# A Hybrid Approach for Building Arabic Diacritizer

**Khaled Shaalan**
The Faculty of Informatics
The British University in Dubai
khaled.shaalan@buid.ac.ae

**Hitham M. Abo Bakr**
Computer & System Dept
Zagazig University
hithamab@yahoo.com

**Ibrahim Ziedan**
Computer & System Dept.
Zagazig University
i.ziedan@yahoo.com

## Abstract

Modern standard Arabic is usually written without diacritics. This makes it difficult for performing Arabic text processing. Diacritization helps clarify the meaning of words and disambiguate any vague spellings or pronunciations, as some Arabic words are spelled the same but differ in meaning. In this paper, we address the issue of adding diacritics to undiacritized Arabic text using a hybrid approach. The approach requires an Arabic lexicon and large corpus of fully diacritized text for training purposes in order to detect diacritics. Case-Ending is treated as a separate post processing task using syntactic information. The hybrid approach relies on lexicon retrieval, bigram, and SVM-statistical prioritized techniques. We present results of an evaluation of the proposed diacritization approach and discuss various modifications for improving the performance of this approach.

## 1 Introduction

Modern Arabic written texts usually include Arabic scripts without short vowels and other diacritic marks. This often leads to considerable ambiguity since several words that have different diacritic patterns may appear identical in a diacritic-less setting. Educated modern Arabic speakers are able to accurately derive/restore diacritics in a document. This is based on the context and their linguistic knowledge of Arabic. However, a text without diacritics brings difficulties for Arabic readers. It is also problematic for Arabic processing applications, such as text-to-speech, speech-to-text, and text analysis, where the lack of diacritics adds another layer of ambiguity when processing the input data. As an example, full vocalization of Arabic text is required for text-to-speech applications, where the mapping from graphemes to phonemes is complicated compared to languages such as English and French; where there is, in most cases, simple one-to-one relationship. Nevertheless, using Arabic text with diacritics has proven an improvement in the accuracy of speech-recognition applications (Zitouni et al., 2006).

The problem of automatic restoration (i.e., derivation) of the diacritic signs of Arabic text can be solved by two approaches. The first is a rule-based approach that involves a complex integration of the Arabic morphological, syntactic, and semantic tools with significant efforts to acquire respective linguistic rules. A morphological analyzer gets the breakdowns of the undiacritized word according to known patterns or templates and recognizes its prefixes and suffixes. A syntax analyzer applies specific syntactic rules to determine the case-ending diacritics, usually, by techniques such as finite-state automata. Semantics handling helps to resolve ambiguous cases and to filter out hypothesis. Hence, rule-based diacritization approach is a complicated process and takes longer time to process an Arabic sentence which is naturally long. The second approach is the statistical approach that requires linguistic resources such as a large tagged corpus (in particular a TreeBank) to extract language statistics for estimating the missing diacritical marks. The approach is fully automated and does not require efforts to acquire respective linguistic knowledge. Results are usually improved by increasing the size of the corpus.

It is worth noting that identifying some of the diacritic marks can be seen as a morphological problem and the relevant letters are called internal characters in this paper. Moreover, diacritic mark of the last character of the Arabic is called case ending (علامة الاعراب). The identification of case-ending diacritics is determined at the syn-

*Proceedings of the EACL 2009 Workshop on Computational Approaches to Semitic Languages*, pages 27–35,
Athens, Greece, 31 March, 2009. 

tactic processing level (case ending depends on the position of the word within the sentence) whereas detecting the internal diacritics is determined at the morphological processing level. In widespread cases, the case-ending come internally rather than with the last character such as "بِقَلَمِها" (by-her-pen).

In this paper, an Arabic diacritizer is proposed. Internal diacritization was restored by a model based on the synergy of three different techniques: retrieval of unambiguous lexicon entries, retrieval of two-word expression from a preprocessed diacritized bigram database, and a prediction using statistical approach based on SVM-learning technique, (Cristianini and Taylor, 2000) and (Hearst, 1998). The later technique tokenizes a text and provides a Reduced Tag Set (RTS) of Part of Speech (POS)[1] for each token. The tags are used to restore the diacritics. From the obtained diacritization results of these techniques, the most consistent one is selected.

The Case-Ending diacritization is treated as a post-process of the internal diacritization task using the same machine learning approach that was trained on Base phrase (BP)-Chunk as well as POS features of individual tokens with correct case-ending tags. A utility has been designed to extract correct case-ending tags from the LDC's Arabic Tree Bank (ATB).

This paper presents a new simple but efficient approach that gets results comparable with the best performing systems, to our knowledge, (Habash and Rambow, 2007). The achieved results are: 11.795% Word Error Rate (WER) and about 3.245% Diacritics Error Rate (DER). The paper is structured as follows. Section 2 reviews closely related work. Section 3 introduces the proposed diacritization approach. Section 4 describes the training process. Section 5 presents the evaluation experiment. Section 6 concludes the article and gives direction for future research.

## 2 Related Work

Diacritic restoration has been receiving increasing attention and has been the focus of several studies. In El-Sadany and Hashish (1988), a rule-based approach that uses morphological analyzer for vowelization was proposed. Another, rule-based grapheme to sound conversion approach appeared in 2003 by Y. El-Imam (2003).
There are many related works dealing with the problem of Arabic diacritization in general (Zitouni et al., 2006), (Habash and Rambow, 2007), (Ananthakrishnan, 2005), (Kirchhoff, 2005). and (Elshafei et al, 2006); all trying to handle this problem using statistical approaches but they tend to handle the case ending diacritic mark in the same way they used to handle the internal (any letter but the last) diacritics. In our proposed approach we differentiate between them as the detection of case-ending diacritics is a syntactic-based problem whereas detecting the internal diacritics is a morphological-based problem. Habash et al. (2007) introduced a system called MADA-D that uses Buckwalter's Arabic morphological analyzer where they used 14 taggers and a lexeme-based language model. MADA is so far the best performing system to date. It has been reported that it achieved a WER of 14.9% and a DER of 4.8%.

## 3 The Proposed Diacritization Approach

The Arabic internal diacritization problem will be addressed from three different proposed techniques, each of which has its own strengths and weaknesses. Such techniques are integrated to optimize the performance of the Arabic diacritizer and to a large extent remove ambiguities. These proposed techniques are: 1) Lexicon Retrieval, 2) diacritized bigram, and 3) SVM-statistical-based diacritizer. Then, the case ending diacritization will be determined after the internal discrimination is performed. Figure 1 shows the architecture of Arabic Diacritization System.

[1] List of POS and RTS that are used here can be found at: http://www.ircs.upenn.edu/arabic/Jan03release/arabic-POStags-collapse-to-PennPOStags.txt

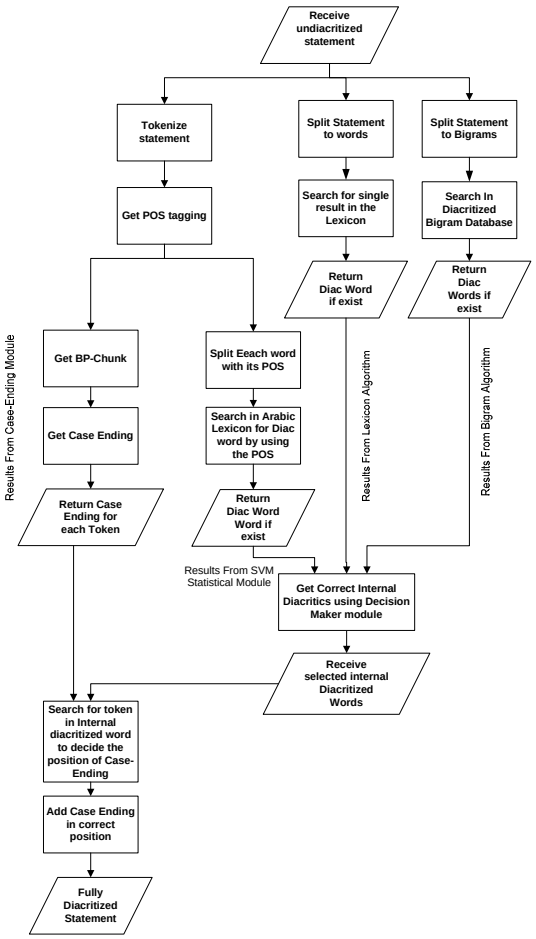


Figure 1: Arabic Diacritization System

***Lexicon Retrieval Technique (LR)***

Lexicon retrieval approach tries to find the result (diacritized word) returned from an Arabic lexicon for a specific input undiacritized word. If only one diacritization is returned, then there is no ambiguity. This solution is final and we do not need to look at the results form the other two techniques. However, this situation is usually rare but when it occurs the result is confirmed.

***Diacritized Bigram Technique (DB)***

When more than one solution is retrieved for an unvowelized input word, i.e., ambiguous diacrticazation, the bigram technique comes into play. The idea behind this technique is to make use of the multiword expressions in Arabic texts. When such expressions are analyzed as separate words, the possibility for ambiguity is increased. In this work, we considered a two-word expression (bigram) that usually occurs with high frequency in Arabic texts such that one word can determine the diacritization of the other. Once the expression is identified and diacritized correctly, it adds a sense of certitude to the diacritization which significantly reduces the ambiguity. Table 1 shows an extraction of the diacritized bigram database.

| Diac. 2nd Word | Diac. 1st Word | Cat | 2nd Word | 1st Word |
|---|---|---|---|---|
| القَدَم | لِكُرَة | 3 | القدم | لكرة |
| المُتَّحِدَة | الولايات | 1 | المتحدة | الولايات |
| الوُزَراء | رَئِيس | 1 | الوزراء | رئيس |
| برس | فرانْس | 1 | برس | فرانس |
| الغَرْبِيَّة | الضِفَّة | 1 | الغربية | الضفة |

Table 1: Diacritized Bigram Database

***SVM-Statistical Technique (SVM)***

The previous two diacritization techniques can be viewed as a lookup process; either for a word in the lexicon or for a two-word expression in a large bigram database. However, statistical methods can be viewed as general approaches because they are heavily dependent on the Arabic syntactic analysis that was manually performed by Arabic specialists.

The main idea of this approach is to tokenize and automatically annotate tokens with the correct POS tags. Then, by searching the Arabic lexicon using a token and the corresponding POS, the correct diacritization result can reached, even though multiple ambiguous words are retrieved from the lexicon.

Buckwalter's morphological analyzer (Buckwalter, 2002) takes an inflected Arabic word and returns fully diacritized ambiguous words. We claim in our approach that only internal diacritics should be handled morphologically whereas case ending should be handled syntactically. Hence, we have used the Buckwalter's morphological analyzer after removing all case ending diacritics from the suffixes table in order to prevent the generation of the case ending output. One advantage of this modification is to considerably reduce the number of alternatives (i.e., overgenerations) returned from the morphological analyzer. Another advantage is that some NLP tasks, such as Information Retrieval, require only diacritic restoration of internal (lexical) vowels which can benefit from such modification. For example, given the word " عامل " to this morphological analyzer, it returns 7 results that have the same internal diacritics with one having no case-ending and 6 having different case-ending diacritics. Consequently, splitting the diacrization into two stages (internal and case ending) will avoid such morphological ambiguity and at the second stage the syntactic case ending is treated

separately as a post processing which ultimately leads to a fully efficient diacritized Arabic word.

***A Hybrid of All Internal Techniques***

When we apply each of the three proposed techniques on an input undiacritized Arabic sentence we may get different diacritization results for each word within this sentence. The selection criteria depend on the agreement among these techniques. Two or more matched results can determine the discrimination of a word. In case of disagreement, a priority is applied in the following, highest to lowest, order: lexicon retrieval, bigram and SVM-Statistical technique respectively. If no solution is reached from all techniques, the undiacritized input word is returned.

***Case Ending Model***

The main idea is to relate the case-ending for each token with its POS and chunk position as well as its position within the sentence (Abo Bakr et al., 2008). We made a training using Support Vector Machines (SVM) technique with undiacritized tokens. This technique involves an Arabic Treebank.

An Arabic Treebank usually created on top of a corpus that has already been annotated with POS tags. We have used the Penn Arabic Treebank (ATB) (Maamouri et al, 2004). ATB has begun in the fall of 2001 and has now completed four full releases of morphologically and syntactically annotated data: Version 1 of the ATB has three parts with different releases; some versions like Part 1 V3.0 and Part 2 V 2.0 are fully diacritized trees. For example, consider the following undiacritized statement:

"لليوم الثاني على التوالي تظاهر طلاب ينتمون الى جماعة...."

"llvwm AlvAnv ElY AltwAlv tZAhr TlAb

The following tree representation is partially extracted from the tree fileU-MAAH_UM.ARB_20020120-a.0007.tree that is part of the ATB Part 2 V.2.

```
(S (S (S (PP-TMP (PREP li-) (NP (NP
(DET+NOUN+CASE_DEF_GEN -Al+yawom+i)
(DET+ADJ Al+vAniy)) (PP (PREP EalaY) (NP
(DET+NOUN Al+tawAliy))))) (VP
(VERB_PERFECT+PVSUFF_SUBJ:3MS N
Al+musolim+iyona) …..
```

Figure 2 shows a graphical representation of this tree[2]. Case-ending is indicated, ovals in Figure 2, by one of the following tags: NCE, CASE_DEF_GEN, CASE_INDEF_GEN, CASE_DEF_NOM, CASE_DEF_ACC, CASE_INDEF_NOM, CASE_DEF_ACCGEN, CASE_INDEF ACC, and CASE_INDEF_ACCGEN.

Table 2 gives the complete description of these tags.

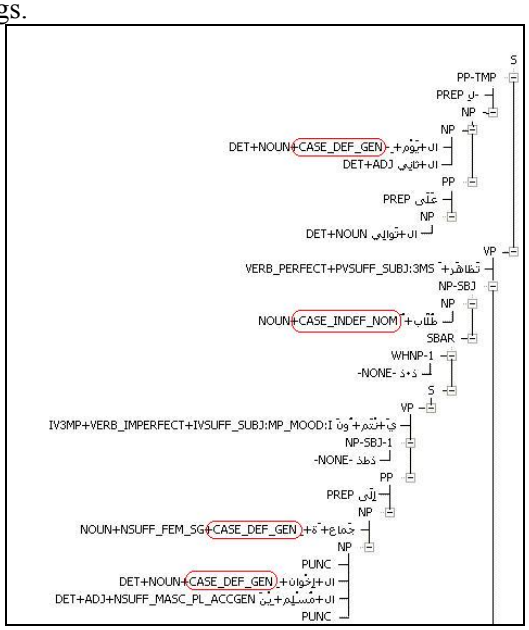


Figure 2: A graphical representation of an Arabic sentence extracted from the Penn Arabic Treebank

| Case Ending Tags | Description |
|---|---|
| NCE | No Case Ending |
| CASE_DEF_GEN | Kasra ـِ |
| CASE_INDEF_GEN | kasratan ـٍ |
| CASE_DEF_NOM | Damma ـُ |
| CASE_DEF_ACC | Fat-ha ـَ |
| CASE_DEF_ACCGEN | Maftouh bi Kasra ـِ |
| CASE_INDEF_NOM | Damatan ـٌ |
| CASE_INDEF_ACCGEN | Fathatan ـً or ـٍ |
| CASE_INDEF_ACC | Fathatan ـً |

Table 2: Description of Case-Ending tags found in ATB

A sequence of tokens with its POS, BP-chunk and Case-Ending is extracted from Treebank using YamCha File Creator (YFC utility[3]). The

[2] This graphical representation of the Treebank files is extracted from our Treebank Viewer tool that is freely available at: http://www.staff.zu.edu.eg/hmabobakr/

[3] We developed YFC utility to extract information from Penn Arabic Treebank ATB and produce the Yamcha standard input format to be able to use this information in the training process.
http://www.staff.zu.edu.eg/hmabobakr/page.asp?id=53

basic approach used in YFC is inspired by the work of Sabine for Treebank-to-chuck conversion script (Sang and Buchholz, 2000), which we have extended to be used with Arabic. This has required adding some features like Case-Ending. The output produced from YFC utility for case ending training process is shown in Table 3.

| Token | POS | Chunk | Case Ending |
|---|---|---|---|
| L | IN | B-PP | NCE |
| Al | DT | B-NP | NCE |
| ywm | NN | I-NP | CASE_DEF_GEN |
| Al | DT | I-NP | NCE |
| vAny | JJ | I-NP | NCE |
| ElY | IN | B-PP | NCE |
| Al | DT | B-NP | NCE |
| twAly | NN | I-NP | NCE |
| tZAhr | VBD | B-VP | NCE |
| TlAb | NN | B-NP | CASE_INDEF_NOM |
| Yntmwn | VBP | B-VP | NCE |
| <lY | IN | B-PP | NCE |
| jmAEp | NN | B-NP | CASE_DEF_GEN |

Table 3: Training file format for detecting Case-Ending

## 4 Training of the Arabic Diacritizer

The diacritization system we present here is trained and evaluated on the LDC's Arabic Treebank of diacritized news articles – Part 2 v2.0: catalog number LDC2004T02 and 1-58563-282-1. The corpus includes complete vocalization (including case endings). We introduce here a clearly defined and replicable split of the corpus, so that the reproduction of the results or future investigations can accurately and correctly be established. This corpus includes 501 stories from the Ummah Arabic News Text. There are a total of 144,199 words (counting non-Arabic tokens such as numbers and punctuation) in the 501 files - one story per file. We split the corpus into two sets: training data and development test (devtest) data. The devtest data are the files ended by character "7" like "UMAAH_UM.ARB_20020120-a.0007.tree" and its count was 38 files. The remaining files are used for training.

## 5 Evaluation

For Arabic tokenizer, POS tagger, BP-chunk, and statistical Case-Ending, we used a standard SVM with a polynomial kernel of degree 2 and C=1.0. Evaluation of the system was done by calculating the performance using the standard evaluation measures: accuracy, precision, recall, and the f-measure[4].We used YamCha (Kudo and Matsumoto, 2003) implementation of SVMs. Diacritization evaluation of our experiments is reported in terms of word error rate (WER), and diacritization error rate (DER)[5].

We conducted experiments to:

1. Evaluate the impact of tokenization, part-of-speech, chunking, and case-ending parameters on the training models, see Section 5.1.
2. Evaluate the impact of including and excluding the case-ending on the performance of the Arabic diacritizer, see Section 5.2.
3. Compare our approach of Tokenization and POS tagger with the ArabicSVMTools tagger using different parameters and feature(s), see Section 5.2.

### 5.1 Results of Tokenization, Part-of-Speech, BP-chunking, and case-ending

The results obtained for tokenization (TOK), part-of-speech (POS), and Chunking (BP-chunk) tasks are comparable with the results presented in the most notable literature (Diab et al, 2007; Diab et al, 2004). We did some modifications of the feature list to compromise between the speed and accuracy. The case ending task is novel, and did not get enough handling in other research. It achieved acceptable results.

***Evaluation of the impact of the tokenization parameter on the training process***

Two tokenization tasks was performed on window sizes of -2 /+2 and -4/+4, for illustration see TOK1 and TOK2 tasks in Figure 3. For each window size there are two columns. The first one contains a sequence of Buckwalter's transliterated Arabic letters shown from top to bottom that resembles the left-to-right Arabic writing system (e.g., ….wyblg Eddhm ….. are the transliteration of the Arabic words ...ويبلغ عددهم..., respectively). The second column contains the corresponding tokenization tags presented by **I**nside-**O**utside-**B**eginning (I-O-B) of a chunk, i.e.,

[4] These results were computed using our developed evluation tool that was developed and tested against Evaluation Tools for CONLL 2000 http://www.cnts.ua.ac.be/conll2000/chunking/conlleval.txt.

[5] These results were computed using our developed evaluation tool that was developed based on information presented in (Habash and Rambow, 2007).

prefix (PRE), word (WRD), and suffix (SUFF), respectively, (Kudo and Matsumoto, 2003). The tokenization tags are: B-PRE1, I-PRE1, B-PRE2, I-PRE2, B-PRE3, I-PRE3, B-WORD-1, I-WORD-1, B-SUFF1, I-SUFF1 and O for outside word boundary. We made segmentation for the determiner "Al" – "ال". This segmentation is important for the case-ending detection for: the adjective and the noun it modifies "الصفة والموصوف", 1st and 2nd Particle of the construction Annexed and Annexed noun "المضاف و المضاف إليه", and Nunation "التنوين ". The result of the evaluation of the two tokenization tasks is shown in Table 4.

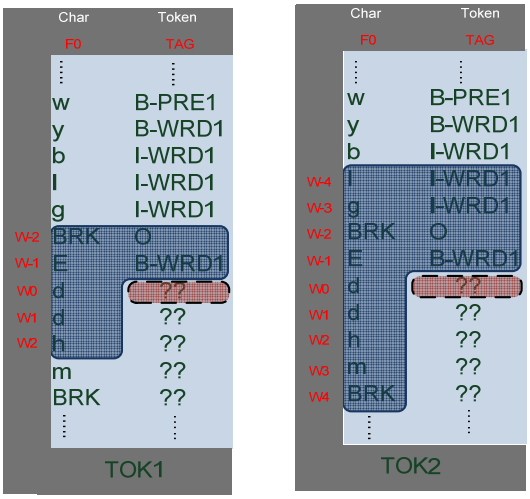


Figure 3: Tokenization evaluation with window sizes of -2/+2 and -4/+4

| Measurement | TOK1 | TOK2 |
|---|---|---|
| Accuracy | 98.59% | 99.56% |
| Precision | 97.17% | 98.95% |
| Recall | 97.29% | 99.06% |
| F-Measure | 97.23% | 99.00% |

Table 4: Tokenization results with window sizes of -2/+2 and -4/+4

### *Evaluation of the impact of the part-of-speech parameter on the training process*

A POS tagging (POS1) task was performed on a sequence of tokens produced from the tokenization task. A window size of +2/-2 tokens centered at the focus token. We made another POS tagging (POS2) task by adding the last two characters as an extra feature for enhancing the accuracy of some tags such as plural or dual noun (NNS) and singular noun (NN). For illustration see POS1 and POS2 tasks in Figure 4. The result of the evaluation of the two POS tagging tasks is shown in Table 5.

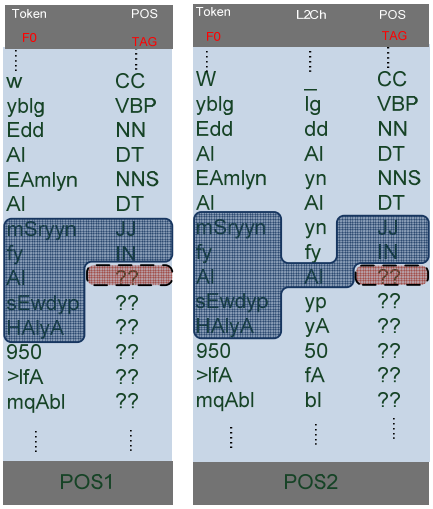


Figure 4: POS evaluations with window size of -2/+2; with and without using the last two characters as an added feature

| Measurement | POS1 | POS2 |
|---|---|---|
| Accuracy | 94.34% | 95.97% |

Table 5: POS results for different window sizes

### *Evaluation of the impact of chunking parameters on the training process*

The chunking task was performed on tokens produced from the tokenization and POS tasks. The evaluation included 16 tag-set (features) of a window size of -2/+2 for both tokens and POS, and only the previous two chunk tags. For illustration see Figure 5. The result of the evaluation of is shown in Table 6.

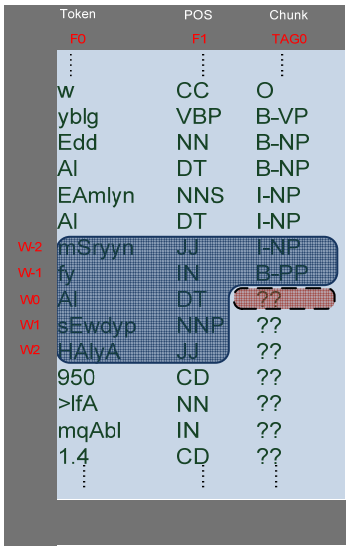


Figure 5: Chunk evaluation with window size of -2/+2

| Measurement | Results |
|---|---|
| Accuracy | 95.52% |
| Precision | 93.19% |
| Recall | 95.90% |
| F-Measure | 94.52% |

Table 6: Results for BP-chunk

***Evaluation of the impact case-ending parameters on the training process***

Two case-ending tasks were performed. The first case-ending (CE1) task was discussed in a previous work (Abo Bakr et al., 2008). It was performed on window size of -3/+3 and 8 tag sets. For illustration see Figure 6.

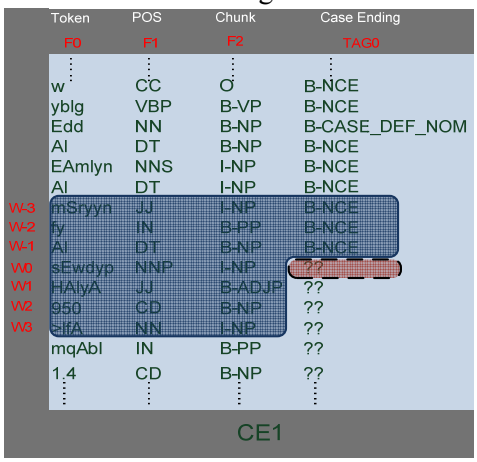


Figure 6: Case-ending evaluation with window size of -3/+3

The evaluation has achieved 95.35% in accuracy. We noticed that in some cases the system can produce unacceptable case ending (e.g., Tanween on the sound plural masculine " جمع المذكر السالم") that we could improved by:

1- Enhancing the POS tagging (POS2) task by adding last two characters (L2Ch) as a feature.
2- Enhancing the case ending (CE2) task by adding the last character (LCh) and the last two characters (L2Ch) as features.

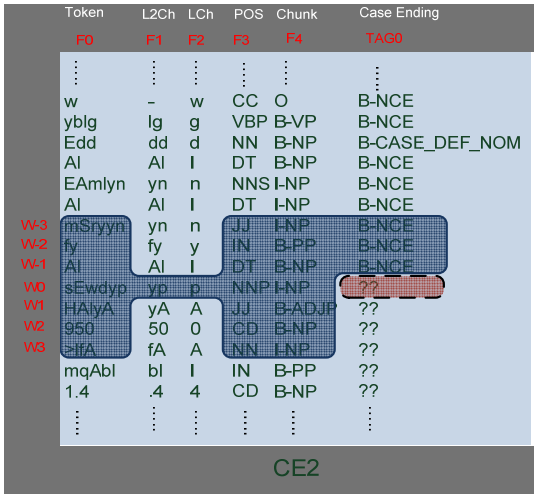


Figure 7: Case-Ending evaluation with widow size of -3/3 and using the last two characters (L2Ch) and the last character (LCh) as added features

The following modifications were done to conduct the second case-ending (CE2) task, for illustration see Figure 7:

- Adding the last two characters (L2Ch) and the last character (LCh) as features.
- Enhancing the case ending representation by adding an extra tagset for "indeclension of the fatha" - "مبني على الفتح" that is presented in Treebank as "PVSUFF_SUNJ:3MS".

Table 7 presents the results obtained for the two case ending (CE1 and CE2) tasks. As shown, the performance is improved.

| Measurement | CE1 | CE2 |
|---|---|---|
| Accuracy | 95.35% | 96.57% |

Table 7: Results of Case Ending evaluation

## 5.2 Diacritization Results

In this section, we compare our approach of Tokenization and POS tagger with ArabicSVMTools tagger. We evaluate the impact of including and excluding different techniques of internal diacritization and case-ending on the overall performance of our Arabic diacritizer. In particular, we show the results from the following techniques: lexicon retrieval (LR), diacritized bigram (DB), SVM, and case-ending (CE), techniques. Results for different combinations were reported and compared. All results were performed using TOK1, POS1, and CE1 tasks and shown in Table 8 through Table 10.

| | Including CE | | Excluding CE[6] | |
|---|---|---|---|---|
| **Technique** | **WER** | **DER** | **WER** | **DER** |
| LR | 90.35% | 40.85% | 31.38% | 36.67% |
| SVM | 69.94% | 23.36% | 16.28% | 11.36% |

Table 8: WER and DER for Lexicon Retrieval and Statistical SVM techniques for including and excluding case ending

Table 8 shows that excluding case ending (letter) from the evaluation gives better results in terms of WER and DER.

As shown in Table 9, it is noted that including the case ending technique has enhanced dramatically the results of diacritic restoration. Further enhancement was obtained by adopting a new method to restore internal diacritics, when all of the hybrid techniques fail to return any solution; the new method, we call it "accepts any" (AA),

[6] Results for "Excluding CE" are calculated manually for a limited number of test files because Case-Ending diacritic is not always at the last character.

is used for arbitrary accepting results from lexicon.

| Technique | WER | DER |
|---|---|---|
| LR+DB | 35.81% | 9.77% |
| LR+DB+SVM | 33.51% | 7.99% |
| **LR+DB+SVM+CE** | **17.31%** | **4.41%** |
| LR+DB+SVM+CE+AA | 16.66% | 3.84% |

Table 9: WER and DER for different combination of diacritization techniques

To investigate the effect of enhancing POS tagging on the internal SVM statistical technique, we adapted our modules to interact with ArabicSVMTools, the up-to-date most famous free tagger[7]. Some modification were made to our module to accept the article 'Al' as it may occur as radical letters inside the Noun (we handle 'Al' separately in our tokenizer). We evaluated our statistical diacritization approach using ArabicSVMTools and our proposed tagger. The use of ArabicSVMTools has improved the performance of our diacrtizer as shown in Table 10. ArabicSVMTools gave better results than our proposed tagger. However, our proposed tagger is about 4 times faster than ArabicSVMTools because we use less features.

| Tagger | WER | DER |
|---|---|---|
| ArabicSVMTools | 12.79% | 9.94% |
| Proposed SVM | 16.28% | 11.36% |

Table 10: WER and DER for statistical approach using different taggers without considering case-ending diacritics.

Table 11, shows the results after modifying both the statistical and the case ending approaches for TOK2, POS2, and CE2 tasks. The last row represent results after adding some simple heuristic rules (SHR) to correctly add Tanween Kasra instead of Tanween el Fatha in case of sound plural feminine "جمع المؤنث السالم".

| Technique | WER | DER |
|---|---|---|
| LR+DB+SVM | 31.86% | 7.92% |
| **LS+DB+SVM+CE** | **12.16%** | **3.78%** |
| LS+DB+SVM+CE+SHR | 11.795% | 3.245% |

Table 11: WER and DER for different techniques

[7] ArabicSVMTools: http://www.cs.columbia.edu/~mdiab/downloads/ArabicSVMTools.tar.gz

## 6 Conclusions and Future work

In this paper, we proposed a diacritization model that distinguishes between internal and case ending diacritization. The overall performance is comparable with the best diacritization model that was reported in the literature so far.

Statistically based methods show great promise in addressing the ambiguity resolution problem in Arabic language diacritization.

The proposed system yields good results in the DER and WER compared with MADA-D system, the modifications for case ending algorithm have enhanced the performance.

The proposed system has an advantage that we can use all internal diacritics approaches in parallel because there is no such dependency between algorithms. Nevertheless, the case ending algorithm can also be processed in parallel with the statistical approach. Such parallel processing advantage can improve the response time that could be critical for some diacritization-based real time systems.

Maintaining the bigram database up-to-date will significantly enhance the performance of the system.

Our future work will include adding some heuristic rules for the proposed model as a post processing. This will enhance the performance for the system especially to restore correct diacritics of the possessive personal pronounce suffixes "ها"،"نا". Moreover, adding extra POS tag sets to distinguish between dual noun and plural nouns will enhance the diacritization results. We plan also to enrich the system by increasing the training set by using latest fully diacritized Treebank like Part1 V3.0 (Maamouri et al, 2008) which is not available due to limitation of our budget. This has the effect of enhancing the system performance and allow us to make a comparison with other systems, such as (Habash and Rambow, 2007) and (Zitouni et al. , 2006) .

## References


Abo Bakr H. M. , Shaalan K., Ziedan I., 2008, "A Statistical Method for Adding Case Ending Diacritics for Arabic Text", The Eighth Conference on Language Engineering, ESOLEC'2008, Page 225-234, Cairo, Egypt,Deceber 17-18 2008.

Ananthakrishnan, Narayanan S., and Bangalore S., (2005), "Automatic diacritization of arabic transcripts for asr". In Proceedings of ICON-05, Kanpur, India.

Buckwalter T., (2002). Buckwalter Arabic morphological analyzer version 1.0. Technical report, Lin-

guistic Data Consortium, LDC2002L49 and ISBN 1-58563- 257-0.

Cristianini N. and Taylor J.S., (2000), “An Introduction to Support Vector Machines and Other Kernel-based Learning Methods”, The Press Syndicate of the University of Cambridge, Cambridge, United Kingdom.

Diab M., Hacioglu K., and Jurafsky D., (2004), "Automatic Tagging of Arabic Text: From Raw Text to Base Phrase Chunks," In Proc. of HLT/NAACL 2004, Boston.

Diab M., Hacioglu K., and Jurafsky D.,(2007), “Arabic Computational Morphology Knowledge-based and Empirical Methods” - Chapter 7 “Automatic Processing of Modern Standard Arabic Text”,ISBN: 978-1-4020-6046-5, SpringerLink.

El-Imam Y., (2003). Phonetization of Arabic: rules and algorithms. Computer Speech and Language, 18:339– 373.

El-Sadany T. and Hashish M., (1988). Semi-automatic vowelization of Arabic verbs. In 10th NC Conference, Jeddah, Saudi Arabia.

Elshafei M., Al-Muhtaseb H., and Alghamdi M., (2006), “Statistical Methods for Automatic Diacritization of Arabic Text”. The Saudi 18th National Computer Conference. Riyadh. 18: 301-306.

Emam O. and Fisher V. (2004)., A hierarchical approach for the statistical vowelization of Arabic text. Technical report, IBM patent filed, DE9-2004-0006, US patent application US2005/0192809 A1.

Gal Y., (2002). An HMM approach to vowel restoration in Arabic and Hebrew. In ACL-02 Workshop on Computational Approaches to Semitic Languages.

Habash N. and Rambow O., (2007), “Arabic Diacritization through Full Morphological Tagging”, In Proceedings of the North American chapter of the Association for Computational Linguistics (NAACL), Rochester, New York.

Hearst M. A., (1998), "Support Vector Machines," IEEE Intelligent Systems, vol. 13, no. 4, pp. 18-28, Jul/Aug, 1998.

Kirchhoff K. and Vergyri D., (2005). Cross-dialectal data sharing for acoustic modeling in Arabic speech recognition. Speech Communication, 46(1):37–51, May.

Kudo T. and Matsumoto Y., (2003), " Fast methods for kernel-based text analysis," In Proceedings of the 41st Annual Meeting on Association For Computational Linguistics - Volume 1 (Sapporo, Japan, July 07 - 12, 2003). Annual Meeting of the ACL. Association for Computational Linguistics, Morristown.

Maamouri, M., Bies, A. & Buckwalter, T. (2004). The Penn Arabic treebank: Building a largescale annotated Arabic corpus. In NEMLAR Conference on Arabic Language Resources and Tools, Cairo, Egypt.

Maamouri M., Bies A., Kulick S., (2008), "Enhanced Annotation and Parsing of the Arabic Treebank"; INFOS 2008, Cairo, Egypt, March 27-29, 2008.

Sang E. and Buchholz S., (2000), ” Introduction to the CoNLL-2000 Shared Task: Chunking”, Proceeding of CoNLL-2000 and LLL-2000,Page 127-132, Lisbon,Portugal.

Zitouni I., Sorensen J. S., and Sarikaya R., (2006), “Maximum entropy based restoration of Arabic diacritics”. In Proceedings of ACL’06.