ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/277021918

A Hybrid Approach for Arabic Diacritization

Conference Paper · June 2013

DOI: 10.1007/978-3-642-38824-8_5

CITATIONS
24

READS
297

4 authors, including:

Eslam Kamal
Microsoft
6 PUBLICATIONS **75** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Microsoft Power Virtual Agents View project

All content following this page was uploaded by Eslam Kamal on 22 May 2015.

The user has requested enhancement of the downloaded file.

A Hybrid Approach for Arabic Diacritization

Ahmed Said, Mohamed El-Sharqwi, Achraf Chalabi, and Eslam Kamal

Microsoft Advanced Technology Lab, Cairo, Egypt.
{v-ahsaid,moels,achalabi,eskam}@microsoft.com

Abstract. The orthography of Modern standard Arabic (MSA) includes a set of special marks called diacritics that carry the intended pronunciation of words. Arabic text is usually written without diacritics which leads to major linguistic ambiguities in most of the cases since Arabic words have different meaning depending on how they are diactritized. This paper introduces a hybrid diacritization system combining both rule-based and data- driven techniques targeting standard Arabic text. Our system relies on automatic correction, morphological analysis, part of speech tagging and out of vocabulary diacritization components. The system shows improved results over the best reported systems in terms of full-form diacritization, and comparable results on the level of morphological diacritization. We report these results by evaluating our system using the same training and evaluation sets used by the systems we compare against.. Our system shows a word error rate (WER) of 4.4% on the morphological diacritization, ignoring the last letter diacritics, and 11.4% on the full-form diacritization including case ending diacritics. This means an absolute 1.1% reduction on the word error rate (WER) over the best reported system.

Keywords: Arabic, Arabic orthography, diacritization, vowelization, morphology, morphology features, morphological analysis, part-of-speech tagging, automatic correction, Viterbi, case ending, natural language processing, language modeling, conditional random fields, CRF

1 Introduction

Arabic text in almost all genres of Modern standard Arabic (MSA) is written without short vowels, called **diacritics**. The restoration of these diacritics is valuable for natural language processing applications such as full-text search and text to speech. Devising a diacritization system for Arabic is a sophisticated task as Arabic language is highly-inflectional and derivational. Moreover, Arabic sentences are characterized with a relatively free word-order. The size of the Arabic vocabulary and the complex Arabic morphological structure can both be managed efficiently via working on the morpheme level (constituents of the words) instead of the word level. The system we have built relies heavily on two core components: the morphological analyzer and the part of speech (POS) tagger. By leveraging the systematic and compact nature of Arabic morphology, we have developed a high quality rule-based morphological analyzer with high recall, driven by a comprehensive lexicon and handcrafted rules.

Moreover, we have developed a lightweight statistical morphological analyzer that is trained on LDC's Arabic Treebank corpus (ATB) [8]. The POS tagger is used to resolve most of the morphological and syntactic ambiguities in context. In the next sections, we present our system as follows: Section 2 covers the linguistic description of Arabic diacritization. Section 3 briefly covers previous related work, in section 4, we elaborate on the different components of the diacritizer, and in section 5, we report our system's results compared to others, using the same evaluation setup: metrics and data.

2 Arabic Diacritization: Linguistic Description

The Arabic alphabet consists of 28 letters; 25 consonants such as "ب" (pronounced /b/) and 3 long vowel letters namelyى ,ا (pronounced /a:/), و (pronounced /u:/), and ي(pronounced /i:/). In addition to these letters, there are Arabic diacritics that are classified into 3 groups as shown in Table 1. In order to pronounce any consonant, it has to be associated with one or more of these diacritics.

The first group consists of the short vowel diacritics: *Fatha, Kasra* and *Damma*. Examples of short vowels association with the letter ب are: (i) بَ (pronounced as /b//a/), (ii) بِ (pronounced as /b//i/) and (iii) بُ (pronounced as /b//u/). The second group represents the doubled case ending diacritics (Nunation). These are vowels occurring at the end of nominal words (nouns, adjectives and adverbs) indicating nominal indefiniteness. This phenomenon is called "*Tanween*" and has the phonetic effect of adding an "N" sound after the short vowel at the word ending. *Tanween* applies to the 3 short vowels as follows: (i) *Tanween Fatha* as بًا (pronounced /b//an/), (ii) *Tanween Kasra* such as بٍ (pronounced /b//in/) and *Tanween Damma* as بٌ (pronounced /b//un/). The third group is composed of *Shadda* and *Sukuun* diacritics. *Shadda* reflects the doubling of a consonant, as in بّ (pronounced /b//b/), and is usually combined with a vowel diacritic as in بَّ (pronounced as /b//b//a/). *Sukuun* indicates the absence of a vowel as in بْ (pronounced /b/), and reflects a glottal stop. Diacritics could also be classified into two main categories based on their function. The first category consists of the lexeme diacritics, which determine the part-of-speech of a word as in (ذَهَبَ → "went" , ذَهَب → "gold"), and also the meaning of the word such as (رَجُل → "man", رِجْل → "leg"), while the second category reflects the syntactic function of the word in the sentence, also called case ending. For example, in the sentence "عَرَفَ اَلرَّجُلُ اَلْحَقِيقَةَ", the diacritic "*Fatha*" of the word "اَلْحَقِيقَة" reflects its "object" role in the sentence. While in sentence "وَضَحَتْ اَلْحَقِيقَةُ" the same word occurs as a "subject" hence its syntactic diacritic is a "*Damma*".

Table 1. Arabic diacritics represented in 3 groups.

Diacritic	Diacritic Name	Association with a consonant	Pronunciation
Short vowels (Group 1)			
◌َ	*Fatha*	بَ	/b//a/
◌ِ	*Kasra*	بِ	/b//i/
◌ُ	*Damma*	بُ	/b//u/
Double case ending (tanween)			
◌ً	*Tanween Fatha*	بً	/b//an/
◌ٍ	*Tanween Kasra*	بٍ	/b//in/
◌ٌ	*Tanween Damma*	بٌ	/b//un/
Syllabification marks			
◌ّ	*Shadda*	بّ	/b//b/
◌ْ	*Sukuun*	بْ	/b/

In almost all genres of the Modern standard Arabic (MSA) written text, diacritics are omitted, leading to a combinatorial explosion of ambiguities, since the same Arabic word can have different part-of-speeches and meanings, based on the associated diacritics, e.g.: (عَقْد → "contract", عِقْد → "necklace", عَقَّدَ → "complicate"). The absence of diacritics adds layers of confusion for novice readers and for automatic computation. For instance, the absence of diacritics becomes a serious obstacle to many of the applications including text to speech (TTS), intent detection, and automatic understanding in general. Therefore, automatic diacritization is an essential component for automatic processing of Arabic text.

3 Related work

We reviewed four approaches in the recent published literature on the diacritization problem; these four publications are the most relevant to our work.

Rashwan et al. [1] designed a stochastic Arabic diacritizer based on a hybrid of factorized and un-factorized textual features. They introduced dual-mode stochastic system to automatically diacritize the raw Arabic text. The first of these modes determines the most likely diacritics by choosing the sequence of full-form Arabic word diacritizations with maximum marginal probability via A* lattice search and long-horizon n-grams probability estimation multilayer Arabic text diacritizer. When full-form words are Out of Vocabulary (OOV), the system resorts to a second mode that factorizes each Arabic word into all its possible morphological constituents, while using the same techniques of the first mode to get the most likely sequence of morphemes,

hence the most likely diacritization. While Rashwan et al. [1] is the approach closest to our work, we introduce new techniques to handle OOVs and generate case endings that leading better results.

Habash and Rambow [2] use a morphological analyzer and a disambiguation system called MADA [4]. They use a feature set including case, mood, and nunation, and use SVMTool [7] as a machine learning tool. They use SRILM toolkit [9] to build an open-vocabulary statistical language model (SLM) with Kneser–Ney smoothing. Habash and Rambow [2] did experiments using the full-form words and the lexemes (prefix, stem, and suffix) citation form. The best results that have been reported are the ones they obtain with the lexemes form with trigram SLM [2]. The system does not handle OOV words which are not analyzed or haven't been seen during training.

Zitouni et al. [3] have built a diacritization framework that based on maximum entropy classification. The classifier is used to restore the missing diacritics on each word letters. They also use a tokenizer (segmenter) and a POS tagger. They use different signals such as the segment n-grams, segment position of the character, the POS of the current segment, and lexical features, including character and word n-grams. Although they don't have a morphological lexicon, they resort to statistical Arabic morphological analysis to segment Arabic words into morphemes (segments). These morphemes consist mainly of prefixes, stems, and suffixes. The maximum entropy model combines all these features together to restore the missing vowels of the input word sequence.

Emam and Fisher [6] introduced a hierarchical approach for diacritization. The approach starts with searching in a set of dictionaries of sentences, phrases and words using a top down strategy. First they search in a dictionary of sentences, if there is a matching sentence, they use the whole text. Otherwise the search starts with another dictionary of phrases, then dictionary of words to restore the missing diacritics. If there is no match at all previous layers, a character n-gram model is used to diacritize each word. No experimental results of this patented work have been mentioned in the available patent document.

The first three systems are trained and tested using LDC's Arabic Treebank (#LDC2004T11) of diacritized news stories text-part 3, v1.0 [8] that includes 600 documents (340 K words) from the Lebanese newspaper "AnNahar". The text is split into a training set (288 K words) and a test set (52 K words) [1], [2], [3]. To our knowledge, these three systems are currently the best performing systems. We adopt their metrics and use the same training and test set for fair comparison.

4 The Hybrid Diacritization System

Our diacritization system is designed as a pipeline of multiple components, each of which addressing a specific aspect of the vowel restoration problem. Figure 1 shows the system's overall architecture, where the diacritization is achieved through 3 main phases: (i) preprocessing, (ii) generation of valid morphological structures (analyses) for each word, combining analyses of both statistical and rule-based morphological analyzers to build a lattice of analyses, and (iii) disambiguation, to generate the diacritized text including case ending diacritics. The preprocessing phase does auto-

correction of the raw input text, targeting common Arabic mistakes (CAMs) [5]. Afterwards, the corrected text is tokenized. Each input word is then analyzed through both the statistical and rule-based morphological analyzers, combining zero or more morphological analyses. Each analysis is composed of zero or more prefix(es), the stem, zero or more suffix(es), the morphological pattern, the part of speech tag, and the word tag probability.

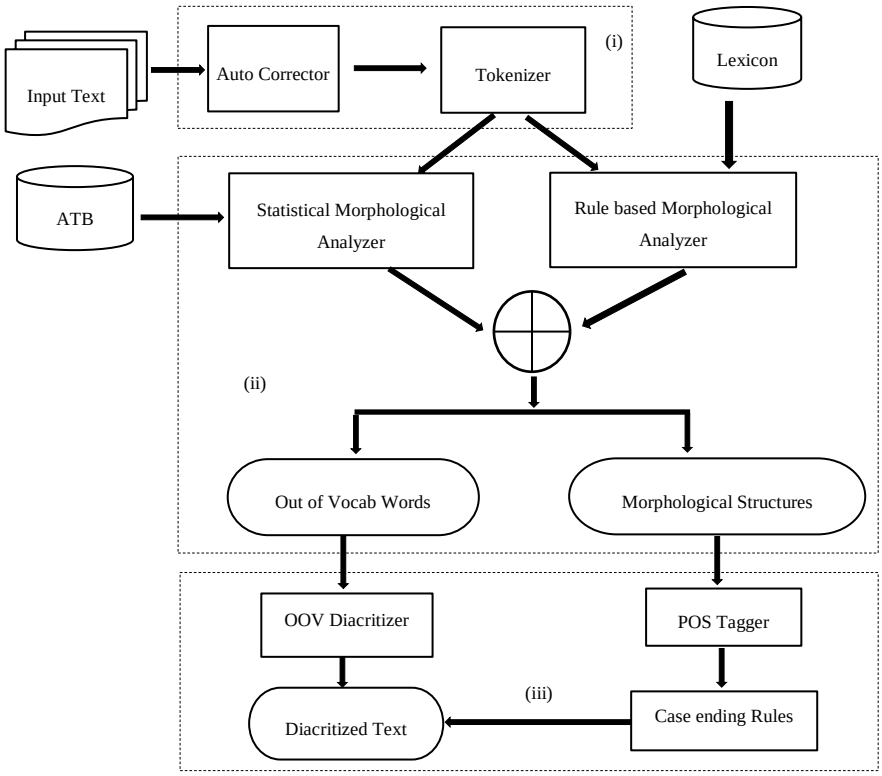

Fig. 1. Architecture of the Arabic Diacritizer.

The next phase is responsible for selecting the most likely sequence of analyses based on the context. This is achieved by the POS tagger, which is presented with a lattice of morphological analyses. In addition to selecting the most probable analysis, this process also disambiguates the residual case ending ambiguities.

The case ending diacritics are resolved in two passes: the first pass is a deterministic one and is driven by rules, and the second pass resolves the residual case ending ambiguities through the POS tagger. Out of Vocabulary (OOV) words, those not analyzed by the morphological analyzers, are diacritized by the out of vocabulary diacritizer (OOV diacritizer) component that works on the character level. The contribution of each component has been measured and the outcomes are reported later on the "Results" section.

4.1 Auto-Corrector

We conducted a thorough analysis of spelling mistakes in Arabic text. A corpus of one thousand articles, picked randomly from public Arabic news sites, has been semi-manually tagged for spelling mistakes. Each spelling mistake has also an associated error type. The analysis has shown a WER of 6% which is considered very high. The diacritization task is significantly affected by this high error rate since the morphological analyzer fails to analyze misspelt words, and hence are left un-diacritized.
Figure 2 shows the distribution of the spelling mistakes in Arabic text according to our analysis. It was found that more than 95% of these mistakes are classified as CAMs [5], and they could be categorized as follows: (i) Confusion between different forms of *Hamza* (آ إ أ) (ii) Missing *Hamza* on plain *Alef* (ا) (iii) Confusion between *Yaa* (ي) and *Alef-Maqsoura* (ى) , and (iv)Confusion between *Haa* (ه) and *Taa-Marbouta* (ة)

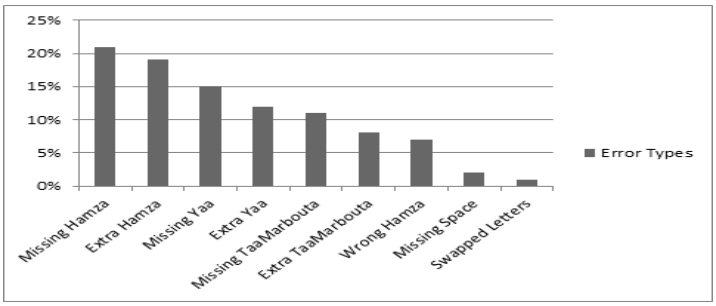

Fig. 2. Distribution of spelling mistakes in Arabic.

Relying on a high-precision and high-recall rule-based Arabic morphological analyzer, the Auto-corrector is able to detect and correct most of the common Arabic mistakes automatically with a precision of 99.53% and a recall of 89.32% as illustrated in figure 3

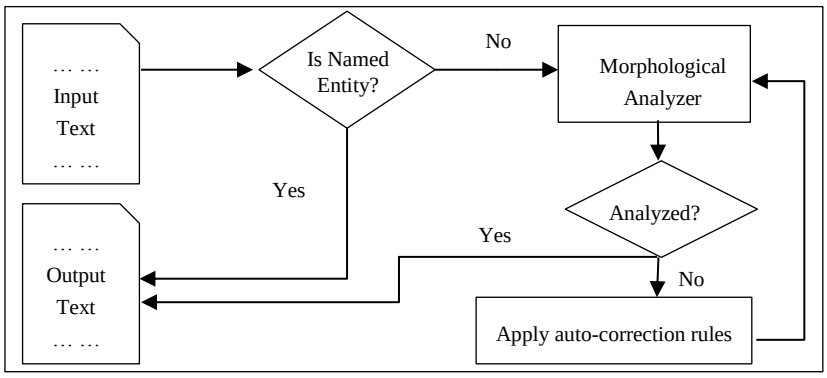

Fig. 3. Autocorrector workflow.

4.2 Statistical Morphological Analyzer

We used ATB corpus [8] to train a statistical morphological analyzer that learns the different possible diacritics, the part of speech tags, and the word tag probability:

$$P(w| t) = count(w,t)/ count(t)$$

Where P(w| t) is the word tag probability.
A major drawback of this model is that it suffers a low coverage since the training size is small, considering a highly-inflectional language such as Arabic. Therefore, we combined its output with the rule-based morphological analyzer output to increase the lexical coverage and resolve the problem of unseen words.

4.3 Rule Based Morphological Analyzer

Any valid Arabic word is represented by a morphological structure that consists of zero or more prefixes, a stem, and zero or more suffixes. Prefixes could be conjunctions, prepositions, determiners, and subject pronouns. Suffixes could be subject pronouns, object pronouns, and possessive pronouns. The Arabic stem is the main part of the word after removing any attached prefixes and suffixes. The stem is usually defined by the tuple (r,m,p) where r is the root (usually three or four characters), m is the morphological pattern, and p is the part of speech. The morphological analyzer has been devised based on a **strip-extract-synthesize** approach driven by a comprehensive lexicon, and handcrafted rules. The rules set consists of (956 rules) to extract lexical stems and restore omitted vowels. The lexicon contains: 46,257 stems, 109 part-of-speeches, 3,472 roots, 691 morphological patterns, and 17,186 prefix-suffix combinations.
Morphological analysis of words is achieved in 3 steps as shown in figure 4. The first phase is the stripping phase, where all possible segmentations of the input word (prefixes-raw stem-suffixes) are assumed as valid hypotheses based on a comprehensive prefix-suffix matrix. Then comes the extraction phase where each raw stem (surface string) is subject to extraction rules, to extract the lexical stem. As an example showing the effect of the extraction rules, the word "سماوات" smAwAt (using Buckwalter transliteration) where the raw stem "سماو" smAw ending with "و" w could be transformed to a lexical stem (real stem) ending with "ء" ', provided that the suffix is an "ات" At. In the 1st two phases both the applicability between the prefixes and the suffixes and between the affixes and the lexical stems, are checked to eliminate invalid hypotheses. For example, assuming the input word is "برق" brq, one possible decomposition would be to consider "ب" b as a preposition prefix and "رق" rq as a past verb stem. However since "ب" b is morphologically incompatible with "رق" rq, then this assumption will be eliminated. The last phase is the synthesis phase, where each remaining hypothesis is subject to synthesis rules, to validate the hypothesis against the

input word. For each valid hypothesis, diacritization rules are applied to restore omitted vowels. For example, if the input word was "مدرسةاً" mdrspAF ending with *Tanween Fatha* the 1st two phases would propose an analysis where the stem equals "مدرسة" mdrsp and the suffix equals "اً" AF, however the synthesis phase would reject this assumption since the synthesis rules would actually generate the word "مدرستاً" mdrstAF converting "ة" p to "ت" t.

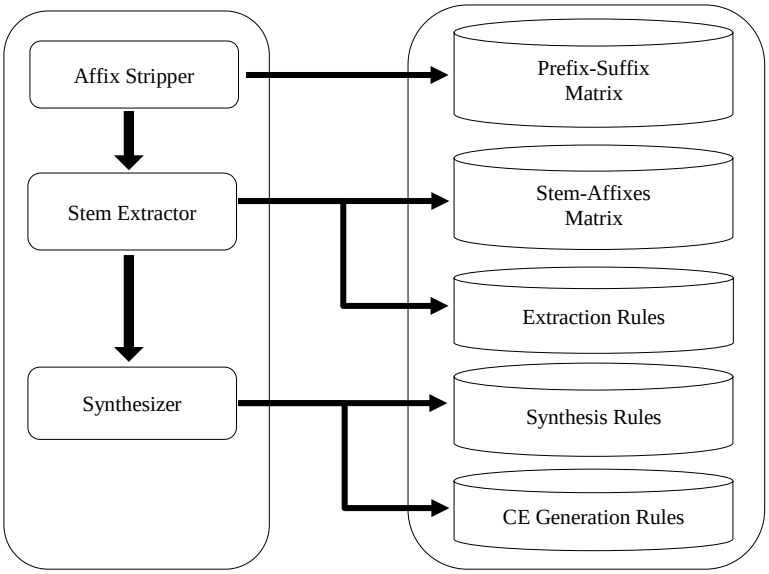

Fig. 4. Morphological Analyzer Architecture.

The synthesizer is also responsible for generating the correct case ending diacritic and to position it on the appropriate letter.

Moreover, the morphological analyzer assigns for each generated morphological structure a set of morphological, lexical and syntactic features. Examples of lexical features are transitivity and verb class, morphological features include definiteness, gender and number and examples of syntactic features are case ending, and genitivity.

4.4 Part of Speech Tagger

Part of speech (POS) tagging is the process of assigning a part-of-speech and optionally other syntactic class markers to each word in a sentence. In the context of diacritization, it is used to resolve the morphological ambiguity for each input word in the Arabic sentence, and by resolving this ambiguity, the associated diacritics, on the stem level get automatically resolved. While the morphological analyzer (described in sections 4.2, 4.3) provides all possible valid analyses for each input word, the POS tagger selects the most likely analysis (diacritics) for each word based on the context, using sequence labeling techniques. We have trained our POS tagger with the ATB

corpus [8], the same training set used by Rashwan et al. [1], Habash and Rambow [2], and Zitouni et al. [3].
While in most of the cases, each analysis is mapped to a different tag, sometimes more than one analysis map to the same tag, in such cases, we pick the first analysis.

Table 2. Example of POS Tagging.

Index	Word	Translation	POS Tag	Diacritics
1	قدمت	Presented	PV+PVSUFF_SUBJ:3FS	قَدَّمَتْ
2	ورشة	Workshop	NOUN+NSUFF_FEM_SG+CASE_DEF_NOM	وَرْشَةُ
3	عمل	Work	NOUN+CASE_DEF_GEN	عَمَلِ
4	الكتاب	Book	DET+NOUN+CASE_DEF_GEN	اَلْكِتَابِ
5	الرقمي	Digital	DET+ADJ+CASE_DEF_GEN	اَلرَّقْمِيِّ
6	لمحة	Insight	NOUN+NSUFF_FEM_SG+CASE_INDEF_ACC	لَمْحَةً
7	عامة	General	ADJ+NSUFF_FEM_SG+CASE_INDEF_ACC	عَامَّةً
8	عنه	About it	PREP+PRON_3MS	عَنْهُ

In addition to the morphological disambiguation, which restores the diacritics on the stem level, the POS tagger also resolves the syntactic ambiguity leading to restoring the syntactic diacritic (case-ending). Table 2 shows the result of POS tagging a short Arabic sentence. The "Index" column in the table is the position of each word in the sentence. The "Word" column is the input word string, the "POS Tag" column is the selected tag by the POS tagger, and each token would initially be presented to the tagger with one or more tag.

The "diacritics" column shows the diacritized word associated with the selected tag, after full disambiguation.
The Hidden Markov Model (HMM) algorithm [11] yields the following function by which the tagger estimates the most probable tag sequence. We used the Viterbi algorithm [11] to compute it. The function contains two kinds of probabilities: word likelihoods and transition probabilities.

$$\hat{t}_1^n \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i|\, t_i) P(t_i|t_{i-1})$$

The POS tagger has been tested on the ATB test set, and it yields accuracy numbers of 86.8% on tag level (including case-ending).

4.5 Out of Vocabulary Diacritizer

OOV words, those for which no single analysis was produced, are classified into 3 main categories: misspelt words, borrowed words (such as "computer" → "كمبيوتر")

and foreign named entities ("John" → "جون"). For these words, the previously described pipeline is still unable to provide any diacritization alternative. In order to solve this class of words, we developed a statistical model to propose diacritics for foreign named entities, and applied this solution to all OOV words. We have built a training set composed of one thousand foreign named entities, which were manually diacritized, and used this set to train a Conditional Random Field (CRF) model [10]. The features used to train the CRF are shape features, and the resulting accuracy using a 10-fold cross validation is 75.7%. We found that using the OOV diacritizer improves the overall diacritization accuracy by 0.9%.

4.6 Case Ending Rules

The case ending rules comprise 30 rules. These are regular expressions, such as "if current word is a non-ambiguous preposition and the next word is noun, then the second word is genitive". Applying these simple rules has contributed in a 0.7 reduction in terms of WER. These rules are manually revised and tuned over several development sets collected from news sources.
The following is an example of a case ending rule that sets the indicative case ending value on present tense verbs:

IF (WORD @POS 0 CONTAIN STEM "بَيْنَمَا" | "عِنْدَمَا") AND (WORD @POS 1 CONTAIN PREFIX #أحرف_أنيت)
STAMP @POS 1 FLAG "مرفوع"

The rule consists of a **CONDITION** part that holds the conditions and an **ACTION** part that carries the actions to be applied whenever the conditions are satisfied.

5 Results

We adopt the same metrics used by Rashwan et al. [1], Habash and Rambow [2], and Zitouni et al. [3]; also we use the same test set they used to compare our results with the three systems. The metrics that were used are:

1. Count all words, including numbers and punctuation.
2. Each letter or digit in a word is a potential host for a set of diacritics.
3. Count all diacritics on a single letter as a single binary choice.
4. Non-variant diacritization (stem level) is approximated by removing all diacritics from the final letter (Ignore Last), while counting that letter in the evaluation.

Two error rates are calculated: the diacritic error rate (DER), which represents the number of letters which diacritics were incorrectly restored, and the WER representing the number of words having at least one DER.

Table 3. Diacritization results, comparing to the best performing systems

Model	Full form		Ignore last	
	WER	DER	WER	DER
Rashwan et al.	12.5%	3.8%	3.1%	1.2%
Habash et al.	14.9%	4.8%	5.5%	2.2%
Zitouni et al.	18.0%	5.5%	7.9%	2.5%
Our System (All Layers)	11.4%	3.6%	4.4%	1.6%
Our System (Disable POST)	40.5%	10.7%	6.5%	2.3%
Our System (Disable Rules)	12.1%	3.8%	No effect	No effect
Our System (Disable OOV diacritizer)	12.3%	4 %	5.2%	2 %

As depicted in Table 3, our system provides the best results in terms of WER on the full form level, and shows comparable results on the other metrics. We found that the test set has a noticeable number of errors such as (misspelt words, colloquial words, undiacritized words, wrong diacritization)[1]. We also tested our system against another blind test set (TestSet2) consisting of 1K sentences that we collected from different sources and had them manually diacritized. Out of this test set, we derived two other test sets: one for full-form diacritization and another one for morphological diacritization. It's worth mentioning here that the way the "ignore last" metric is handled is not always linguistically correct. In many cases, the syntactic diacritics do not show on the last letter and rather appear on the last letter of the stem as in "مَدْرَسَتَهُ". The actual case ending here is the *Fatha* appearing on the before-last letter "ت". In Table 4 below, the first two rows show the results of our system using both the ATB test set, and TestSet2. The remaining rows in the table show the results of our system on TestSet2 after disabling the POS tagger, the case ending rules, the NED, and the auto corrector respectively.

Table 4. Blind test set results

System	Test set	Full form		Morphological Diacritization	
		WER	DER	WER	DER
All Layers enabled	ATB	11.4%	3.6%	4.4%	1.6%
All Layers enabled	TestSet2	8%	2%	2.5%	0.8%
Disable POST	TestSet2	52%	11%	3.2%	1.1%
Disable CE Rules	TestSet2	8.6%	2.3%	No effect	No effect
Disable OOV diacritizer	TestSet2	8.8%	2.4%	3.2	1.2
Disable auto corrector	TestSet2	8.3%	2.2%	2.8	1

[1] The corpus was revised by a set of linguists who reported these errors

6 Conclusion

We presented in this paper a hybrid approach for Arabic diacritics restoration. The approach is combining both rule-based and data driven techniques. Our system is trained and tested using the standard ATB corpus for fair comparison with other systems. The system shows improved results over the best reported systems in terms of full form diacritization. As a future work we speculate that further work on POS tagging and disambiguation techniques such as word sense disambiguation could further improve our morphological diacritization.

References

1. Rashwan, M.A.A., et al.: A stochastic arabic diacritizer based on a hybrid of factorized and unfactorized textual features. Audio, Speech, and Language Processing, IEEE Transactions on 19, 166-175 (2011)
2. Habash, N., Rambow, O.: Arabic diacritization through full morphological tagging. In: NAACL-Short '07 Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers. pp. 53-56 (2007)
3. Zitouni, I., Sorensen, J.S., Sarikaya, R.: Maximum entropy based restoration of arabic diacritics. In: In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL. pp. 577-584 (2006)
4. Habash, N., Rambow, O.: Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In: ACL '05 Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. pp. 573-580 (2005)
5. Buckwalter, T.: Issues in Arabic orthography and morphology analysis. In: In Proceedings of the COLING 2004 Workshop on computational approaches to Arabic script-based languages. pp. 31-34 (2004)
6. Emam, O., Fisher, V.: A hierarchical approach for the statistical vowelization of arabic text. Tech. rep., IBM (2004)
7. Gimnez, J., Mrquez, L.: Svmtool: A general pos tagging generator based on support vector machines. In: LERC'04. pp. 573-580 (2004)
8. Maamouri, M., Bies, A., Buckwalter, T., Mekki, W.: The penn arabic treebank: Building a large-scale annotated arabic corpus. In: Arabic Lang. Technol. Resources Int. Conf.; NEMLAR, Cairo, Egypt (2004)
9. Stolcke, A.: Srilman extensible language modeling toolkit. In: In Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP 2002). pp. 901-904 (2002)
10. Laerty, J.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: the Eighteenth Inter-national Conference on Machine Learning. pp. 282-289 (2001)
11. Jurafsky, D., Martin, J.H.: Speech and Language Processing; an Introduction to Natural Language Processing, Computational Linguistics, and Speech Processing. Prentice-Hall (2000)

View publication stats