Maximum Entropy Based Restoration of Arabic Diacritics

Imed Zitouni, Jeffrey S. Sorensen, Ruhi Sarikaya
IBM T.J. Watson Research Center
1101 Kitchawan Rd, Yorktown Heights, NY 10598
{izitouni, sorenj, sarikaya}@us.ibm.com

Abstract

Short vowels and other diacritics are not part of written Arabic scripts. Exceptions are made for important political and religious texts and in scripts for beginning students of Arabic. Script without diacritics have considerable ambiguity because many words with different diacritic patterns appear identical in a diacritic-less setting. We propose in this paper a maximum entropy approach for restoring diacritics in a document. The approach can easily integrate and make effective use of diverse types of information; the model we propose integrates a wide array of lexical, segment-based and *part-of-speech* tag features. The combination of these feature types leads to a state-of-the-art diacritization model. Using a publicly available corpus (LDC's Arabic Treebank Part 3), we achieve a diacritic error rate of 5.1%, a segment error rate 8.5%, and a word error rate of 17.3%. In case-ending-less setting, we obtain a diacritic error rate of 2.2%, a segment error rate 4.0%, and a word error rate of 7.2%.

1 Introduction

Modern Arabic written texts are composed of 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 restore diacritics in a document. This is based on the context and their knowledge of the grammar and the lexicon of Arabic. However, a text without diacritics becomes a source of confusion for beginning readers and people with learning disabilities. A text without diacritics is also problematic for applications such as text-to-speech or speech-to-text, where the lack of diacritics adds another layer of ambiguity when processing the data. As an example, full vocalization of text is required for text-to-speech applications, where the mapping from graphemes to phonemes is simple compared to languages such as English and French; where there is, in most cases, one-to-one relationship. Also, using data with diacritics shows an improvement in the accuracy of speech-recognition applications (Afify et al., 2004). Currently, text-to-speech, speech-to-text, and other applications use data where diacritics are placed manually, which is a tedious and time consuming excercise. A diacritization system that restores the diacritics of scripts, i.e. supply the full diacritical markings, would be of interest to these applications. It also would greatly benefit nonnative speakers, sufferers of dyslexia and could assist in restoring diacritics of children's and poetry books, a task that is currently done manually.

We propose in this paper a statistical approach that restores diacritics in a text document. The proposed approach is based on the maximum entropy framework where several diverse sources of information are employed. The model implicitly learns the correlation between these types of information and the output diacritics.

In the next section, we present the set of diacritics to be restored and the ambiguity we face when processing a non-diacritized text. Section 3 gives a brief summary of previous related works. Section 4 presents our diacritization model; we explain the training and decoding process as well as the different feature categories employed to restore the diacritics. Section 5 describes a clearly defined and replicable split of the LDC's Arabic Treebank Part 3 corpus, used to built and evaluate the system, so that the reproduction of the results and future comparison can accurately be established. Section 6 presents the experimental results. Section 7 reports a comparison of our approach to the finite state machine modeling technique that showed promissing results in (Nelken and Shieber, 2005). Finally, section 8 concludes the paper and discusses future directions.

2 Arabic Diacritics

The Arabic alphabet consists of 28 letters that can be extended to a set of 90 by additional shapes, marks, and vowels (Tayli and Al-Salamah, 1990). The 28 letters represent the consonants and long vowels such as ـا, ـَى (both pronounced as /a:/), ـي (pronounced as /i:/), and ـو (pronounced as /u:/). Long vowels are constructed by combining ـا, ـَى, ـي, and ـو with the short vowels. The short vowels and certain other phonetic information such as consonant doubling (shadda) are not represented by letters, but by diacritics. A diacritic is a short stroke placed above or below the consonant. Table 1 shows the complete set of Arabic diacritics. We split the Arabic diacritics into three sets: short vowels, doubled case endings, and syllabification marks. Short vowels are written as symbols either above or below the letter in text with diacritics, and dropped all together in text without diacritics. We find three short vowels:

Diacritic on ة	**Name**	**Meaning/ Pronunciation**
Short vowels		
ةَ	fatha	/a/
ةُ	damma	/u/
ةِ	kasra	/i/
Doubled case ending ("tanween")		
ةً	tanween al-fatha	/an/
ةٌ	tanween al-damma	/un/
ةٍ	tanween al-kasra	/in/
Syllabification marks		
ةّ	shadda	consonant doubling
ةْ	sukuun	vowel absence

Table 1: Arabic diacritics on the letter – consonant – ة (pronounced as */t/*).

- fatha: it represents the /a/ sound and is an oblique dash over a consonant as in ةَ (c.f. fourth row of Table 1).
- damma: it represents the /u/ sound and is a loop over a consonant that resembles the shape of a comma (c.f. fifth row of Table 1).
- kasra: it represents the /i/ sound and is an oblique dash under a consonant (c.f. sixth row of Table 1).

The doubled case ending diacritics are vowels used at the end of the words to mark case distinction, which can be considered as a double short vowels; the term "tanween" is used to express this phenomenon. Similar to short vowels, there are three different diacritics for tanween: tanween al-fatha, tanween al-damma, and tanween al-kasra. They are placed on the last letter of the word and have the phonetic effect of placing an "N" at the end of the word. Text with diacritics contains also two syllabification marks:

- *shadda*: it is a gemination mark placed above the Arabic letters as in ةّ. It denotes the doubling of the consonant. The shadda is usually combined with a short vowel such as in ةَّ.
- *sukuun*: written as a small circle as in ةْ. It is used to indicate that the letter doesn't contain vowels.

Figure 1 shows an Arabic sentence transcribed with and without diacritics. In modern Arabic, writing scripts without diacritics is the most natural way. Because many words with different vowel patterns may appear identical in a diacritic-less setting, considerable ambiguity exists at the word level. The word كتب, for example, has 21 possible forms that have valid interpretations when adding diacritics (Kirchhoff and Vergyri, 2005). It may have the interpretation of the verb "to write" in كَتَبَ (pronounced /kataba/). Also, it can be interpreted as "books" in the noun form كُتُبٌ (pronounced /kutubun/). A study made by (Debili et al., 2002) shows that there is an average of 11.6 possible diacritizations for every non-diacritized word when analyzing a text of 23,000 script forms.

كتب الرئيس المذكرة.

كَتَبَ الرَّئِيسُ المُذَكِّرَةَ.

Figure 1: The same Arabic sentence without (upper row) and with (lower row) diacritics. The English translation is "*the president wrote the document.*"

Arabic diacritic restoration is a non-trivial task as expressed in (El-Imam, 2003). Native speakers of Arabic are able, in most cases, to accurately vocalize words in text based on their context, the speaker's knowledge of the grammar, and the lexicon of Arabic. Our goal is to convert knowledge used by native speakers into features and incorporate them into a maximum entropy model. We assume that the input text does not contain any diacritics.

3 Previous 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 method that uses morphological analyzer for vowelization was proposed. Another, rule-based grapheme to sound conversion approach was appeared in 2003 by Y. El-Imam (El-Imam, 2003). The main drawbacks of these rule based methods is that it is difficult to maintain the rules up-to-date and extend them to other Arabic dialects. Also, new rules are required due to the changing nature of any “living” language.

More recently, there have been several new studies that use alternative approaches for the diacritization problem. In (Emam and Fisher, 2004) an example based hierarchical top-down approach is proposed. First, the training data is searched hierarchically for a matching sentence. If there is a matching sentence, the whole utterance is used. Otherwise they search for matching phrases, then words to restore diacritics. If there is no match at all, character n-gram models are used to diacritize each word in the utterance.

In (Vergyri and Kirchhoff, 2004), diacritics in conversational Arabic are restored by combining morphological and contextual information with an acoustic signal. Diacritization is treated as an unsupervised tagging problem where each word is tagged as one of the many possible forms provided by the Buckwalter's morphological analyzer (Buckwalter, 2002). The Expectation Maximization (EM) algorithm is used to learn the tag sequences.

Y. Gal in (Gal, 2002) used a HMM-based diacritization approach. This method is a white-space delimited word based approach that restores only vowels (a subset of all diacritics).

Most recently, a weighted finite state machine based algorithm is proposed (Nelken and Shieber, 2005). This method employs characters and larger morphological units in addition to words. Among all the previous studies this one is more sophisticated in terms of integrating multiple information sources and formulating the problem as a search task within a unified framework. This approach also shows competitive results in terms of accuracy when compared to previous studies. In their algorithm, a character based generative diacritization scheme is enabled only for words that do not occur in the training data. It is not clearly stated in the paper whether their method predict the diacritics shedda and sukuun.

Even though the methods proposed for diacritic restoration have been maturing and improving over time, they are still limited in terms of coverage and accuracy. In the approach we present in this paper, we propose to restore the most comprehensive list of the diacritics that are used in any Arabic text. Our method differs from the previous approaches in the way the diacritization problem is formulated and because multiple information sources are integrated. We view the diacritic restoration problem as *sequence classification*, where given a sequence of characters our goal is to assign diacritics to each character. Our appoach is based on Maximum Entropy (MaxEnt henceforth) technique (Berger et al., 1996). MaxEnt can be used for sequence classification, by converting the activation scores into probabilities (through the soft-max function, for instance) and using the standard dynamic programming search algorithm (also known as Viterbi search). We find in the literature several other approaches of sequence classification such as (McCallum et al., 2000) and (Lafferty et al., 2001). The *conditional random fields* method presented in (Lafferty et al., 2001) is essentially a MaxEnt model over the entire sequence: it differs from the Maxent in that it models the sequence information, whereas the Maxent makes a decision for each state independently of the other states. The approach presented in (McCallum et al., 2000) combines Maxent with Hidden Markov models to allow observations to be presented as arbitrary overlapping features, and define the probability of state sequences given observation sequences.

We report in section 7 a comparative study between our approach and the most competitive diacritic restoration method that uses finite state machine algorithm (Nelken and Shieber, 2005). The MaxEnt framework was successfully used to combine a diverse collection of information sources and yielded a highly competitive model that achieves a 5.1% DER.

4 Automatic Diacritization

The performance of many natural language processing tasks, such as shallow parsing (Zhang et al., 2002) and named entity recognition (Florian et al., 2004), has been shown to depend on integrating many sources of information. Given the stated focus of integrating many feature types, we selected the MaxEnt classifier. MaxEnt has the ability to integrate arbitrary types of information and make a classification decision by aggregating all information available for a given classification.

4.1 Maximum Entropy Classifiers

We formulate the task of restoring diacritics as a classification problem, where we assign to each character in the text a label (i.e., diacritic). Before formally describing the method[1], we introduce some notations: let $\mathcal{Y} = \{y_1, \ldots, y_n\}$ be the set of diacritics to predict or restore, $\mathcal{X}$ be the example space and $\mathcal{F} = \{0,1\}^m$ be a feature space. Each example $x \in \mathcal{X}$ has associated a vector of binary features $f(x) = (f_1(x), \ldots, f_m(x))$. In a supervised framework, like the one we are considering here, we have access to a set of training examples together with their classifications: $\{(x_1, y_1), \ldots, (x_k, y_k)\}$.

[1]This is not meant to be an in-depth introduction to the method, but a brief overview to familiarize the reader with them.

The MaxEnt algorithm associates a set of weights $(\alpha_{ij})_{j=1\ldots m}^{i=1\ldots n}$ with the features, which are estimated during the training phase to maximize the likelihood of the data (Berger et al., 1996). Given these weights, the model computes the probability distribution over labels for a particular example x as follows:

$$P(y|x) = \frac{1}{Z(x)} \prod_{j=1}^{m} \alpha_{ij}^{f_j(x)},\ Z(x) = \sum_i \prod_j \alpha_{ij}^{f_j(x)}$$

where $Z(\mathcal{X})$ is a normalization factor. To estimate the optimal α_j values, we train our MaxEnt model using the *sequential conditional generalized iterative scaling* (SCGIS) technique (Goodman, 2002). While the MaxEnt method can nicely integrate multiple feature types seamlessly, in certain cases it is known to overestimate its confidence in especially low-frequency features. To overcome this problem, we use the regularization method based on adding *Gaussian priors* as described in (Chen and Rosenfeld, 2000). After computing the class probability distribution, the chosen diacritic is the one with the most *aposteriori* probability. The decoding algorithm, described in section 4.2, performs sequence classification, through dynamic programming.

4.2 Search to Restore Diacritics

We are interested in finding the diacritics of all characters in a script or a sentence. These diacritics have strong interdependencies which cannot be properly modeled if the classification is performed independently for each character. We view this problem as *sequence classification*, as contrasted with an *example-based classification* problem: given a sequence of characters in a sentence $x_1 x_2 \ldots x_L$, our goal is to assign diacritics (labels) to each character, resulting in a sequence of diacritics $y_1 y_2 \ldots y_L$. We make an assumption that diacritics can be modeled as a limited order Markov sequence: the diacritic associated with the character i depends only on the diacritics associated with the k previous diacritics, where k is usually equal to 3. Given this assumption, and the notation $x_1^L = x_1 \ldots x_L$, the conditional probability of assigning the diacritic sequence y_1^L to the character sequence x_1^L becomes

$$\begin{aligned} & p\left(y_1^L | x_1^L\right) = \\ & \quad p\left(y_1 | x_1^L\right) p\left(y_2 | x_1^L, y_1\right) \ldots p\left(y_L | x_1^L, y_{L-k+1}^{L-1}\right) \end{aligned} \tag{1}$$

and our goal is to find the sequence that maximizes this conditional probability

$$\hat{y}_1^L = \arg\max_{y_1^L} p\left(y_1^L | x_1^L\right) \tag{2}$$

While we restricted the conditioning on the classification tag sequence to the previous k diacritics, we do not impose any restrictions on the conditioning on the characters – the probability is computed using the entire character sequence x_1^L.

To obtain the sequence in Equation (2), we create a *classification tag lattice* (also called *trellis*), as follows:

- Let x_1^L be the input sequence of character and $S = \{s_1, s_2, \ldots, s_m\}$ be an enumeration of $\mathcal{Y}^k$ $(m = |\mathcal{Y}|^k)$ - we will call an element s_j a state. Every such state corresponds to the labeling of k successive characters. We find it useful to think of an element s_i as a vector with k elements. We use the notations $s_i[j]$ for j^{th} element of such a vector (the label associated with the token $x_{i-k+j+1}$) and $s_i[j_1 \ldots j_2]$ for the sequence of elements between indices j_1 and j_2.
- We conceptually associate every character $x_i, i = 1, \ldots, L$ with a copy of S, $S^i = \{s_1^i, \ldots, s_m^i\}$; this set represents all the possible labelings of characters x_{i-k+1}^i at the stage where x_i is examined.
- We then create links from the set S^i to the S^{i+1}, for all $i = 1 \ldots L-1$, with the property that

$$w\left(s_{j_1}^i, s_{j_2}^{i+1}\right) = \begin{cases} p\left(s_{j_1}^{i+1}[k] \,|x_1^L,\, s_{j_2}^{i+1}[1..k-1]\right) \\ \quad if\ s_{j_1}^i[2..k] = s_{j_2}^{i+1}[1..k-1] \\ 0 \qquad\qquad \text{otherwise} \end{cases}$$

  These weights correspond to probability of a transition from the state $s_{j_1}^i$ to the state $s_{j_2}^{i+1}$.
- For every character x_i, we compute recursively[2]

$$\begin{aligned} \beta_0(s_j) &= 0, j = 1, \ldots, k \\ \beta_i(s_j) &= \max_{j_1=1,\ldots,M} \beta_{i-1}(s_{j_1}) + \log w\left(s_{j_1}^{i-1}, s_j^i\right) \\ \gamma_i(s_j) &= \\ & \arg\max_{j_1=1,\ldots,M} \beta_{i-1}(s_{j_1}) + \log w\left(s_{j_1}^{i-1}, s_j^i\right) \end{aligned}$$

  Intuitively, $\beta_i(s_j)$ represents the log-probability of the most probable path through the lattice that ends in state s_j after i steps, and $\gamma_i(s_j)$ represents the state just before s_j on that particular path.
- Having computed the $(\beta_i)_i$ values, the algorithm for finding the best path, which corresponds to the solution of Equation (2) is
  1. Identify $\hat{s}_L^L = \arg\max_{j=1\ldots L} \beta_L(s_j)$
  2. For $i = L-1 \ldots 1$, compute

$$\hat{s}_i^i = \gamma_{i+1}\left(\hat{s}_{i+1}^{i+1}\right)$$

[2]For convenience, the index i associated with state s_j^i is moved to β; the function $\beta_i(s_j)$ is in fact $\beta\left(s_j^i\right)$.

3. The solution for Equation (2) is given by

$$\hat{y} = \left\{\hat{s}_1^1[k], \hat{s}_2^2[k], \ldots, \hat{s}_L^L[k]\right\}$$

The runtime of the algorithm is $\Theta\left(|\mathcal{Y}|^k \cdot L\right)$, linear in the size of the sentence L but exponential in the size of the Markov dependency, k. To reduce the search space, we use *beam-search.*

4.3 Features Employed

Within the MaxEnt framework, any type of features can be used, enabling the system designer to experiment with interesting feature types, rather than worry about specific feature interactions. In contrast, with a rule based system, the system designer would have to consider how, for instance, lexical derived information for a particular example interacts with character context information. That is not to say, ultimately, that rule-based systems are in some way inferior to statistical models – they are built using valuable insight which is hard to obtain from a statistical-model-only approach. Instead, we are merely suggesting that the output of such a rule-based system can be easily integrated into the MaxEnt framework as one of the input features, most likely leading to improved performance.

Features employed in our system can be divided into three different categories: lexical, segment-based, and part-of-speech tag (POS) features. We also use the previously assigned two diacritics as additional features.

In the following, we briefly describe the different categories of features:

- **Lexical Features:** we include the character n-gram spanning the curent character x_i, both preceding and following it in a window of 7: $\{x_{i-3}, \ldots, x_{i+3}\}$. We use the current word w_i and its word context in a window of 5 (forward and backward trigram): $\{w_{i-2}, \ldots, w_{i+2}\}$. We specify if the character of analysis is at the beginning or at the end of a word. We also add joint features between the above source of information.

- **Segment-Based Features :** Arabic blank-delimited words are composed of zero or more prefixes, followed by a stem and zero or more suffixes. Each prefix, stem or suffix will be called a segment in this paper. Segments are often the subject of analysis when processing Arabic (Zitouni et al., 2005). Syntactic information such as POS or parse information is usually computed on segments rather than words. As an example, the Arabic white-space delimited word قَابلتهم contains a verb قَابل, a third-person feminine singular subject-marker ت (she), and a pronoun suffix هم (them); it is also a complete sentence meaning "*she met them.*" To separate the Arabic white-space delimited words into segments, we use a segmentation model similar to the one presented by (Lee et al., 2003). The model obtains an accuracy of about 98%. In order to simulate real applications, we only use segments generated by the model rather than true segments. In the diacritization system, we include the current segment a_i and its word segment context in a window of 5 (forward and backward trigram): $\{a_{i-2}, \ldots, a_{i+2}\}$. We specify if the character of analysis is at the beginning or at the end of a segment. We also add joint information with lexical features.

- **POS Features :** we attach to the segment a_i of the current character, its POS: $POS(a_i)$. This is combined with joint features that include the lexical and segment-based information. We use a statistical POS tagging system built on Arabic Treebank data with MaxEnt framework (Ratnaparkhi, 1996). The model has an accuracy of about 96%. We did not want to use the true POS tags because we would not have access to such information in real applications.

5 Data

The diacritization system we present here is trained and evaluated on the LDC's Arabic Treebank of diacritized news stories – Part 3 v1.0: catalog number LDC2004T11 and ISBN 1-58563-298-8. 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 600 documents from the An Nahar News Text. There are a total of 340,281 words. We split the corpus into two sets: training data and development test (devtest) data. The training data contains 288,000 words approximately, whereas the devtest contains close to 52,000 words. The 90 documents of the devtest data are created by taking the last (in chronological order) 15% of documents dating from "20021015_0101" (i.e., October 15, 2002) to "20021215_0045" (i.e., December 15, 2002). The time span of the devtest is intentionally non-overlapping with that of the training set, as this models how the system will perform in the real world.

Previously published papers use proprietary corpus or lack clear description of the training/devtest data split, which make the comparison to other techniques difficult. By clearly reporting the split of the publicly available LDC's Arabic Treebank corpus in this section, we want future comparisons to be correctly established.

6 Experiments

Experiments are reported in terms of word error rate (WER), segment error rate (SER), and diacritization error rate (DER). The DER is the proportion of incorrectly restored diacritics. The WER is the percentage of incorrectly diacritized white-space delimited words: in order to be counted as incorrect, at least one character in the word must have a diacritization error. The SER is similar to WER but indicates the proportion of incorrectly diacritized segments. A segment can be a prefix, a stem, or a suffix. Segments are often the subject of analysis when processing Arabic (Zitouni et al., 2005). Syntactic information such as POS or parse information is based on segments rather than words. Consequently, it is important to know the SER in cases where the diacritization system may be used to help disambiguate syntactic information.

Several modern Arabic scripts contains the consonant doubling "shadda"; it is common for native speakers to write without diacritics except the shadda. In this case the role of the diacritization system will be to restore the short vowels, doubled case ending, and the vowel absence "sukuun". We run two batches of experiments: a first experiment where documents contain the original shadda and a second one where documents don't contain any diacritics including the shadda. The diacritization system proceeds in two steps when it has to predict the shadda: a first step where only shadda is restored and a second step where other diacritics (excluding shadda) are predicted.

To assess the performance of the system under different conditions, we consider three cases based on the kind of features employed:

1. system that has access to lexical features only;
2. system that has access to lexical and segment-based features;
3. system that has access to lexical, segment-based and POS features.

The different system types described above use the two previously assigned diacritics as additional feature. The DER of the shadda restoration step is equal to 5% when we use lexical features only, 0.4% when we add segment-based information, and 0.3% when we employ lexical, POS, and segment-based features.

Table 2 reports experimental results of the diacritization system with different feature sets. Using only lexical features, we observe a DER of 8.2% and a WER of 25.1% which is competitive to a state-of-the-art system evaluated on Arabic Treebank Part 2: in (Nelken and Shieber, 2005) a DER of 12.79% and a WER of 23.61% are reported. The system they described in (Nelken and Shieber, 2005) uses lexical, segment-based, and morphological information. Table 2 also shows that, when segment-based information is added to our system, a significant improvement is achieved: 25% for WER (18.8 vs. 25.1), 38% for SER (9.4 vs. 13.0), and 41% for DER (5.8 vs. 8.2). Similar behavior is observed when the documents contain the original shadda. POS features are also helpful in improving the performance of the system. They improved the WER by 4% (18.0 vs. 18.8), SER by 5% (8.9 vs. 9.4), and DER by 5% (5.5 vs. 5.8).

True shadda			**Predicted shadda**		
WER	*SER*	*DER*	*WER*	*SER*	*DER*
Lexical features					
24.8	12.6	7.9	25.1	13.0	8.2
Lexical + segment-based features					
18.2	9.0	5.5	18.8	9.4	5.8
Lexical + segment-based + POS features					
17.3	8.5	5.1	18.0	8.9	5.5

Table 2: The impact of features on the diacritization system performance. The columns marked with "True shadda" represent results on documents containing the original consonant doubling "shadda" while columns marked with "Predicted shadda" represent results where the system restored all diacritics including shadda.

Case-ending in Arabic documents consists of the diacritic attributed to the last character in a white-space delimited word. Restoring them is the most difficult part in the diacritization of a document. Case endings are only present in formal or highly literary scripts. Only educated speakers of modern standard Arabic master their use. Technically, every noun has such an ending, although at the end of a sentence no inflection is pronounced, even in formal speech, because of the rules of 'pause'. For this reason, we conduct another experiment in which case-endings were stripped throughout the training and testing data without the attempt to restore them.

We present in Table 3 the performance of the diacritization system on documents without case-endings. Results clearly show that when case-endings are omitted, the WER declines by 58% (7.2% vs. 17.3%), SER is decreased by 52% (4.0% vs. 8.5%), and DER is reduced by 56% (2.2% vs. 5.1%). Also, Table 3 shows again that a richer set of features results in a better performance; compared to a system using lexical features only, adding POS and segment-based features improved the WER by 38% (7.2% vs. 11.8%), the SER by 39% (4.0% vs. 6.6%), and DER by 38% (2.2% vs.

True shadda			Predicted shadda		
WER	*SER*	*DER*	*WER*	*SER*	*DER*
Lexical features					
11.8	6.6	3.6	12.4	7.0	3.9
Lexical + segment-based features					
7.8	4.4	2.4	8.6	4.8	2.7
Lexical + segment-based + POS features					
7.2	4.0	2.2	7.9	4.4	2.5

Table 3: Performance of the diacritization system based on employed features. System is trained and evaluated on documents without case-ending. Columns marked with "True shadda" represent results on documents containing the original consonant doubling "shadda" while columns marked with "Predicted shadda" represent results where the system restored all diacritics including shadda.

3.6%). Similar to the results reported in Table 2, we show that the performance of the system are similar whether the document contains the original shadda or not. A system like this trained on non case-ending documents can be of interest to applications such as speech recognition, where the last state of a word HMM model can be defined to absorb all possible vowels (Afify et al., 2004).

7 Comparison to other approaches

As stated in section 3, the most recent and advanced approach to diacritic restoration is the one presented in (Nelken and Shieber, 2005): they showed a DER of 12.79% and a WER of 23.61% on Arabic Treebank corpus using *finite state transducers* (FST) with a Katz language modeling (LM) as described in (Chen and Goodman, 1999). Because they didn't describe how they split their corpus into training/test sets, we were not able to use the same data for comparison purpose.

In this section, we want essentially to duplicate the aforementioned FST result for comparison using the identical training and testing set we use for our experiments. We also propose some new variations on the finite state machine modeling technique which improve performance considerably.

The algorithm for FST based vowel restoration could not be simpler: between every pair of characters we insert diacritics if doing so improves the likelihood of the sequence as scored by a statistical n-gram model trained upon the training corpus. Thus, in between every pair of characters we propose and score all possible diacritical insertions. Results reported in Table 4 indicate the error rates of diacritic restoration (including shadda). We show performance using both Kneser-Ney and Katz LMs (Chen and Goodman, 1999) with increasingly large n-grams. It is our opinion that large n-grams effectively duplicate the use of a lexicon. It is unfortunate but true that, even for a rich resource like the Arabic Treebank, the choice of modeling heuristic and the effects of small sample size are considerable. Using the finite state machine modeling technique, we obtain similar results to those reported in (Nelken and Shieber, 2005): a WER of 23% and a DER of 15%. Better performance is reached with the use of Kneser-Ney LM.

These results still under-perform those obtained by MaxEnt approach presented in Table 2. When all sources of information are included, the MaxEnt technique outperforms the FST model by 21% (22% vs. 18%) in terms of WER and 39% (9% vs. 5.5%) in terms of DER.

The SER reported on Table 2 and Table 3 are based on the Arabic segmentation system we use in the MaxEnt approach. Since, the FST model doesn't use such a system, we found inappropriate to report SER in this section.

	Katz LM		Kneser-Ney LM	
n-gram size	WER	DER	WER	DER
3	63	31	55	28
4	54	25	38	19
5	51	21	28	13
6	44	18	24	11
7	39	16	23	11
8	37	15	23	10

Table 4: Error Rate in % for n-gram diacritic restoration using FST.

We propose in the following an extension to the aforementioned FST model, where we jointly determines not only diacritics but segmentation into affixes as described in (Lee et al., 2003). Table 5 gives the performance of the extended FST model where Kneser-Ney LM is used, since it produces better results. This should be a much more difficult task, as there are more than twice as many possible insertions. However, the choice of diacritics is related to and dependent upon the choice of segmentation. Thus, we demonstrate that a richer internal representation produces a more powerful model.

8 Conclusion

We presented in this paper a statistical model for Arabic diacritic restoration. The approach we propose is based on the Maximum entropy framework, which gives the system the ability to integrate different sources of knowledge. Our model has the advantage of successfully combining diverse sources of information ranging from lexical, segment-based and POS features. Both POS and segment-based features are generated by separate statistical systems – not extracted manually – in order to simulate real world applications. The segment-based features are extracted from a statistical morphological analysis system using WFST approach and the POS features are generated by a parsing model

	True Shadda		Predicted Shadda	
n-gram size	Kneser-Ney		Kneser-Ney	
	WER	DER	WER	DER
3	49	23	52	27
4	34	14	35	17
5	26	11	26	12
6	23	10	23	10
7	23	9	22	10
8	23	9	22	10

Table 5: Error Rate in % for n-gram diacritic restoration and segmentation using FST and Kneser-Ney LM. Columns marked with "True shadda" represent results on documents containing the original consonant doubling "shadda" while columns marked with "Predicted shadda" represent results where the system restored all diacritics including shadda.

that also uses Maximum entropy framework. Evaluation results show that combining these sources of information lead to state-of-the-art performance.

As future work, we plan to incorporate Buckwalter morphological analyzer information to extract new features that reduce the search space. One idea will be to reduce the search to the number of hypotheses, if any, proposed by the morphological analyzer. We also plan to investigate additional conjunction features to improve the accuracy of the model.

Acknowledgments

Grateful thanks are extended to Radu Florian for his constructive comments regarding the maximum entropy classifier.

References

M. Afify, S. Abdou, J. Makhoul, L. Nguyen, and B. Xiang. 2004. The BBN RT04 BN Arabic System. In *RT04 Workshop*, Palisades NY.

A. Berger, S. Della Pietra, and V. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

T. Buckwalter. 2002. Buckwalter Arabic morphological analyzer version 1.0. Technical report, Linguistic Data Consortium, LDC2002L49 and ISBN 1-58563-257-0.

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. computer speech and language. *Computer Speech and Language*, 4(13):359–393.

Stanley Chen and Ronald Rosenfeld. 2000. A survey of smoothing techniques for me models. *IEEE Trans. on Speech and Audio Processing.*

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

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

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

O. Emam and V. Fisher. 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.

R. Florian, H. Hassan, A. Ittycheriah, H. Jing, N. Kambhatla, X. Luo, N Nicolov, and S Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *Proceedings of HLT-NAACL 2004*, pages 1–8.

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

Joshua Goodman. 2002. Sequential conditional generalized iterative scaling. In *Proceedings of ACL'02.*

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

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML.*

Y.-S. Lee, K. Papineni, S. Roukos, O. Emam, and H. Hassan. 2003. Language model based Arabic word segmentation. In *Proceedings of the ACL'03*, pages 399–406.

Andrew McCallum, Dayne Freitag, and Fernando Pereira. 2000. Maximum entropy markov models for information extraction and segmentation. In *ICML.*

Rani Nelken and Stuart M. Shieber. 2005. Arabic diacritization using weighted finite-state transducers. In *ACL-05 Workshop on Computational Approaches to Semitic Languages*, pages 79–86, Ann Arbor, Michigan.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Conference on Empirical Methods in Natural Language Processing.*

M. Tayli and A. Al-Salamah. 1990. Building bilingual microcomputer systems. *Communications of the ACM*, 33(5):495–505.

D. Vergyri and K. Kirchhoff. 2004. Automatic diacritization of Arabic for acoustic modeling in speech recognition. In *COLING Workshop on Arabic-script Based Languages*, Geneva, Switzerland.

Tong Zhang, Fred Damerau, and David E. Johnson. 2002. Text chunking based on a generalization of Winnow. *Journal of Machine Learning Research*, 2:615–637.

Imed Zitouni, Jeff Sorensen, Xiaoqiang Luo, and Radu Florian. 2005. The impact of morphological stemming on Arabic mention detection and coreference resolution. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 63–70, Ann Arbor, June.