pulls it down to 5.20. Their independent combination brings it to 5.19. In the *$DER_{abs}$ no CE* column, we note than any improvement is brought about only by POS, with the PRC1 offering no error reduction on its own or in combination with POS.

To the *WER* baseline of 19.44, POS brings a 0.1 reduction in error, PRC1 brings 0.9, and their independent combination brings a reduction of 0.16. To *WER no CE*, POS brings a reduction of 0.1 to the baseline again, while PRC1 brings only 0.02. Together their induced reduction is 0.1.

Combining the results from the above two experiments, we conclude that word and morpheme-level features are independent and that word-level features are most useful in improving diacritization. From Table 9.1 we also deduced that phrase-level features are the least useful in inflectional diacritization.

Considering the fact that our experiments have shown PRC1 to be next to POS in increasing accuracy, that PRC1 may carry information relevant to case endings, and that Rashwan et al. [16] achieve superior results using diacritization at morpheme-level in one of the modes of their diacritizer, we are interested to see how far we can take advantage of morpheme-level diacritization using CRFs.

## 9.2 Token-Level Diacritization

We use TOKAN [49], MADA's companion tokenizing tool, to tokenize the training data. We tokenize all clitics[11], both proclitics and enclitics. To take an example of clitic tokenization, we consider the word *AlkitAbu* ("the book") once again. Al is a clitic, it is the definite article in Arabic (meaning "the"), and is prefixed to the word it defines. In tokenized form, *AlkitAbu* will appear as follows: *Al+ kitAbu*.

[11] This is done using the D3 scheme. Details can be found in the user manual available with the MADA+TOKAN download from [49].

The training text was tokenized and used to train the CRFs. However, distinct from Rashwan [16] and Neilken and Sheiber [9], we *retained* the word level information during training, as shown in the figure below, via the word boundaries.

```
A         noun    Al        AlnA}b  verb    ATlE      ATlE      noun    nA}b      AlnA}b
l         noun    Al        AlnA}b  verb    ATlE      ATlE      noun    nA}b      AlnA}b
n         noun    nA}b      AlnA}b  noun    Al        AlnA}b  adj       Al        AlEAm
A         noun    nA}b      AlnA}b  noun    Al        AlnA}b  adj       Al        AlEAm
}         noun    nA}b      AlnA}b  noun    Al        AlnA}b  adj       Al        AlEAm
b         noun    nA}b      AlnA}b  noun    Al        AlnA}b  adj       Al        AlEAm
<space><space><space><space><space><space><space><space><space><space>
A         adj     Al        AlEAm   noun    nA}b      AlnA}b  adj       EAm       AlEAm
l         adj     Al        AlEAm   noun    nA}b      AlnA}b  adj       EAm       AlEAm
....
```

Figure 9.1 Tokenized words for CRF training.

The format of training our CRFs is:

<consonant> <current_pos> <current_token> <current_word> <previous_pos> …[12]<previous_token> <current_word> <next_pos> <next_token> <next_word>

The results for the above experiment are given in the table below. The error rates improve across all metrics except WER.

| Features | DER | $\text{DER}_{abs}$ no CE | WER | WER no CE |
|---|---|---|---|---|
| consonants+POS (words) | 5.39 | 3.42 | 17.06 | 7.39 |
| consonants+POS (tokens) | 4.95 | 2.6 | 19.06 | 7 |

Table 9.3 Tokenized words versus full words

No previous studies have discussed diacritization at token level using CRFs. Our results are in agreement with Rashwan et al [16] that morphemes are more useful units from which to predict diacritics than words. Different templates, training formats, and feature combinations could be experimented with in future. Although further investigation is required to discover which features besides POS, if any, are most useful for inflectional diacritization, we have found promising results in tokenizing before diacritization. In combination with the speech-based diacritizer, we expect that

[12] The format of features is in one line, broken here due to lack of space.

using tokenized words in training for the text-based diacritizer will help the overall combined diacritization to generate more accurate results.

# 10. Conclusions and Future Work

Many studies have dealt with diacritization of Arabic in a text-based environment, but have not been able to predict case endings with reliable accuracy.

This thesis explores the possibility of bringing speech into the process. No previous work on diacritization has been done that studies the effects of speech for its own merits, rather than as a sub-task of a broader objective.

The main contribution of this thesis is to propose and demonstrate a system which validates the hypothesis that speech input has advantages that can be used to build successful diacritization systems. The results reported are promising, and the diacritization of the speech-and-text combined system has lower error rates than the best systems reported in the literature. The most significant contribution is the improvement of the diacritization of case endings. Other sub-areas have been investigated, and are summarized below with the overall conclusions to the thesis' contributions.

## 10.1 Conclusions

**1. Combining speech with text-based diacritization**. Individual speech and text-based diacritizers were built and evaluated. A weighted combination of the systems was proposed and described. Interpolating text and speech poses some challenges since the processing of phonetic information for speech recognition and orthographic information for linguistic methods are different. The method of combination proposed consists of a weighted interpolation of the scoring of all possible diacritization solutions. Acoustic scores are generated by a HMM-model of speech, and textually derived scores by modelling the consonant and diacritic dependencies using CRFs. For interpolation, the acoustic scores were generated at word-level and the textually-derived scores were generated at diacritic-level.

N-best lists for the system were explored, and it was found that N=5 is sufficient for the system's performance.

Within the combined framework, the use of different base solutions was studied. The solutions provided by MADA, a SVM-based morphological analyzer, and BAMA, a rule-based analyzer, were compared. It was found that BAMA produces the solutions that best allow the advantages of acoustic information to be used. The disparity between the speech-based and text-based diacritizers is what allows their strengths to be combined.

We also varied the text-based model of the combined framework. CRFs and SVMs were compared in this context. The comparison revealed that a CRF-based model without being trained on morphological information did almost as well as a model with that training. CRFs were found to have strengths over SVMs that can be utilized for diacritization even in the absence of sophisticated annotated corpora. The advantages stem from the very structure of the probabilistic, contextual model of CRFs, compared to the linear classification technique of SVMs.

**2. Varying features at different levels for text-based diacritization in CRFs.**

The above comparison between CRFs and SVMs led to a study of features that are most suitable for predicting diacritics. Phrase-level, word-level and morpheme-level features were compared, using CRFs. Previous studies [5], [9], [12], [16] have studied diacritization at various levels but typically in the context of chunks of text, rather than features.

It was hypothesized that phrase-level features would carry long syntactic context across words and aid in inflectional diacritics. The initial experiments showed that phrasal information is in fact not significantly helpful in predicting case endings. The morpheme-level information carried by the proclitic feature was found to be more useful than phrase-level BPC tags. Word-level POS tags and morpheme-level PRC1 tags were most useful for inflectional diacritization. These results were concluded after experimenting with various combinations of the three features on a CRF-based model as well as a SVM-based one.

A model was also built on data in which all clitics were tokenized, while retaining word information. The results obtained were better than any of the other feature combinations.

## 10.2 Future work

The experiments in Chapter 9 only explored a few features. Further research is needed into features that prove useful for predicting case endings, using CRFs. Although the features studied do not improve diacritic error rates at character level, it may be possible to incorporate a model of the optimal features into the prediction of case endings specifically. This may be done using factored language models (FLMs) [52].

Based on features that are discovered to be useful, it is planned to involve LMs in the system. They will be incorporated into the solutions-scoring layer of weighted interpolations. Vowel LMs are currently being experimented with. The improved system results can be further used to train a new and more accurate CRF model.

Inflectional diacritics are difficult to predict even for lay humans, and consequently, fully diacritized books for educational purposes tend to have high error rates. Pure text-based diacritization systems, while accurate in lexemic diacritization, do not produce the best results in case endings either. The current system has an accuracy rate of inflectional diacritization better than typically available in existing reading texts for non-natives, and since it uses speech, it may be found to be a more natural and efficient way to diacritize text. It could therefore be used to complement the production of fully diacritized texts, which are necessary for learners of the language.

Moreover, since combined diacritization is a general framework, more sophisticated underlying speech and text models are likely to lead to greater reduction in error.

However, in the interest of efficiency and limited expert knowledge, an interesting side question that arises is how much speech is necessary for an accurate diacritization system. To begin with, the vocalization of case endings alone may prove to be sufficient.

## Bibliography

1. Habash, Nizar; Roth, Ryan; Rambow Owen; Kulick, Seth and Mitch Marcus. 2007. Determining Case in Arabic: Learning Complex Linguistic Behavior Requires Complex Linguistic Features. In Proceedings of EMNLP-CoNLL'2007. pp.1084-1092, Prague.

2. Kirchoff, Katrin and Vergyri, Dimitra. 2004. Automatic Diacritization of Arabic for Acoustic Modeling in Speech Recognition. In COLING Workshop on Arabic-script Based Languages, Geneva, Switzerland.

3. Habash, Nizar and Rambow, Owen. 2007. Arabic Diacritization Through Full Morphological Tagging. In Proceedings of NAACL HLT 2007. Companion Volume, Short Papers, New York, USA.

4. Habash, Nizar; Rambow Owen; Roth, Ryan. 2009. MADA+TOKAN: A Toolkit for Arabic Tokenization, Diacritization, Morphological Disambiguation, POS Tagging, Stemming and Lemmatization. In Proceedings of the 2nd International Conference on Arabic Language Resources and Tools (MEDAR), Cairo, Egypt.

5. Schlippe, Tim; Nguyen, ThuyLinh; Vogel, Stephan. 2008. Diacritization as a Machine Translating Problem and as a Sequence Labeling Problem. In Proceedings of the Eighth Conference of the Association for Machine Translation in the Americas (AMTA), Hawai'i, USA.

6. Zitouni, Imed; Sorensen, Jeffrey S; Ruhi, Sarikaya. 2006. Maximum entropy based restoration of arabic diacritics. In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pp.577-584, Sydney, Australia.

7. Ya'akov Gal. 2002. An HMM approach to vowel restoration in Arabic and Hebrew. In Proceedings of the ACL-02 workshop on Computational approaches to semitic languages, PA, USA

8. Husni Al Muhtaseb Mustafa Elshafei and Mansour Al-Ghamidi. 2006. Statistical Methods for Automatic Diacritization of Arabic Text. In Proceedings of the Saudi 18th National Computer Conference (NCC18), Riyadh Saudi Arabia.

9. Nelken, Rani and Sheiber, Stuart M. 2005. Arabic Diacritization Using Weighted Finite-State Transducers. In Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages, Ann Arbor, Michigan, USA.

10. Tarek A. El-Sadany and Mohamed A. Hashish. 1989. An Arabic Morphological System. IBM Systems Journal, 28(4).

11. Brown, Peter; Pietra, Della; Mercer, Robert. 1993. The Mathematics of Statistical Machine Translation. Computational Linguistics - Special issue on using large corpora: II, vol 19 Issue 2. MIT Press Cambridge, MA, USA.

12. Emam, Ossama and Fischer, Volker. 2005. Hierarchical Approach for the Statistical Vowelization of Arabic Text. Technical report, IBM Corporation Intellectual Property Law, Austin, TX, US.

13. Mohri, Mehryar (2004). Weighted Finite-State Transducer Algorithms: An Overview. Formal Languages and Applications 148 (620): pp.551–564.

14. Berger, Adam L.; Della Pietra, Stephen A.; Della Pietra, Vincent J. 1996. A Maximum Entropy Approach to Natural Language Processing Computational Linguistics, Vol. 22, No. 1. (1996), pp. 39-71.

15. Jaynes, E. T. 1957. "Information Theory and Statistical Mechanics II". Physical Review Series II 108 (2): pp.171–190

16. Rashwan, M.A.A.; Al-Badrashiny, M.A.S.A.A.; Attia, M.; Abdou, S.M.; Rafea, A. 2011. A Stochastic Arabic Diacritizer Based on a Hybrid of Factorized and Unfactorized Textual Features. IEEE Transactions on Audio, Speech, and Language Processing, vol.19, no.1, pp.166-175.

17. Dempster, A.P.; Laird, N.M.; Rubin, D.B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal of the Royal Statistical Society. Series B (Methodological) 39 (1): pp.1–38.

18. Ananthakrishnan, S.; Narayanan, S.; Bangalore, S. (2006). Automatic diacritization for arabic transcripts for automatic speech recognition. Proceedings of International Conference on Natural Language Processing.

19. Nizar, Habash. 2010. Introduction to Arabic natural language processing (Synthesis Lectures on Human Language Technologies). 1 ed., pp.27-37. Morgan & Claypool.

20. Hayes, Judith. 1993. Language recognition using two- and three-letter clusters. School of Computer Studies, University of Leeds.

21. Buckwalter, Tim. 2004. Arabic Morphological Analyzer version 2.0. LDC2004L02. < http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2004L02> retrieved 17th January 2012.

22. Nizar, Habash and Rambow, Owen. Arabic Tokenization, Morphological Analysis, and Part-of-Speech Tagging in One Fell Swoop. In Proceedings of the Conference of American Association for Computational Linguistics (ACL05).

23. Nizar, Habash. 2010. Introduction to Arabic natural language processing (Synthesis Lectures on Human Language Technologies). 1 ed., pp.70-72. Morgan & Claypool.

24. Burges, Christopher J.C. 1998. “A Tutorial on Support Vector Machines for Pattern Classification.” Data mining and Knowledge Discovery, 2, pp.121-167. The Netherlands.

25. Maamouri, Mohamed; Bies, Ann; Buckwalter, Tim; Jin, Hubert. 2004. Arabic Treebank: Part 3 v 1.0. LDC Catalog No.: LDC2004T11. ISBN: 1-58563-298-8. <http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2004T11>

26. Roth, Ryan; Owen, Rambow; Nizar, Habash; Diab, Mona; Rudin, Cynthia. 2008. Arabic Morphological Tagging, Diacritization, and Lemmatization Using Lexeme Models and Feature Ranking. In Proceedings of Association for Computational Linguistics (ACL), Columbus, Ohio.

27. Lafferty, John D. ; McCallum, Andrew ; Pereira, Fernando C. N. 2001.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., - ISBN 1-55860-778-1, S. pp.282-289

28. Young, S. J.; Evermann, G.; Gales, M. J. F.; Hain, T.; Kershaw, D.; Moore, G.; Odell, J.; Ollason, D.; Povey, D.; Valtchev, V.; Woodland, P. C.. 2006. The HTK Book, version 3.4, pp. 75-76. Cambridge University.

29. Davis, S. B. and Mermelstei, P. N. 1980. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE Transactions on Acoustic, Speech and Signal Processing, 28(4):pp.357-366.

30. Young, S. J.; Evermann, G.; Gales, M. J. F.; Hain, Kershaw, T.; D.; Moore, G.; Odell, J.; Ollason, D.; Valtchev, V.; Povey, D.; V.; Woodland, P. C.. 2006. The HTK Book, version 3.4, pp.79-80. Cambridge University.

31. Rabiner, Lawrence R.. 1989. A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE. pp. 257-286.

32. Eisner, Jason. 2002. An interactive spreadsheet for teaching the forward-backward algorithm. In Dragomir Radev and Chris Brew (eds.), Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching NLP and CL, pp. 10-18. Philadelphia.

33. Khe Chai, Sim. www.comp.nus.edu.sg/~simkc/slides/lecture06.pdf <lecture slides>

34. Forney GD. 1973. "The Viterbi algorithm". Proceedings of the IEEE 61 (3): 268–278

35. Khe Chai, Sim. www.comp.nus.edu.sg/~simkc/slides/lecture05.pdf <lecture slides>

36. Gales, M. J. F.; Diehl, F.; Raut, C. K.; Tomalin, M.; Woodland, P. C.; Yu, K. 2008. Development of a Phonetic System for Large Vocabulary Arabic Speech Recognition. IEEE Workshop on Automatic Speech Recognition & Understanding 2007 (ASRU). pp. 24-29. Kyoto, Japan.

37. http://www.ldc.upenn.edu/myl/morph/buckwalter.html

38. Linguistic Data Consortium. http://www.ldc.upenn.edu/

39. Arabic Gigaword 4th Edition.
http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2009T30

40. Global Autonomous Language Exploitation. (*GALE* ).
http://www.darpa.mil/Our_Work/I2O/Programs/Global_Autonomous_Language_Exploitation_%28GALE%29.aspx

41. Kristina Toutanova and Christopher D. Manning. 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000), pp. 63-70.44. http://crfpp.sourceforge.net/

42. Young, S. J.; Evermann, G.; Gales, M. J. F.; Hain, Kershaw, T.; D.; Moore, G.; Odell, J.; Ollason, D.; Valtchev, V.; Povey, D.; V.; Woodland, P. C.. 2006. The HTK Book, version 3.4. Cambridge University

43. Baum, L. E.; Petrie, T.; Soules, G; Weiss, N. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains, Ann. Math. Statist., vol. 41, no. 1, pp. 164–171.

44. Young, S. J.; Evermann, G.; Gales, M. J. F.; Hain, Kershaw, T.; D.; Moore, G.; Odell, J.; Ollason, D.; Valtchev, V.; Povey, D.; V.; Woodland, P. C.. 2006. The HTK Book, version 3.4, pp.35-36. Cambridge University.

45. Elshafei, M. 1991. Toward an Arabic text-to-speech system. The Arabian Journal for Science and Engineering 16(4), pp. 565–583.

46. Liu, Bing; Xia, Yiyuan; Yu, Philip S. 2000. Clustering Via Decision Tree Construction. In Proceedings of the ninth international conference on Information and knowledge management (CIKM-00). pp. 20-29. ACM Press. NY, USA

47. Young, S. J.; Evermann, G.; Gales, M. J. F.; Hain, Kershaw, T.; D.; Moore, G.; Odell, J.; Ollason, D.; Valtchev, V.; Povey, D.; V.; Woodland, P. C.. 2006. The HTK Book, version 3.4, pp.279. Cambridge University.

48. Young, S. J.; Evermann, G.; Gales, M. J. F.; Hain, Kershaw, T.; D.; Moore, G.; Odell, J.; Ollason, D.; Valtchev, V.; Povey, D.; V.; Woodland, P. C.. 2006. The HTK Book, version 3.4, pp.276. Cambridge University.

49. MADA+TOKAN. http://www1.ccls.columbia.edu/MADA/

50. Diab, Mona. 2009. Second Generation Tools (AMIRA 2.0): Fast and Robust Tokenization, POS tagging, and Base Phrase Chunking. In Proceedings of the Second International Conference on Arabic Language Resources and Tools. Cairo, Egypt.

51. AMIRA. http://www.ccls.columbia.edu/project/amira

52. Bilmes, Jeff and Kirchhoff Katrin. 2003. Factored language models and generalized parallel backoff. In Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of (HLT-NAACL 2003).

53. http://crfpp.googlecode.com/svn/trunk/doc/index.html

# Appendix A

Buckwalter Arabic Transliteration.

| | | | | | |
|---|---|---|---|---|---|
| ء | ' | ذ | * | ل | l |
| آ | \| | ر | r | م | m |
| أ | > | ز | z | ن | n |
| ؤ | & | س | s | ه | h |
| إ | < | ش | $ | و | w |
| ئ | } | ص | S | ى | Y |
| ا | A | ض | D | ي | y |
| ب | b | ط | T | ـً | F |
| ة | p | ظ | Z | ـٌ | N |
| ت | t | ع | E | ـٍ | K |
| ث | v | غ | g | ـَ | a |
| ج | j | ـ | _ | ـُ | u |
| ح | H | ف | f | ـِ | i |
| خ | x | ق | q | ـّ | ~ |
| د | d | ك | k | ـْ | o |

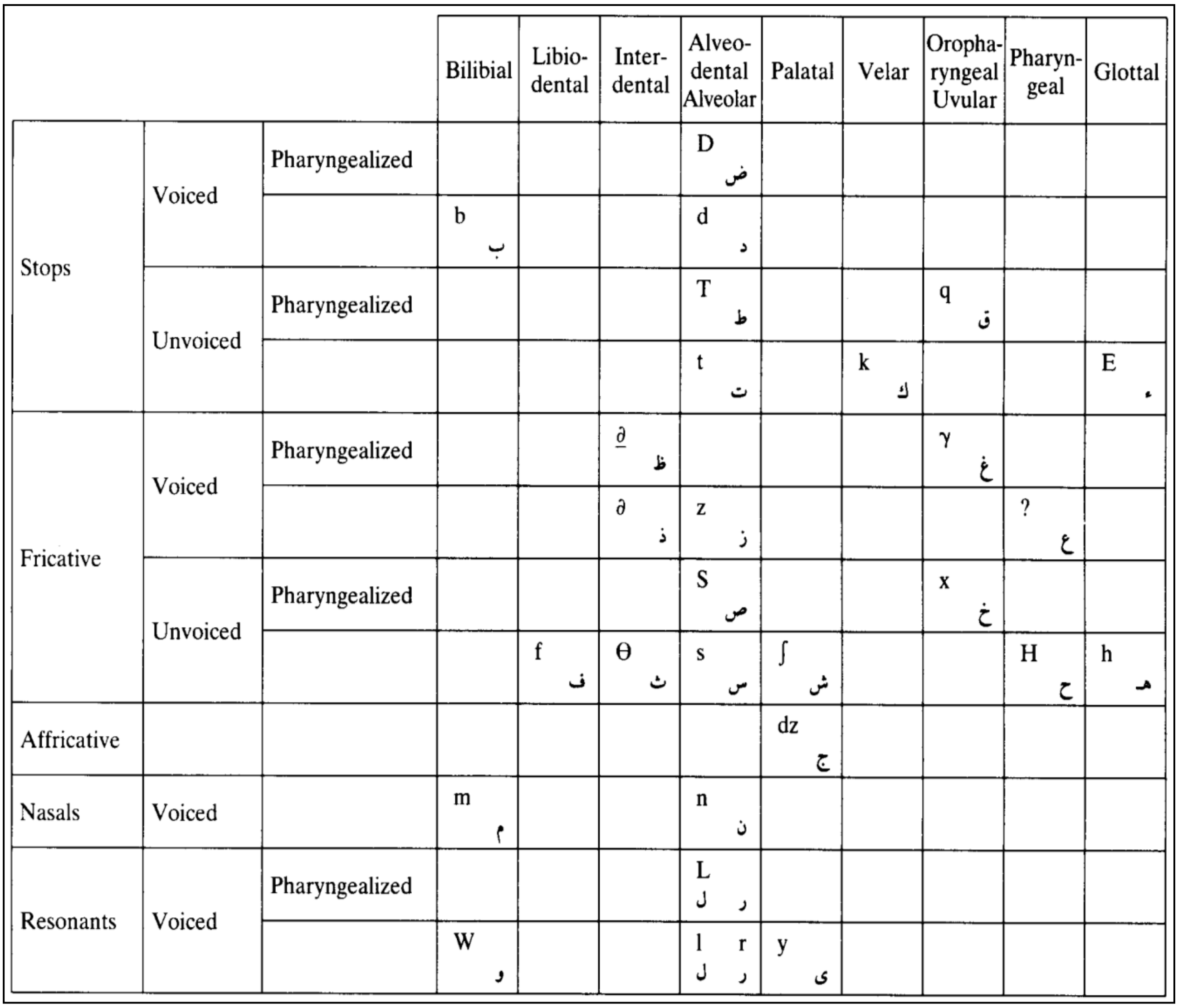
# Appendix B

Template used for CRF++ training:

```
U00:%x[-4,0]
U01:%x[-3,0]
U02:%x[-2,0]
U03:%x[-1,0]
U04:%x[0,0]
U05:%x[1,0]
U06:%x[2,0]
U07:%x[3,0]
U08:%x[4,0]

U09:%x[-1,0]/%x[0,0]
U10:%x[0,0]/%x[1,0]
U11:%x[-2,0]/%x[-1,0]
U12:%x[1,0]/%x[2,0]
U13:%x[-3,0]/%x[-2,0]
U14:%x[2,0]/%x[3,0]
U15:%x[-3,0]/%x[0,0]
U16:%x[0,0]/%x[3,0]
U17:%x[-2,0]/%x[0,0]
U18:%x[0,0]/%x[2,0]
U19:%x[-4,0]/%x[-3,0]
U20:%x[3,0]/%x[4,0]
U21:%x[-4,0]/%x[0,0]
U22:%x[0,0]/%x[4,0]

U23:%x[-1,0]/%x[0,0]/%x[1,0]
U24:%x[-2,0]/%x[-1,0]/%x[0,0]
U25:%x[0,0]/%x[1,0]/%x[2,0]
U26:%x[1,0]/%x[2,0]/%x[3,0]
U27:%x[-3,0]/%x[-2,0]/%x[-1,0]
U28:%x[2,0]/%x[3,0]/%x[4,0]
U29:%x[-4,0]/%x[-3,0]/%x[-2,0]

U30:%x[-1,0]/%x[0,0]/%x[1,0]/%x[2,0]
U31:%x[-2,0]/%x[-1,0]/%x[0,0]/%x[1,0]
U32:%x[0,0]/%x[1,0]/%x[2,0]/%x[3,0]
U33:%x[-3,0]/%x[-2,0]/%x[-1,0]/%x[0,0]

U37:%x[-3,0]/%x[-2,0]/%x[-1,0]/%x[0,0]/%x[1,0]
U38:%x[-2,0]/%x[-1,0]/%x[0,0]/%x[1,0]/%x[2,0]
U39:%x[-1,0]/%x[0,0]/%x[1,0]/%x[2,0]/%x[3,0]
U100:%x[0,1]
U200:%x[0,2]
U300:%x[0,3]
U400:%x[0,4]
U500:%x[0,5]
U600:%x[0,6]
B
```

# Appendix C

Classification of Arabic consonants by M. Elshefai, 1989.



| | | | Bilibial | Libio-dental | Inter-dental | Alveo-dental Alveolar | Palatal | Velar | Oropha-ryngeal Uvular | Pharyn-geal | Glottal |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Stops | Voiced | Pharyngealized | | | | D ض | | | | | |
| | | | b ب | | | d د | | | | | |
| | Unvoiced | Pharyngealized | | | | T ط | | | q ق | | |
| | | | | | | t ت | | k ك | | | E ء |
| Fricative | Voiced | Pharyngealized | | | ∂̲ ظ | | | | ɣ غ | | |
| | | | | | ∂ ذ | z ز | | | | ? ع | |
| | Unvoiced | Pharyngealized | | | | S ص | | | x خ | | |
| | | | | f ف | θ ث | s س | ʃ ش | | | H ح | h هـ |
| Affricative | | | | | | | dz ج | | | | |
| Nasals | Voiced | | m م | | | n ن | | | | | |
| Resonants | Voiced | Pharyngealized | | | | L<br>ل ر | | | | | |
| | | | W و | | | l r<br>ل ر | y ى | | | | |

# Appendix D

Tag Lists.

**MADA:**

verb, noun, adj, noun_prop, prep, noun_num, punc, conj_sub, pron_rel, part_verb, part_neg, conj, abbrev, part_interrog, pron_dem, pron, adv_rel, adv, part_restrict, part_det, part_fut, part, part_focus, interj

**Stanford POS Tagger:**

VBD, DTNN, VBP, NN, DTJJ, IN, WP, NNP, RP, JJ, NNS, NOUN, DTNNS, RB, DTNNP, PRP, CD, CC, VBG, DT, VN, ADJ, VBN, WRB, PRP$, JJR, DTJJR, PUNC, VB, DTNNPS, UH

**AMIRA 2.0 BPC**:

VP, NP, PP, PUNC, SBAR, WHNP, ADJP, WHADVP, PRN, CONJP, ADVP, NAC, S, WHPP, LST

# Appendix E

Values of PRC1, taken from MADA+TOKAN user manual, v3.1.[13]

| **LABEL** | **prc1** | **part** |
|---|---|---|
| No proclitic | 0 | NO |
| Not applicable | na | NA |
| Particle *bi* | bi_part | YES |
| Preposition *bi* | bi_prep | YES |
| Preposition *ka* | ka_prep | YES |
| Emphatic Particle *la* | la_emph | YES |
| Preposition *la* | la_prep | YES |
| Response conditional *la* | la_rc | YES |
| Jussive *li* | li_jus | YES |
| Preposition *li* | li_prep | YES |
| Future marker *sa* | sa_fut | YES |
| Preposition *ta* | ta_prep | YES |
| Particle *wa* | wa_part | YES |
| Preposition *wa* | wa_prep | YES |
| Preposition *fy* | fy_prep | YES |
| Negative particle *lA* | lA_neg | YES |
| Negative particle *mA* | mA_neg | YES |
| Vocative *yA* | yA | YES |
| Vocative *wA* | wA | YES |
| Vocative *hA* | hA | YES |

[13] Available at http://www1.ccls.columbia.edu/MADA/.