Maximum Entropy Modeling for Diacritization of Arabic Text

Ruhi Sarikaya, Ossama Emam, Imed Zitouni and Yuqing Gao*

IBM T.J. Watson Research Center, Yorktown Heights NY 10598
* IBM Egypt, El-Ahram, Giza, Egypt
{sarikaya,izitouni,yuqing}@us.ibm.com emam@eg.ibm.com

Abstract

We propose a novel modeling framework for automatic diacritization of Arabic text. The framework is based on Markov modeling where each grapheme is modeled as a state emitting a diacritic (or none) from the diacritic space. This space is exactly defined using 13 diacritics[1] and a null-diacritic and covers all the diacritics used in any Arabic text. The state emission probabilities are estimated using maximum entropy (MaxEnt) models. The diacritization process is formulated as a search problem where the most likely diacritization realization is assigned to a given sentence. We also propose a diacritization parse tree (DPT) for Arabic that allows joint representation of diacritics, graphemes, words, word contexts, morphologically analyzed units, syntactic (parse tree), semantic (parse tree), part-of-speech tags and possibly other information sources. The features used to train MaxEnt models are obtained from the DPT. In our evaluation we obtained 7.8% diacritization error rate (DER) and 17.3% word diacritization error rate (WDER) on a dialectal Arabic data using the proposed framework.

1. Introduction

Semitic languages such as Arabic and Hebrew are not as much studied as English for computer speech and language processing. However, in recent years, Arabic in particular has been receiving tremendous attention. Arabic poses a unique problem for automatic speech and language processing. Typically Arabic text is presented without vowels and other diacritical marks that are placed either above or below the graphemes. The process of adding vowels and other diacritical marks to Arabic text can be called Diacritization or, Vowelization. Vowelization defines the sense and meaning of each word, and how it will be pronounced. However, use of vowels and other diacritics has lapsed in modern Arabic writing.

Undiacritized text may cause confusion in meaning and pronunciation. A native Arabic speaker can insert the diacritics while speaking or reading undiacritized text to convey the intended meaning. While humans perform quite well on this task using their linguistic, semantic and syntactic knowledge, omission of vowels in written text leads to some serious problems for automatic speech and language processing systems. For example, the baseforms used for automatic speech recognition or the transcription [1] used for speech synthesis require diacritized text in order to resolve ambiguities and to achieve high performance. For the construction of such speech technology components, current state-of-the-art speech recognition applications usually use manually diacritized text, which is tedious and time consuming to generate. Even more obvious is the fact that online diacritization of written text is indispensable for a text-to-speech (TTS) system, in order to correctly pronounce the input text.

We propose a new diacritization scheme based on a principled statistical framework and information integration using diacritization parse tree (DPT). DPT is a tree structured joint representation of lexical, morphological, syntactic and semantic content of the sentence. As any statistical system it requires a training phase in which the system learns how to diacritize the text from an already diacritized training data where each sentence is organized in the form of a DPT. The new method ensures the generation of highly accurate diacritization thereby eliminating the cost of tedious and time consuming manual diacritization when used for bootstrapping to generate diacritized text. This results in high quality baseform generation for automatic speech recognition. Furthermore, in such applications as TTS it provides highly accurate diacritization of the text that improves the synthesis quality.

The rest of the paper is organized as follows. In Section 2 we summarize the related prior work on diacritization issue. A brief description of Arabic language is given in Section 3. A short overview of Maximum Entropy modeling is presented in Section 4. We present the proposed diacritization scheme in Section 5. Section 6 describes the experimental results followed by the conclusions and future research directions in Section 7.

2. The Arabic Language

As most Semitic languages Arabic is usually written without diacritical marks. In Table 1 we present diacritics with grapheme ل (/lam/) to demonstrate where they are placed in the text along with their names and meaning. In Table 2 we present the diacritic combinations that are treated a single unit in this study along with grapheme ل. The goal of using diacritic combinations in Table 2 is to make one-to-one assignment between a grapheme and a diacritic which allows us to formulate diacritization as a local classification task.

Arabic has 28 letters (graphemes), 25 of which are consonants and the remaining 3 are long vowels. Unlike many other languages short vowels are not represented by letters, hence they are not part of the alphabet. They are written as special symbols either above or below the graphemes. Here are the three short vowels:

1. The Fatha sign (َ) represents the "a" sound and is an oblique dash over a consonant (3rd row in Table 1).
2. The Kasra sign (ِ) represents the "i" sound and is an oblique dash under a consonant (5th row in Table 1).

[1] We consider shadda combined with short vowels and doubled case endings as a single diacritic as shown in Table 2.

Diacritics with ل	Name	Meaning
ل	NULL	Vowel absence
لَ	Fatha	/a/
لُ	Damma	/u/
لِ	Kasra	/i/
لً	Tanween al-fatha	/an/
لٌ	Tanween ad-damm	/un/
لٍ	Tanween al-kasr	/in/
لْ	Sukuun	Vowel absence

Table 1 Arabic Diacritics with grapheme ل

Combined diacritics with ل	Name
لَّ	Fatha-with-shadda
لُّ	Damma with shadda
لِّ	Kasra with shadda
لًّ	Tanween al-fatha-with shadda
لٌّ	Tanween ad-damm with shadda
لٍّ	Tanween al-kasr-with shadda

Table 2 Combined Arabic Diacritics with grapheme ل.

3. The Damma sign (ُ) represents the "u" sound and is a loop over a consonant that resembles the shape of a comma (4th row in Table 1).

In addition there are three kinds of diacritics:

1. "Sukuun", written as a small circle (ْ) above the Arabic consonant, is used to indicate that the letter is not vowelized (last row in Table 1).
2. "Shadda" (ّ) is a gemination mark that is placed above the Arabic letters and results in a repetition of the letter at the phonemic level.
3. "Nunation" (or tanween) is expressed by one of three different diacritics (Fathatan, Dammatan, Kasratan). These are placed above the last letter of the word and have the phonetic effect of placing an "N" at the end of the word.

Long vowels are constructed by combining 4 graphemes (و, ا, ى, ي) with the short vowels. Next, we present an overview of the prior work on Arabic diacritization.

3. Relevant Prior Work

Prior to recent attention there have been relatively few studies tackling the diacritization issue in Arabic. In [2] a rule based method based on morphological analyzer is proposed for vowelization. In [3] another rule based grapheme to sound conversion method is proposed. The main disadvantage of rule based methods is that it is difficult to maintain the rules up-to-date, or extend to new applications due to the productive nature of any "living" spoken language.

More recently, there have been several new studies addressing diacritization problem. In [4] an example based top-down approach is adopted where each utterance to be diacritized is compared to the training data for matching sentence. New words are diacritized using character based n-gram models. In [5] conversational Arabic is diacritized by combining morphological and contextual information with the acoustic signal. Here diacritization is treated as an unsupervised tagging problem where each word is tagged as one of the many possible diacritizations provided by the Buckwalter's morphological analyzer [6]. In [7] an HMM-based diacritization method is presented where diacritized sentences were decoded from un-diacritized sentences. This method considered fully word based approach and considered only vowels (no additional diacritics). Recently, a weighted finite state transducer based algorithm [8] is proposed that employs characters and morphological units in addition to words. A character based generative diacritization scheme is enabled only for words that do not occur in the training data. It is not clear whether this method handles the case of two syllabification marks (shadda) showing the doubling of the preceding consonant and sukuun denoting the lack of a vowel. These methods provide a limited solution to the problem in terms of accuracy and diacritics coverage.

We propose to generate the full list of the diacritics that have been used in any Arabic text. Our method differs from the previous approaches in the way the diacritization problem is formulated and multiple information sources are integrated using DPT. DPT resembles the way we integrate semantic and lexical information sources for language modeling [9]. Here, we take full advantage of multiple available information sources by combining them within a joint model. Next, we give a brief description of Maximum Entropy (MaxEnt) modeling.

4. Maximum Entropy Modeling

The Maximum Entropy (MaxEnt) method is a flexible statistical modeling framework that has been used widely in many areas of natural language processing [10, 11]. Maximum entropy modeling produces a probability model that is as uniform as possible while matching empirical feature expectations exactly. This can be interpreted as making as few assumptions as possible in the model. The MaxEnt modeling combines multiple overlapping information sources. The information sources are combined as follows:

$$P(o \mid h) = \frac{e^{\sum_i \lambda_i f_i(o,h)}}{\sum_{o'} e^{\sum_j \lambda_j f_j(o',h)}}$$

which describes the probability of a particular outcome (e.g. one of the diacritics) given the history or context. Notice that the denominator includes a sum over all possible outcomes, o', which is essentially a normalization factor for probabilities to sum to 1. The indicator functions, f_i or features are "activated" when certain outcomes are generated for certain context.

$$f_i(o \mid h) = \begin{cases} 1, & \text{if } o = o_i \text{ and } q_i(h) = 1 \\ 0, & \text{otherwise} \end{cases}$$

where o_i is the outcome associated with feature f_i and $q_i(h)$ is an indicator function on histories. The MaxEnt models are trained using the improved iterative scaling algorithm [10]. Next, we present the MaxEnt based diacritization method.

5. Maximum Entropy Based Diacritization Using DPT

DPT allows joint modeling of all such information types as diacritic/grapheme,/morphological/lexical/semantic/syntactic[1]. We

[1] We refer to these as "*information-space*" from now on.

hypothesize that as we increase the *information-space* and the tightness of integration, diacritization model should further improve. We can construct a single probability model that models the joint probability of all of the available information sources in the *information-space*. To compute the joint probability of the sentence diacritization and its *information-space*, we can use features extracted from the *information-space* for predicting both diacritics and labels associated with each word. Even though the framework is generic to jointly represent the information sources in the *information-space*, we limit ourselves to using only grapheme, lexical and morphological content of the sentence, simply because we neither have syntactic nor semantic information for the data we use in this work. Nevertheless we applied a morphological analysis to the data. For example we split the word, الميعاد as ال and ميعاد in Fig. 1 to reduce the vocabulary and improve the coverage of the data. In this paper, we represent DPT using a bracket notation [9]. In this notation, each diacritic is associated with a grapheme (this association is denoted by "=") and the lexical and morphological labels are represented by opening and closing tokens, [LABEL and LABEL] respectively. The token sequence for the semantic parse in Fig. 1 is shown below:

[!S! [تغير d13=ر d12=ي d11=غ d10=ت تغير] [قد d9=د d8=ق قد] [الميعاد [ميعاد d7=د d6=ا d5=ع d4=ي d3=م ميعاد] [ال d2=ل d1=ا ال] الميعاد] !S!]

where d1-to-d13 stands for the diacritics assigned to graphemes. This representation completely defines the diacritization sequence and its lexical and morphological content. It also makes it easy to define a joint statistical model since it enables us to compute the probability of diacritics, morphological segment and word tokens using similar context information. We consider every token in the bracket representation as an outcome of the joint model. Note that each diacritic and the associated grapheme counts as two separate tokens.

The main benefit of using DPT for joint modeling becomes apparent when a set of alternatives are generated for a sentence rather than just a single surface form. For example we may have more than one DPTs for a given sentence because of alternative morphological analysis, tagging or semantic/syntactic parses. Joint modeling in these cases allows us not only to get the best diacritization but also the best morphological analysis and/or tagging and/or semantic/syntactic parses of the sentence. We have applied the very same idea to the joint modeling of semantic parses and the word sequence to improve the language modeling [9], where N-best list for each sentence from the speech recognition engine is parsed and rescored with the joint model to rank the sentences. Even though DPT is a compact representation we do not have to construct it to extract features to train the classifier. However, constructing DPT subsumes the typical way of training the MaxEnt model where only the diacritics which are the leaves of the DPT are used as outcomes whereas in joint modeling not only leaves but also internal nodes of DPT are also outcomes. As such they are predicted as well. Let $P(D)$ denote the probability of a diacritization sequence, *D,* for a sentence. However, we can jointly calculate the probability of *D* and the DPT within single model:

$$P(D) \approx P(D, DPT)$$

We consider every node (token) in the parse tree as an outcome of the joint model.

$$P(D,DPT) = \sum_{i=1}^{M} P(t_i | t_1, \ldots t_{i-1})$$

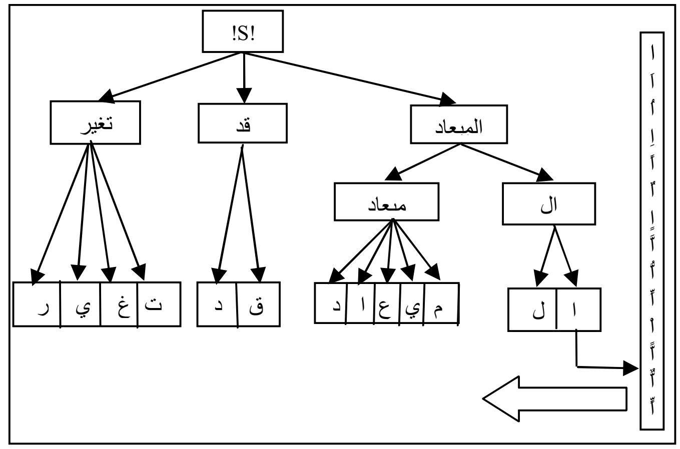

Fig 1. Diacritization Parse Tree.

where t_i represents a node in DPT represented in the bracketed form as explained in Section 5 and *M* is the total number of tokens in the DPT. Here are some of the question types $q(h)$ used to estimate, $P(t_i | t_1, \ldots, t_{i-1})$

1. Unigram history (empty history).
2. Previous diacritics (for bigram features, skipping word nodes).
3. Two previous diacritics (for trigram features, skipping word nodes).
4. grapheme bigram
5. grapheme trigram
6. # of diacritics since the beginning of the current word and current word.
7. # of diacritics since the beginning of the current word, current word and previous diacritic.
8. token 4gram.
9. current and grandparent words (for morphologically tokenized data)

Note that any question one can ask from the DPT is a legitimate feature. As such there are numerous features one can generate. The "best" feature set depends on the task, type of diacritization parse tree and the amount of data. The features obtained from the diacritization parse tree are used to model state observation probabilities.

The diacritization process starts by assigning probabilities to each of the 14 diacritics listed in Table 1 and 2. We show the diacritics and diacritic combinations in Table 1 and 2, respectively. We treat each entry in these two tables as a unit in assigning them to graphemes where Arabic grapheme ل (/lam/) is used as an example in Tables 1 & 2. In Fig.1 we depict the diacritization process that starts by assigning diacritics to the right most grapheme. Each grapheme is treated as a state emitting a diacritic from Table 1 & 2. A Viterbi search is conducted by aligning the most likely diacritic sequence to the grapheme sequence in right-to-left direction for the whole sentence. The individual diacritic probabilities for each grapheme are obtained using MaxEnt models. Modeling state distributions using MaxEnt uses features that are longer span than the graphemes. Hence, prediction of next diacritics depends not only on current diacritic and current grapheme but also previous diacritics, current word, previous word, may be next word or some other high level semantic and syntactic units that may span several words. Therefore, Viterbi search that allows multiple alternative diacritics per grapheme and then finds the best diacritic sequence through back-tracing is not fully warranted. The first order Markov chain assumption which states that history has no influence on the chain's future evolution if the present is specified, is not fulfilled for this problem for the reasons mentioned above. Nevertheless, keeping the most likely diacritic candidate for each grapheme may not take full advantage of the true Viterbi search but can provide highly accurate results to be used for consecutive predictions. Unlike many of the previous approaches our method provides a score for each possible diacritization of a word. It also generates n-best list of possible diacritizations ranked according their scores.

6. Experiments

We use a manually diacritized dialectal Arabic corpus of 30891 sentences. This corpus has been labeled by the linguists who are the native speakers of this Arabic dialect. The corpus is randomly split into training and test set of sizes 29861 and 1030 sentences respectively. Training and test data have 170K (24953 unique) and 5897 (2578 unique) words, respectively. About 21% of the words in the test vocabulary are not covered in the training vocabulary. After removing the diacritics the training vocabulary size is reduced 15726 and the test data 2101 words. This implies that there are about 9K (undiacritized) words with multiple diacritizations. In order to reduce the vocabulary size and increase coverage, we also applied a morphological analysis to the data. This analysis starts with a predefined set of prefixes and suffixes and splits words in accordance with the Buckwalter's morphological analysis [6]. Applying morphological splitting reduced the vocabulary size to 17K for undiacritized and 10K for undiacritized training data. For each model we report two results: word level diacritization error rate (WDER) and diacritization error rate (DER). WDER stands for percentage of words that contain at least one diacritization mistake. DER measures percentage of wrong assignment of diacritics to graphemes. There are a number of features that can be obtained from the diacritization parse tree given in Fig. 1. We have not exhaustively searched for the "best" feature set that minimizes the diacritization error rates but rather investigated a small subset that we thought would help the prediction of the diacritization.

The results are reported in Table 3. In the table *ME-base* denotes the case where only the diacritics are predicted as the output of the model. This is the typical way of using MaxEnt type predictor whereas *ME-base-joint* denotes the case where the complete DPT is predicted along with the diacritics. *ME-morph* denotes the model that is trained on the morphologically tokenized data. Again *ME-morph-joint* denotes the case the complete DPT (constructed on the morphologically tokenized data) is predicted along with the diacritics. As seen in the table adding morphological information improves the WDER by 22% and DER by 24%. Using joint modeling to predict the DPT improves the diacritization performance more for the whole word model (*ME-base*) and slightly for the morphological model (*ME-morph*). The figures inside the parenthesis in the last two rows indicate the diacritization accuracy for the morphological segments. Note that for morphological segments are glued back to make full words before computing WDER in Table 3. We should also point out that we have not yet fully explored features we can extract from the DPT for improved modeling. It is worth mentioning that we also implemented [8]. Preliminary results indicate that our method outperforms [8] by about 20% percentage on DER and 35% on WDER on the LDC MSA data that is often used for diacritization. Detailed results will be presented at the conference.

MODELS	WDER	DER
ME-base	23.3	10.8
ME-base-joint	21.9	9.7
ME-morph	18.1 (15.0)	8.2
ME-morph-joint	**17.3 (14.3)**	**7.8**

Table 3 Diacritization Performance Results

7. Conclusions

We presented a new framework for diacritization of Arabic. The framework is based on Markov modeling with Maximum Entropy (MaxEnt) based state density estimation. MaxEnt modeling uses the newly proposed *diacritization parse tree* (DPT) to integrate multiple overlapping information sources in a unified manner. The results presented here are encouraging given the small size of training data and large out-of-vocabulary words in the test data. Our future research will focus on applying some Arabic linguistic rules to constrain the diacritization search (i.e. there are only six syllable types in Arabic, CV, CVC, CVV, CVVC, CVCC and CVVCC). Using improved MaxEnt training via feature selection and smoothing will also be investigated.

8. References

[1] M. Afify, et.al, "The BBN RT04 BN Arabic System, RT04 Workshop", Palisades NY. 2004.
[2] T. El Sadany and M. Hashish. "Semi-Automatic Vowelization of Arabic Verbs", Proc. 10th NC Conf., S. Arabia, 1988.
[3] Y. El-Imam, "Phonetization of Arabic: rules and algorithms", Comp. Speech and Lang. pp. 339-373, 2003.
[4] O. Emam and V. Fisher. "A Hierarchical Approach for the Statistical Vowelization of Arabic Text, US patent application US2005/0192809 A1, 2004.
[5] D. Vergyri and K. Kirchhoff. "Automatic Diacritization of Arabic for Acoustic Modeling in Speech Recognition", COLING Workshop on Arabic-script Based Languages, Geneva, Switzerland, 2004.
[6] T. Buckwalter. "Buckwalter Arabic morphological analyzer version 1.0", LDC2002L49 and ISBN 1-58563-257-0, 2002.
[[7] Y. Gal. "An HMM Approach to Vowel Restoration in Arabic and Hebrew", Proc. ACL-02 Workshop on Computational Approaches to Semitic Languages, 2002.
[8] R. Nelken and S. M. Shieber. "Arabic Diacriti-zation Using Weighted Finite-State Transducers", Proc. ACL-05 Workshop on Computational Approaches to Semitic Languages, pages 79-86, Ann Arbor, Michigan, 2005.
[9] H. Erdogan, R. Sarikaya, S.F. Chen, Y. Gao, M. Picheny. "Using semantic analysis to improve speech recognition performance", Comp. Speech and Lang. 19 321–343, 2005
[10] S. D Pietra, V. D. Pietra, J. Lafferty. "Inducing features of random fields", IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (4), 380–393, 1997.
[11] S. F. Chen, R. Rosenfeld. "A survey of smooth-ing techniques for ME models", IEEE Trans. Speech and Audio Process. 8 (1), 37–50, 2000.
[12] M. Afify, R. Sarikaya, H-K J. Kuo, L. Besacier, and Y. Gao, "On the Use of Morphological Analysis for Dialectal Arabic Speech Recognition", Interspeech2006 (submitted).