Arabic Diacritization through Full Morphological Tagging

Nizar Habash and **Owen Rambow**
Center for Computational Learning Systems
Columbia University
New York, NY 10115, USA
{habash,rambow}@cs.columbia.edu

Abstract

We present a diacritization system for written Arabic which is based on a lexical resource. It combines a tagger and a lexeme language model. It improves on the best results reported in the literature.

1 Introduction

Arabic is written without certain orthographic symbols, called **diacritics**, which represent among other things short vowels.[1] The restoration of diacritics to written Arabic is an important processing step for several natural language processing applications, including training language models for automatic speech recognition, text-to-speech generation, and so on. For a discussion of the role of diacritization, see (Maamouri et al., 2006). In this paper, we present a new diacritization module that outperforms the best previously published results, using a new combination of techniques. A more detailed presentation can be found in (Habash and Rambow 2007).

2 Diacritization in Arabic: Linguistic Description

Arabic script consists of two classes of symbols: letters and diacritics. Letters are always written whereas diacritics are optional: written Arabic can be fully diacritized, it can have some diacritics (to disambiguate certain words), or it can be entirely undiacritized. There are three types of diacritics: vowel, nunation, and shadda. Vowel diacritics represent Arabic's three short vowels and the absence of any vowel. The following are the four vowel-diacritics exemplified in conjunction with the letter ب *b* (we use Buckwalter transliteration): بَ *ba*, بُ *bu*, بِ *bi* and بْ *bo (no vowel)*. Nunation diacritics can only occur in word final positions in nominals (nouns, adjectives and adverbs). They represent a short vowel followed by an *n* sound: بًا[2] *bF*, بٌ *bN* and بٍ *bK*. Nunation is an indicator of nominal indefiniteness. Shadda is a consonant doubling diacritic: بّ *b*∼. The shadda can combine with vowel or nunation diacritics: بُّ *b*∼*u* or بٌّ *b*∼*N*. Additional *diacritical marks* in Arabic include the hamza, which appears in conjunction with a small number of letters (e.g., أ, إ, آ, ؤ , ئ). Since most Arabic encodings do not consider the hamza a diacritic, but rather a part of the letter (like the dot on the lowercase Roman *i* or under the Arabic *b*: ب), we do not count it here as part of the diacritic set.

Functionally, diacritics can be split into two different kinds: **lexemic diacritics** and **inflectional diacritics**. Lexemic diacritics distinguish between two lexemes.[3] For example, the diacritization difference between the lexemes كَاتِب *kAtib* 'writer' and كَاتَب *kAtab* 'to correspond' distinguish between the meanings of the word rather than their inflections. Thus, there are lexemes that look alike when undiacritized but are spelled differently when diacritized. Note that there are also distinct lexemes that are always spelled the same way, even when diacritized – their difference is only a difference in word sense.

Inflectional diacritics distinguish different inflected forms of the *same* lexeme. For instance, the final diacritics in كَتَبْتُ *katabtu* 'I wrote' and كَتَبْتَ *katabta* 'you wrote' distinguish the person of the subject of the verb. We further distinguish be-

[1]This research was supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of DARPA.

[2]Arabic orthography calls for adding a silent Alif (ا) in conjunction with ـً in words ending with a consonant.

[3]A **lexeme** is an abstraction over inflected wordforms which groups together all those wordforms that differ only in terms of one of the morphological categories such as number, gender, aspect, or voice. The **lemma** is the distinguished word form which serves as citation form.

tween two types of inflectional diacritics: variant inflectional diacritics and invariant inflectional diacritics. The distinction is made with respect to two morphosyntactic features: nominal case and verbal mood. The variant inflectional diacritics need not always appear at the end of the word. For instance, the variant inflectional diacritics at the penultimate positions of the following two words distinguish their case: كَاتِبُهُ *kAtibuhu* 'his writer [nominative]' and كَاتِبَهُ *kAtibahu* 'his writer [accusative]'.

3 The MADA-D System

In a previous publication, we described the Morphological Analysis and Disambiguation of Arabic (MADA) system (Habash and Rambow, 2005). The basic approach used in MADA is inspired by the work of Hajič (2000) for tagging morphologically rich languages, which was extended to Arabic independently by Hajič et al. (2005). In this approach, a set of taggers are trained for individual linguistic features which are components of the full morphological tag (such as core part-of-speech, tense, number, and so on). In Arabic, we have ca. 2,000 to 20,000 morphological tags, depending on how we count. The Buckwalter Arabic Morphological Analyzer (BAMA) (Buckwalter, 2004) is consulted to produce a list of possible analyses for a word. BAMA returns, given an undiacritized inflected word form, all possible morphological analyses, *including full diacritization* for each analysis. The results of the individual taggers are used to choose among these possible analyses. The algorithm we proposed in (Habash and Rambow, 2005) for choosing the best BAMA analysis simply counts the number of predicted values for the set of linguistic features in each candidate analysis. Hajič et al. (2005), however, weigh the predicted values by their probability or confidence measure. To our knowledge, no results on diacritization have been previously reported using this particular approach to tagging.[4]

In this paper, we extend our basic MADA system in the following ways: First, we follow Hajič et al. (2005) in including case, mood, and nunation as features, because of its importance to diacritization. Second, we replace the YAMCHA (Kudo and Matsumoto, 2003) implementation of Support Vector Machines (SVMs) with SVMTool (Giménez and Màrquez, 2004) as our machine learning tool, for reasons of speed, at the cost of a slight decrease in accuracy. Like Hajič et al. (2005), we do not use Viterbi decoding. Finally, we introduce a specialized module for resolving residual ambiguity after the basic tagging is done. We explain this module in detail next. We train our classifiers on the exact training set defined by Zitouni et al. (2006), a subpart of the third segment of the Penn Arabic Treebank (Maamouri et al., 2004) ("ATB3-Train", 288,000 words). We also (reluctantly) follow them in having a single set for development and testing ("ATB3-Devtest", 52,000 words), rather than separate development and test sets (as is common), in order to be able to compare our results to theirs.

Up until this point, MADA-D has narrowed the list of possible analyses of a word (supplied by BAMA) down to a small number. This number can sometimes be greater than one for two reasons: first, the way in which we use the output of the taggers to choose among the analyses may yield a tie among several analyses; second, there may be lexeme-based diacritic ambiguity, and the morphological taggers cannot disambiguate lexemic diacritization. To address the residual ambiguity, we implemented a second component. Ideally, this would be (or include) a full word sense disambiguation (WSD) system, but WSD is a hard problem. Instead, we approximate WSD using standard n-gram language models. We use two types of data for training: fully diacritized word forms, and data in which we have replaced the inflected word by the diacritized citation form of the lexeme. Note that this procedure conflates lexemes that differ *only* in meaning, not in diacritization, as we are not actually interested in WSD for its own sake in this paper. The training corpus is the same corpus we use for the classifiers, ATB3-Train. This means that the diacritization and the choice of lexeme are done by hand, but it also means that the training set is quite small by the standards of language models. We build an open-vocabulary language model with Kneser-Ney smoothing using the SRILM toolkit (Stolcke, 2002). We will call the resulting language models X**LM**-n, where X is "D" for the fully diacritized word forms, or "L" for the lexeme citation forms, and n is the order of the n-

[4]Smith et al. (2005) also use the Buckwalter Analyzer in their Arabic morphological tagger, but then use a rather different approach to choosing among the possible analyses. They represent the possible analyses in a lattice, and a noisy channel model to choose among them. We leave to future work how the issue of diacritization can be integrated with their model.

grams ($n = 1, 2, 3$). When all candidate tokens (diacritized word or lexeme citation form) are unknown (out-of-vocabulary), the language model does not actually make a choice among them. We then use a diacritization unigram model, and then finally random choice. In the case of a preceding DLM-n model, this simply amounts to random choice, but in the case of a preceding LLM-n model, the diacritization model may actually make a non-random choice.

4 Related Work

We review three approaches that are directly relevant to us; we refer to the excellent literature review in (Zitouni et al., 2006) for a general review. Vergyri and Kirchhoff (2004) follow an approach similar to ours in that they choose from the diacritizations proposed by BAMA. However, they train a single tagger using unannotated data and EM, which necessarily leads to a lower performance. The most salient difference, however, is that they are motivated by the goal of improving automatic speech recognition, and have an acoustic signal parallel to the undiacritized text. All their experiments use acoustic models. They show that WER for diacritization decreases by nearly 50% (from 50%) when BAMA is added to the acoustic information, but the tagger does not help. It would be interesting to investigate ways of incorporating acoustic model information in our approach.

Ananthakrishnan et al. (2005) also work on diacritization with the goal of improving ASR. They use a word-based language model (using both diacritized and undiacritized words in the context) but back off to a character-based model for unseen words. They consult BAMA to narrow possible diacritizations for unseen words, but BAMA does not provide much improvement used in this manner.

Zitouni et al. (2006) use a maximum entropy classifier to assign a set of diacritics to the letters of each word. They use the output of a tokenizer (segmenter) and a part-of-speech tagger (which presumably tags the output of the tokenizer). They then use segment n-grams, segment position of the character being diacritized, the POS of the current segment, along with lexical features, including letter and word n-grams. Thus, while many of the same elements are used in their and our work (word n-grams, features related to morphological analysis), the basic approach is quite different: while we have one procedure that chooses a correct analysis (including tokenization, morphological tag, and diacritization), they have a pipeline of processors. Furthermore, Zitouni et al. (2006) do not use a morphological lexicon. To our knowledge, their system is the best performing currently, and we have set up our experiments to allow us to compare our results directly to their results.

	All Diacritics		**Ignore Last**	
Model	**WER**	**DER**	**WER**	**DER**
Only-DLM-1	39.4	14.5	13.8	6.6
Tagger-DLM-1	15.9	5.3	6.2	2.5
Tagger-DLM-2	15.2	5.1	5.8	2.4
Tagger-DLM-3	15.1	5.0	5.7	2.4
Tagger-LLM-1	16.0	5.3	6.3	2.6
Tagger-LLM-2	15.0	4.9	5.6	2.2
Tagger-LLM-3	**14.9**	**4.8**	**5.5**	**2.2**
Only-LLM-3	35.5	10.8	8.8	3.6
Tagger-noLM	16.0	5.3	6.3	2.6
Zitouni	18.0	5.5	7.9	2.5

Figure 1: Diacritization Results (all followed by single-choice-diac model); our best results are shown in boldface; Only-DLM-1 is the baseline; "Zitouni" is (Zitouni et al., 2006)

5 Results

There are several ways of defining metrics for diacritization. In order to assure maximal comparability with the work of Zitouni et al. (2006), we adopt their metric.[5] We count all words, including numbers and punctuation. Each letter (or digit) in a word is a potential host for a set of diacritics; we count all diacritics on a single letter as a single binary choice. So, for example, if we correctly predict a shadda but get the vowel wrong, it counts as a wrong choice. We approximate non-variant diacritization by removing all diacritics from the final letter (**Ignore Last**), while counting that letter in the evaluation. We give diacritic error rate (**DER**) which tells us for how many letters we incorrectly restored all diacritics, and word error rate (**WER**), which tells us how many words had at least one DER.

The results are shown in Figure 1. Going top to bottom, we first see the baseline, **Only-DLM-1**, which is simply a diacritization unigram model with

[5]We thank Imed Zitouni (personal communication) for details on their evaluation.

random choice for unseen words. We then show the results using the morphological tagger along with a language model. We first show results for the diacritization model, with 1-, 2-, and 3-grams. As we can see, the bigram language model helps slightly. The next three lines are the three lexeme n-gram models. Here we see that the unigram model performs worse than the unigram diacritization model, while the bigram and trigram models perform better (the trigram lexeme model is our best result). We interpret this as meaning that the lexeme model is useful only when context is taken into account, because it is actually performing a rudimentary form of WSD. We tease apart the contribution of the tagger and of the language model with two further experiments, in the next two lines: using just the lexeme language model (trigrams), and running just the tagger, followed by random choice. We can see that the tagger alone does as well as the tagger with the unigram lexeme model, while the lexeme model on its own does much worse. However, as expected, the lexeme model on its own for the Ignore Last measure is much closer to the performance of the tagger on its own. We conclude from this that the quite simple lexeme model is in fact contributing to the correct choice of the lexemic diacritics. Finally, we give the results of Zitouni et al. (2006) on the last line, which we understand to be the best published results currently. We see that we improve on their results in all categories. We can see the effect of our different approaches to diacritization in the numbers: while for WER we reduce the Zitouni et al error by 17.2%, the DER error reduction is only 10.9%. This is because we are choosing among complete diacritization options for white space-tokenized words, while Zitouni et al. (2006) make choices for each diacritic. This means that when we make a mistake, it may well affect several diacritics at once, so that the diacritic errors are concentrated in fewer words. This effect is even stronger when we disregard the final letter (30.4% reduction in WER versus 12.0% reduction in DER), suggesting that singleton errors in words tend to be in the final position (case, mood), as it is often hard for the tagger to determine these features.

6 Conclusion

We have shown that a diacritizer that uses a lexical resource can outperform a highly optimized *ad-hoc* diacritization system that draws on a large number of features. We speculate that further work on WSD could further improve our results. We also note the issue of unknown words, which will affect our system much more than that of (Zitouni et al., 2006). It is possible to construct a combined system which uses a lexicon, but backs off to a Zitouni-style system for unknown words. However, a large portion of the unknown words are in fact foreign words and names, and it is not clear whether the models learned handle such words well.

References

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

T. Buckwalter. 2004. Buckwalter Arabic morphological analyzer version 2.0.

J. Giménez and L. Màrquez. 2004. Svmtool: A general pos tagger generator based on support vector machines. In *Proceedings of LREC'04*.

N. Habash and O. Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of (ACL'05)*.

N. Habash and O. Rambow. 2007. Arabic Diacritization through Full Morphological Tagging: A Detailed Discussion. Techncial Report, Center for Computational Learning Systems, Columbia University.

Jan Hajič, Otakar Smrž, Tim Buckwalter, and Hubert Jin. 2005. Feature-based tagger of approximations of functional Arabic morphology. In *Proceedings of the Workshop on Treebanks and Linguistic Theories (TLT)*, Barcelona, Spain.

Jan Hajič. 2000. Morphological tagging: Data vs. dictionaries. In *Proceedings of (NAACL'00)*.

Taku Kudo and Yuji Matsumoto. 2003. Fast methods for kernel-based text analysis. In *Proceedings of (ACL'03)*.

Mohamed Maamouri, Ann Bies, and Tim Buckwalter. 2004. The Penn Arabic Treebank: Building a large-scale annotated arabic corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*, Cairo, Egypt.

Mohamed Maamouri, Ann Bies, and Seth Kulick. 2006. Diacritization: A challenge to arabic treebank annotation and parsing. In *Proceedings of the Conference of the Machine Translation SIG of the British Computer Society*.

Noah A. Smith, David A. Smith, and Roy W. Tromble. 2005. Context-based morphological disambiguation with random fields. In *Proceedings of (EMNLP05)*.

Andreas Stolcke. 2002. SRILM - an Extensible Language Modeling Toolkit. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*.

Dimitra Vergyri and Katrin Kirchhoff. 2004. Automatic diacritization of arabic for acoustic modeling in speech recognition. In *COLING 2004 Workshop on Computational Approaches to Arabic Script-based Languages*, Geneva, Switzerland.

Imed Zitouni, Jeffrey S. Sorensen, and Ruhi Sarikaya. 2006. Maximum entropy based restoration of arabic diacritics. In *Proceedings of ACL'06*.