# A Layered Language Model based Hybrid Approach to Automatic Full Diacritization of Arabic

**Mohamed Al-Badrashiny, Abdelati Hawwari, Mona Diab**
Department of Computer Science
The George Washington University
{badrashiny,abhawwari,mtdiab}@gwu.edu

**Abstract**

In this paper we present a system for automatic Arabic text diacritization using three levels of analysis granularity in a layered back off manner. We build and exploit diacritized language models (LM) for each of three different levels of granularity: surface form, morphologically segmented into prefix/stem/suffix, and character level. For each of the passes, we use Viterbi search to pick the most probable diacritization per word in the input. We start with the surface form LM, followed by the morphological level, then finally we leverage the character level LM. Our system outperforms all of the published systems evaluated against the same training and test data. It achieves a 10.87% WER for complete full diacritization including lexical and syntactic diacritization, and 3.0% WER for lexical diacritization, ignoring syntactic diacritization.

## 1 Introduction

Most languages have an orthographical system that reflects their phonological system. Orthographies vary in the way they represent word pronunciations. Arabic orthography employs an alphabetical system that comprises consonants and vowels. Short vowels are typically underspecified in the orthography. When present they appear as diacritical marks. Moreover, other phonological phenomena are represented with diacritics, such as letter doubling, syllable boundary markers, elongation, etc. In this paper, we are interested in restoring most of these diacritics, making them explicit in the written orthography. This process is referred to as diacritization/vowelization, or ”tashkeel” in Arabic. Absence of these diacritics from the orthography renders the text extremely ambiguous. Accordingly, the task of diacritization is quite important for many NLP applications such as morphological analysis, text to speech, POS tagging, word sense disambiguation, and machine translation.

Moreover, from a human processing perspective, having the orthography reflect the diacritics explicitly makes for better readability comprehension and pronunciation.

## 2 Linguistic Background

Unlike English, Arabic comprises an alphabet list of 28 letters. Short vowels are not explicitly marked in typical orthography as stand alone letters. The Arabic orthographic system employs a list of diacritics to express short vowels. The Arabic writing system maybe conceived to comprise two levels: consonantal letters including consonants and long vowels; and diacritics indicating short vowels and other pronunciation markers which are typically written above and/or below such consonantal letters.

The Arabic diacritics relevant to our study can be characterized as follows:[1]

- Short vowels *(a, i, u)*:[2], corresponding to the three short vowels (fatha ’a’, kasra ’i’, damma ’u’). They can occur word medially and/or word finally;

- Nunation “Tanween” *(F, K, N)*: these occur word finally only and they correspond to either an *an* ’F’ , *in* ’K’, or an *un* ’N’ sound. They indicate indefinite nominals as well as

[1]There are other diacritics that we don’t consider in the context of this work.

[2]We use Buckwalter (BW) transliteration scheme to represent Arabic in Romanized script throughout the paper. http://www.qamus.org/transliteration.htm

they could mark adverbials and some frozen expressions.

- Gemination (~), aka "shaddah": indicating doubling of the preceding character;
- Sukoun *o*: marks the absence of a vowel, typically appears between syllables, as well as word finally to indicate jussive syntactic mood for verbs.

Diacritization reflects morphological (including phonology) and grammatical information. Accordingly, in this paper we make a distinction between the two types of diacritization as follows:

**A) Morphological Diacritization:** Reflect the manner by which words are pronounced, not including the word final diacritization except the last letter diacritization. Morphological diacritization could be further subdivided into:

- Word structure or lexical diacritization: this represents the internal structure of words, that distinguish different possible readings of a phonologically ambiguous word (homograph) when the diacritics are missing. For instance, the Arabic word *mlk* could have the following readings: *malik* (king), *malak*(angel/he possessed), *mulok*(kingdom/property), *milok*(ownership), or *mal ak* (gave possession to another);
- Inflectional diacritization: this represents the morphophonemic level of handling affixations (prefixes, suffixes and clitics), how morphemes interact with each other, making possible morphophonemic changes which are reflected in the phonological and orthographic systems. For example the Arabic word *qAblthm* could be *qAbalatohum* (I met them), *qAbalotahum* (you_masc. met them), *qAbalatohum* (she met them) or *qAbalotihim*(you_fem. met them).

**B) Syntactic Diacritization:** Syntactic functions are represented by adding one of short vowels or nunation to the end of most of Arabic words, indicating the word's grammatical function in the sentence. For example, in a sentence like "*zAra Alwaladu zamiylahu*" (the boy visited his colleague), the diacritization of the last letters in the words *Alwaladu* and *zamiyla* indicate the syntactic roles of grammatical subject **u**, and grammatical object **a**, respectively.

### 2.1 Levels of Diacritization

Although native speakers of Arabic can read the majority of Arabic script without explicit diacritical marks being present, some diacritic symbols in some cases are crucial in order to disambiguate/pronounce homographical words. Historically, diacritics were invented by Arabic grammarians more than 200 years after the emergence of the Arabic writing system which was primarily consonantal. In Modern Standard Arabic (MSA) script, there are several levels of possible diacritization:

- **No Diacritization**: This level is completely underspecified. The script is subject to ambiguity, especially with homographical words;
- **Full Diacritization**: The reverse where there is complete specification, namely where each consonant is followed by a diacritic. This level is used more in classical and educational writing;
- **Partial Diacritization**: This level is anywhere in between the two previous levels, typically writer dependent. In this case, the writer adds diacritics where s/he deems fit (Zaghouani et al., 2016).

### 2.2 Challenges

There are a number of challenges in Arabic diacritization, we can list some of them as follows:

- **Morphological aspects**: Some Arabic words serve as a phrase or full sentence such as *waS alatohA* (she delivered her), *waS alotuhA* (I delivered her), and *waS alotihA* (you_feminine drove her);
- **Syntactic aspects**: Arabic is a free word-order language, syntactic functions are realized on the morphological level via word final diacritization in most cases. However, we note changes to the penultimate orthographic realization of the consonants due to syntactic position. For example, *>abonA&uhu*, *>abonA}ihi*, and *>abonA'ahu*, all corresponding to "his sons" but reflect different syntactic case: nominative, genitive, accusative, respectively.
- **Phonological aspects**: The phonological system exhibits assimilation in cases of affixation word finally. For example the

word final possessive suffix *h* meaning "his" in the following word takes on the same vowel/diacritic as that of the lexeme it is attached to: *kitAbi+hi* (his book) and *kitAbu+hu* (his book). It is important to note that the short vowel diacritic attached to the *h* suffix has no semantic or syntactic interpretation, it is a pure assimilation vowel harmony effect.

## 3 Approach

Figure 1 illustrates the system architecture of our proposed solution for MSA Full diacritization. Our approach relies on having fully diacritized data for building various types of language models at training time: a word level language model (WLM), a morpheme level language model (MLM), a character+diacritic level language model (CLM). The WLM is created in the diacritized untokenized surface level words. We experiment with 1-5 gram WLMs. The MLM are created using the same WLMs but after tokenizing them into prefix, stem, and suffix components where each is fully diacritized. Thus each 1 gram in the WLM is equivalent to 3 grams in the MLM, i.e. this renders MLMs of 3, 6, 9, 12, and 15, corresponding to the WLM of 1, 2, 3, 4, 5, respectively. Finally for CLMs, we are using the WLMs but after segmenting them into characters+associated diacritics. The maximum gram size we managed to build is 20 grams. Thus, each 1 gram in the word level WLM is equivalent to 4 grams in the character level, given that the smallest word in Arabic is two consonants long which is equivalent to 4 characters, i.e. each consonant is associated with at least one diacritic. This means that the LMs we are experimenting with for the character level are of sizes 4, 8, 12, 16, and 20 grams.

At test time, the undiacritized input text goes through the following pipeline:

**a) Word-Level Diacritization:** In this step, we leverage the WLM created at train time using all possible diacritizations for each word in the input raw text using the training data. If there are new words (out of vocabulary [OOV]) that have not been seen in the training data, they are tagged as unknown (UNK). A lattice search technique (for example: Viterbi or A* search) is then used to select the best diacritization for each word based on context.

**b) Morpheme Level Diacritization:** The output from the first step is being morphologically analyzed using SAMA (Maamouri et al., 2010). We only keep the morphological analyses that match the diacritization from the WLM. But if there is any word that is tagged as UNK, we keep all of its morphological analyses if they exist. If SAMA failed to find a possible morphological solution for any word (ex: non-Arabic word), it is marked as UNK. The MLM is used via a lattice search technique to pick the best morphological solution for each word; hence the best diacritization.

**c) Character-Level Diacritization:** If there are still some UNK words after steps (a) and (b), the CLM is used to find a plausible solution for them.

## 4 Experimental Setup

### 4.1 Data

Several studies have been carried out on the problem of full automatic diacritization for MSA. Five of these studies, that also yield the most competitive results despite approaching the problem in different ways, use and report on the same exact data sets. These studies are Zitouni et al. (2006), Habash and Rambow (2007), Rashwan et al. (2011), Abandah et al. (2015), and Belinkov and Glass (2015). We will use the same data which is 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 Full diacritization comprising both morphological and syntactic diacritization. This corpus includes 600 documents from the Annahar News Text. There are a total of 340,281 words. The data is split as follows into two sets:

- Training data comprising approximately 288K words;
- Test data (TEST): comprises 90 documents selected by taking the last 15% of the total number of documents in chronological order dating from "20021015 0101" to "20021215 0045". It comprises approximately 52K words.

But having a single set TEST serving as both test and dev data is not correct which is what previous studies have done. Therefore, we split the data into three parts instead of two. We split off 10% of the training data and use it as a development set, rendering our training data (TRAIN) to

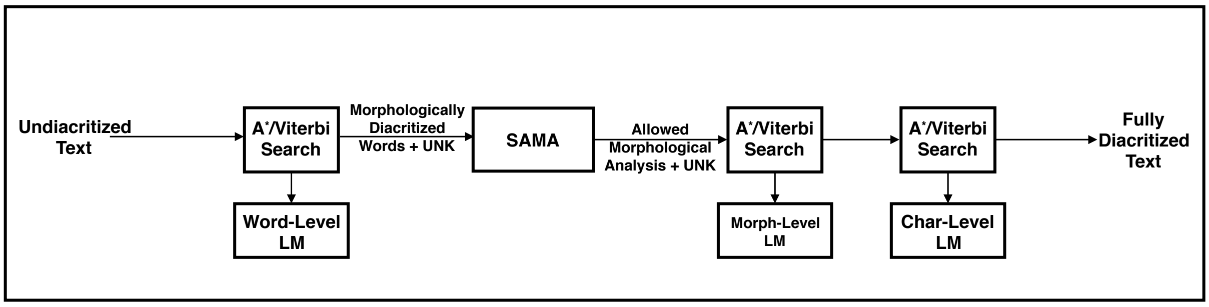

Figure 1: System Architecture.

comprise only 90% of the original training data. We keep the same exact test data, TEST, as the previous studies however. Accordingly, the new current training data for this paper is roughly 259K words and the development set (DEV) comprises approximately 29K words. DEV is used for tuning our system.

In all of our experiments, we use TRAIN to train and build our models and DEV to find the best configuration parameters. TEST is used as held out data. It is only evaluated using the resulting best models on DEV.

### 4.2 Evaluation Metrics

We adopt the same metrics used by Zitouni et al. (2006), Habash and Rambow (2007), Rashwan et al. (2011), Abandah et al. (2015), and Belinkov and Glass (2015). These are word error rate (WER) and character error rate (CER). CER compares the predicted words to the gold words on the character level. WER compares the predicted diacritized word as a whole to the gold diacritized word. If there is one error in a word, the whole word is considered incorrect. All words are evaluated including digits and punctuation. In the case of morphological diacritization, word final diacritics are ignored. In the case of syntactic diacritization only word final diacritics are considered. Finally in the Full diacritization case, both morphological and syntactic diacritization are considered.

### 4.3 Baselines

We compare our approach against the following baselines:[3]

- Zitouni et al.: The best published results by Zitouni et al. (2006);
- Habash et al.: The best published results by Habash and Rambow (2007);
- Rashwan et al.: The best published results by Rashwan et al. (2011);
- Abandah et al.: The best published results by Abandah et al. (2015);
- Belinkov and Glass: The best published results by Belinkov and Glass (2015).

## 5 Evaluation

Table 1 illustrates the morphological and Full (morphological+syntactic) diacritization performance on DEV using the lattice search on the word, morpheme, and character levels. The language models are all built using the TRAIN dataset.

The table shows five experiments using A* search using 1, 2, 3, 4, and 5 grams LMs.[4] And two experiments using Viterbi search because the implementation we have for the Viterbi search supports 2 grams as a maximum size. The best performance is yielded by the Viterbi algorithm and 2-grams LMs (i.e. 2-grams for WLM, 6-grams for MLM, and 8-grams for CLM). It yields 6.11% WER for Full diacritization (FULL), corresponding to 2.61% WER for morphological diacritization (MORPH), i.e. by ignoring word final syntactic diacritics.

Table 2 compares the performance of our system to the baselines systems. It shows that our system is outperforming all of the published CER and WER on "MORPH" level. On "FULL" level, we outperform all of the baselines except (Abandah et al., 2015). They are doing better on the syntactic level diacritization. Their system is based

[3]The descriptions of these baselines systems are in section: 7-Related Work

[4]Note: every 1-gram in word level is equivalent to 3-grams morphological level and 4-grams in characters level

|  |  | FULL |  | MORPH |  |
|---|---|---|---|---|---|
| **Lattice Search Method** | **LM-Size** | **WER** | **CER** | **WER** | **CER** |
| Viterbi | 1 | 6.67% | 1.14% | 3.13% | 0.61% |
| **Viterbi** | **2** | **6.11%** | **1.06%** | **2.61%** | **0.55%** |
| A* | 1 | 6.51% | 1.09% | 3.01% | 0.57% |
| A* | 2 | 6.28% | 1.04% | 2.77% | 0.53% |
| A* | 3 | 6.26% | 1.04% | 2.74% | 0.53% |
| A* | 4 | 6.26% | 1.04% | 2.74% | 0.53% |
| A* | 5 | 6.18% | 1.03% | 2.66% | 0.51% |

Table 1: System performance on DEV. The best setup is by using the Viterbi algorithm via 2 grams LMs.

on a deep bidirectional long short-term memory (LSTM) model. These kinds of models can exploit long-range contexts; which could yield the better performance on the syntactic diacritization level. It is also worth mentioning that they are using a post-processing correction layer that applies some rules to fix some of the diacritization errors after the LSTM.

It should be highlighted, that in contrast to the previous studies, TEST remained a complete held out data set that was not explored at all during the tuning phase of the system development, where for the previous studies TEST was used as both a development and test set.

## 6 Error Analysis

By reviewing the errors rendered by our system and comparing them to the gold data we discovered several issues in the training and test data that affected the performance and evaluation results. We list them as follows:

**Undiacritized words:** There are many cases in both the training and test data where the words are completely undiacritized. Since we rely on fully diacritized texts to build our various language models, this type of error affects our system in two ways:

- Errors in the training data affect the quality of the language models that are built;
- Errors in the test data decrease the accuracy of our system because such cases are being counted as incorrect even if they are correctly diacritized by our system. Upon manual inspection, for example, our system renders the correct diacritization for the words: *xal af* ”left behind/gave birth” and *bAruwd* ”gun powder” are counted as errors because they are not diacritized at all in the gold test data set).

**Missing Case marker:** 25.2% of the syntactic diacritization errors are due to missing syntactic diacritization from the gold TEST words. Table 3 illustrates some examples of that.

## 7 Related Work

Many research efforts addressed the problem of automatic full Arabic diacritization, especially for MSA.

Gal (2002) developed a statistical system using HMM to restore Arabic diacritics and applied it on the Holy Quran as a corpus. Their approach did not include any language-specific knowledge. This system achieved a WER of 86% for morphological diacritization without syntactic diacritization.

El-Imam (2004) developed a comprehensive set of well-defined language-dependent rules, that are augmented by a dictionary, to be used in the transcription of graphemes into phonemes.

Nelken and Shieber (2005) developed a probabilistic model for Arabic diacritization using a finite state transducer, and trigram word and character based language models. Their approach used the ATB and achieved 7.33% WER without case endings (morphological diacritization) and 23.61% WER with case ending.

Ananthakrishnan et al. (2005) leveraged a word-level trigram model combined with a four-gram character language model. The authors used ATB as training data and used the LDC TDT4 Broadcast News data set as test data. The reported word accuracy using this model was 80.21%.

Zitouni et al. (2006) presented a statistical model based on a Maximum Entropy framework. Their approach integrates different sources of

| | | FULL | | MORPH | |
|---|---|---|---|---|---|
| **Training Data** | **System** | **WER** | **CER** | **WER** | **CER** |
| TRAIN+DEV | Zitouni et al. | 18.00% | 5.50% | 7.90% | 2.50% |
| TRAIN+DEV | Habash et al. | 14.90% | 4.80% | 5.50% | 2.20% |
| TRAIN+DEV | Rashwan et al. | 12.50% | 3.80% | 3.10% | 1.20% |
| TRAIN+DEV | Abandah et al. | **9.07%** | **2.72%** | 4.34% | 1.38% |
| TRAIN | Belinkov and Glass | N/A | 4.85 | N/A | N/A |
| TRAIN | Our System | 10.90% | 1.60% | **3.10%** | **0.60%** |
| TRAIN+DEV | Our System | 10.87% | 1.60% | **3.00%** | **0.59%** |

Table 2: Our System performance against baselines

| Word | POS |
|---|---|
| *AlHumayoDiy∼* | DET+NOUN_PROP |
| *mud∼ap* | NOUN+NSUFF_FEM_SG |
| *waragom* | CONJ+NOUN |
| *Eam∼An* | NOUN_PROP |
| *gayor* | NOUN |
| *IixorAj* | NOUN |

Table 3: Examples for the missing syntactic diacritics in TEST

knowledge including lexical, segment-based and POS features. They achieved a CER of 5.5% and a WER of 18.0% for morphological and syntactic diacritization. By ignoring case endings, they obtained a CER of 2.5% and a WER of 7.9%.

Elshafei et al. (2006) proposed a diacritic restoration system which uses HMM for modeling and a Viterbi algorithm to select the most probable diacritized form of a sentence. The result was 4.1% errors in the diacritical marking of letters.

Habash and Rambow (2007) proposed a diacritization system that is based on a lexical resource, combining a tagger and a lexeme language model. The system gets a list with all potential analysis for each word, then applies a series of Support Vector Machine (SVM) classifiers to several morphological dimensions, then combines the various values for the dimensions to decide on the final analysis chosen from among the various possible analyses provided by an underlying morphological analyzer such as BAMA. They achieved a CER of 5.5% and a WER of 14.9% for morphological and syntactic diacritization. And CER of 2.2% and a WER of 5.5% by ignoring case endings.

Shaalan et al. (2009) proposed a hybrid approach that relies on lexicon retrieval, a bigram word level language model, and SVM classification. The system achieves a reported WER of 12.16% for combined morphological and syntactic diacritization.

Rashwan et al. (2011) developed a hybrid approach with a two-layer stochastic system. They split the input sentence into smaller segments, where each segment is consisting of at leas one word. Then they use a WLM to diacritize the segments that all of its words can be found in the unigrams of the WLM. Another MLM is used to diacritize the segments that are out of vocabulary from the point of view if the WLM. The final output is the combination of the all segments. They achieved 12.5% WER and 3.8% for combined morphological and syntactic diacritization. And 3.1% WER and 1.2% CER by ignoring the case ending.

Hifny (2012) developed a diacritic restoration system which uses dynamic programming (DP), n-gram language model, and smoothing. The author reported a WER of 3.4% for morphological diacritization and a WER 8.9% for combined morphological and syntactic diacritization.

MADAMIRA (Pasha et al., 2014) is a morphological analysis and disambiguation tool of Arabic. It applies SVM and language models to predict the word's morphological features. The diacritization accuracy of MADAMIRA is 86.3% on MSA and 83.2% on the Egyptian dialect.

Abandah et al. (2015) trained a recurrent neural network (RNN) to transcribe undiacritized Arabic text with fully diacritized sentences. After that they used some post-processing correction rules to correct the output from the RNN. For example, if the undiacritized word can be found in the training data but its diacritization by the the RNN does not exist, they replace the output diacritization by the variant from the training data leveraging a minimum edit distance algorithm. They achieved a CER of 2.72% and a WER of 9.07% for morphological and syntactic diacritization. And CER of 1.38% and a WER of 4.34% by ignoring case endings.

Belinkov and Glass (2015) developed a recurrent neural network with long-short term memory (LSTM) layers for predicting diacritics in Arabic text. They achieved a CER of 4.85% for morphological and syntactic diacritization.

Although the system of Rashwan et al. (2011) looks close to our system, but there is a significant difference between the two systems. The method they used of splitting the input sentence into smaller segments and diacritizing each segment separate from others, results in information loss yielding a suboptimal solution. Unlike, their approach, we do not split the sentences at the OOV words. Instead, we pass on the probability values of the unknowns. Therefore, even if there are one or more words that are OOV from the point of view of any of our LMs, the searching technique remains able to benefits from surrounding words.

## 8 Conclusion

In this paper we introduce a hybrid approach to full Arabic diacritization that leverages three underlying language models on different levels of linguistic representation with a filtering step that relies on a morphological analyzer to find the most probable diacritization for undiacritized surface form Arabic text in context. The results show that the presented approach outperforms all published systems to date using the same training and test data.

## References

Gheith A. Abandah, Alex Graves, Balkees Al-Shagoor, Alaa Arabiyat, Fuad Jamour, and Majid Al-Taee. 2015. Automatic diacritization of arabic text using recurrent neural networks. *International Journal on Document Analysis and Recognition (IJDAR)*, 18(2):183–197.

Sankaranarayanan Ananthakrishnan, Srinivas Bangalore, and Shrikanth S. Narayanan. 2005. Automatic diacritization of arabic transcripts for automatic speech recognition. In *Proceedings of the International Conference on Natural Language Processing (ICON)*, Kanpur, India, December.

Yonatan Belinkov and James Glass. 2015. Arabic diacritization with recurrent neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2281–2285, Lisbon, Portugal, September. Association for Computational Linguistics.

Yousif A. El-Imam. 2004. Phonetization of arabic: rules and algorithms. *Computer Speech and Language*, 18(4):339 – 373.

Moustafa Elshafei, Husni Al-muhtaseb, and Mansour Alghamdi. 2006. Statistical methods for automatic diacritization of arabic text. In *Proceedings of Saudi 18th National Computer Conference (NCC18)*, Riyadh, Saudi Arabia.

Ya'akov Gal. 2002. An hmm approach to vowel restoration in arabic and hebrew. In *Proceedings of the ACL-02 Workshop on Computational Approaches to Semitic Languages*, SEMITIC '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nizar Habash and Owen Rambow. 2007. Arabic diacritization through full morphological tagging. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, NAACL-Short '07, pages 53–56, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yasser Hifny. 2012. Higher order n-gram language models for arabic diacritics restoration. In *Proceedings of the 12th Conference on Language Engineering (ESOLEC 12)*, Cairo, Egypt.

Mohamed Maamouri, Dave Graff, Basma Bouziri, Sondos Krouna, Ann Bies, and Seth Kulick. 2010. Ldc standard arabic morphological analyzer (sama) version 3.1.

Rani Nelken and Stuart M. Shieber. 2005. Arabic diacritization using weighted finite-state transducers. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, Semitic '05, pages 79–86, Stroudsburg, PA, USA. Association for Computational Linguistics.

Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan M. Roth. 2014. MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. In *Proceedings of LREC*, Reykjavik, Iceland.

M. A.A. Rashwan, M. A.S.A.A. Al-Badrashiny, M. Attia, S. M. Abdou, and A. Rafea. 2011. A stochastic arabic diacritizer based on a hybrid of factorized and unfactorized textual features. *Trans. Audio, Speech and Lang. Proc.*, 19(1):166–175, January.

Khaled Shaalan, Hitham M. Abo Bakr, and Ibrahim Ziedan. 2009. A hybrid approach for building arabic diacritizer. In *Proceedings of the EACL 2009 Workshop on Computational Approaches to Semitic*

*Languages*, Semitic '09, pages 27–35, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wajdi Zaghouani, Houda Bouamor, Abdelati Hawwari, Mona Diab, Ossama Obeid, Mahmoud Ghoneim, Sawsan Alqahtani, and Kemal Oflazer. 2016. Guidelines and framework for a large scale arabic diacritized corpus. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Imed Zitouni, Jeffrey S. Sorensen, and Ruhi Sarikaya. 2006. Maximum entropy based restoration of arabic diacritics. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 577–584, Stroudsburg, PA, USA. Association for Computational Linguistics.