Arabic Diacritics in the Wild: Exploiting Opportunities for Improved Diacritization

Salman Elgamal, Ossama Obeid, Tameem Kabbani,[†] Go Inoue,[‡] Nizar Habash
Computational Approaches to Modeling Language Lab, New York University Abu Dhabi
[†]American University of Sharjah
[‡]Mohamed bin Zayed University of Artificial Intelligence
{salman.elgamal,oobeid,nizar.habash}@nyu.edu
b00088948@aus.edu, go.inoue@mbzuai.ac.ae

Abstract

The widespread absence of diacritical marks in Arabic text poses a significant challenge for Arabic natural language processing (NLP). This paper explores instances of naturally occurring diacritics, referred to as "diacritics in the wild," to unveil patterns and latent information across six diverse genres: news articles, novels, children's books, poetry, political documents, and ChatGPT outputs. We present a new annotated dataset that maps real-world partially diacritized words to their maximal full diacritization in context. Additionally, we propose extensions to the analyze-and-disambiguate approach in Arabic NLP to leverage these diacritics, resulting in notable improvements. Our contributions encompass a thorough analysis, valuable datasets, and an extended diacritization algorithm. We release our code and datasets as open source.

1 Introduction

Arabic orthography is infamous for its high degree of ambiguity due to its infrequently used optional diacritical marks. While other Semitic languages like Hebrew and Syriac use similar systems, Arabic has a richer inflectional space with case endings and other orthographic choices that make Arabic more complex. Interestingly, diacritical marks in Arabic are common in limited contexts where correct reading is a goal: holy texts, poetry and children's books, as well as books for adult literacy and non-native learners. But in general reading contexts, for literate Arabic native speaker adults, diacritical marks are used frugally: $\sim$1-2% of words are partially diacritized (Habash, 2010). We refer to these as Diacritics in the Wild (**WildDiacs**).

In this paper, we follow in the footsteps of other researchers to investigate whether such precious occurrences can be exploited to help improve the quality of Arabic NLP tools (Diab et al., 2007; Habash et al., 2016; Bahar et al., 2023). We specifically focus on the Arabic diacritization task, which is at once (a) a final target downstream application given the sparse and variable use of diacritics in Arabic; and (b) an enabling technology for other applications such as speech synthesis (Halabi, 2016).

While the percentage of **WildDiacs** is small, our guiding intuitions are that on large scales, these are objects worthy of study, and given the extra information provided in such contexts, we assume the writers who added them wanted to provide hints to support optimal reading, e.g., to avoid garden path sentences.

For a well-rounded exploration, we analyze and compare the diacritization patterns across six genres: news of multiple agencies, novels, children's books, poetry, political/legal documents of the UN, and ChatGPT output (which sometimes introduces diacritics unprompted). Furthermore, we examine the diacritization patterns and choices in two commonly used datasets for evaluating Arabic diacritization: The Penn Arabic Treebank (Maamouri et al., 2004) and WikiNews (Darwish et al., 2017). We also develop a new annotated dataset that includes instances of partially diacritized words along with their *full* diacritization, which we define carefully, acknowledging different practices. And finally, we propose an extension to the *analyze-and-disambiguate* approach (Habash and Rambow, 2005; Pasha et al., 2014; Inoue et al., 2022) to improve the quality of its choices, and we evaluate on the data we annotated.

Our contributions are the following:

- We provide careful analysis and comparison of diacritization patterns in six genres of Arabic texts, shedding light on the needs and latent information in different Arabic genres.
- We annotate a new dataset for studying maximal diacritization from partial diacritic signals (**Wild2MaxDiacs**), and we extend an existing dataset, WikiNews, to address unhandled phenomena (WikiNewsMaxDiacs).

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 14815–14829

(a)

Fatha	Damma	Kasra	Nunation			Shadda	Sukun	Dagger	Diacritic Clusters…			
بَ	بُ	بِ	بً	بٌ	بٍ	بّ	بْ	بٰ	بُّ	بٌّ	بٍّ	بّٰ
ba	*bu*	*bi*	*bã*	*bũ*	*bĩ*	*b∼*	*b.*	*bá*	*b∼u*	*b∼ũ*	*b∼ĩ*	*b∼á*
ba	*bu*	*bi*	*ban*	*bun*	*bin*	*bb*	*b*	*bā*	*bbu*	*bbun*	*bbin*	*bbā*

(b)

وَالشُّمُوسُ السَّاطِعَةُ

waAlš∼umuwsu Als∼aATiςaħu

waš šu mū su-s sā Ti ςa tu

Figure 1: (a) The nine Arabic diacritics commonly used in Modern Standard Arabic, grouped by function; and four examples of diacritic clusters. (b) A visually annotated example of a diacritized phrase meaning 'and the bright suns [lit. and-the-suns the-bright]'. Diacritics are marked in red; and so are the undiacritized vowel-lengthening helping letters. Silent letters appear in dotted boxes.

- We extend a hybrid (neuro-symbolic) algorithm for Arabic diacritization to make use of the existence of **WildDiacs**, and demonstrate improved performance.

Our code and datasets will be open-source and publicly available.[1]

The paper is organized as follows. Section 2 discusses Arabic diacritics. Section 3 reviews related work. Section 4 covers the datasets, including our annotated datasets, and **WildDiacs** usage statistics. Section 5 details the approach we build on and how we extend it. Section 6 presents the experimental setup, evaluation, and error analysis.

2 Arabic Diacritics

We present an overview of Arabic diacritics in terms of their form, function, and use.

2.1 Arabic Diacritic Forms

Arabic diacritics are zero-width characters that adorn Arabic letters to supplement Arabic's Abjad orthography with additional phonological signals. There are many diacritical marks in the Arabic script: Unicode currently boasts 52 such symbols.[2] However, the basic *Tashkil* (diacritization) set used in most Modern Standard Arabic (MSA) contexts includes nine symbols. See Figure 1 (a).[3] In this paper, we focus on these MSA diacritics, which are all readily accessible by most Arabic keyboards.

Diacritic clusters can occur but are highly constrained. The Shadda (◌ّ ∼) can combine with any one of the other diacritics, and none of them can combine with each other in MSA, except for the Dagger Alif (◌ٰ *á*), which can follow Fatha (◌َ *a*). While in proper Arabic spelling the Shadda character should appear first in the string (and writing order) as it indicates doubling of previous consonant letter, it is important to note that a flipped order (e.g., Shadda after vowel) is impossible to detect visually. We find both orders in the wild. For example *kat∼ab* and *kata∼b* will always be rendered to appear as *kat∼ab* (كَتَّب).[4]

2.2 Arabic Diacritic Functions

Basic Phonological Mapping The diacritics primarily denote phonological information that supplements the Arabic Abjad orthography: vowel diacritics (Fatha, Damma, Kasra) indicate the presence of a short vowel; nunation (تنوين *tanwiyn*) diacritics indicate a short vowel followed by /n/; the gemination diacritic, Shadda, indicates doubling of the consonant letter it follows; the Sukun (silence) diacritic indicates that no vowel is present; and finally, the special elongation diacritic (aka Dagger Alif) indicates a long /ā/ vowel. See the red colored diacritics and their mapping to phonology in Figure 1 (b).

An important and a not so obvious detail about the use of diacritics is that even under *maximal diacritization*, some letters remain *bare*, i.e., with no diacritics, to indicate specific phonological information. Examples include (i) *Elongation*: the *weak* letters (ا *A*, و *w*, ي *y*) remain bare when used as helping letters to elongate short vowel diacritics (Fig. 1 (b)'s red letters); and (ii) *Silence*: letters that

[1] https://github.com/CAMeL-Lab/wild_diacritics

[2] https://unicode.org/charts/PDF/U0600.pdf

[3] Arabic transliteration is in the HSB scheme (Habash et al., 2007): ' ء, Ā آ, Â أ, ŵ ؤ, Ǎ إ, ŷ ئ, A ا, b ب, ħ ة, t ت, θ ث, j ج, H ح, x خ, d د, ð ذ, r ر, z ز, s س, š ش, S ص, D ض, T ط, Ď ظ, ς ع, γ غ, f ف, q ق, k ك, l ل, m م, n ن, h هـ, w و, ý ى, y ي, *ã* ◌ً, *ũ* ◌ٌ, *ĩ* ◌ٍ, *a* ◌َ, *u* ◌ُ, *i* ◌ِ, ∼ ◌ّ, . ◌ْ, *á* ◌ٰ, Ä ٱ.

[4] Most text rendering libraries that support Arabic implement the Arabic Mark Transient Reordering Algorithm (AMTRA) which normalizes the display of diacritic clusters (Pournader et al., 2021). Furthermore, many systems utilize Unicode Normalization Forms (UNFs) (Whistler, 2023b) which, among other things, order adjacent diacritics based on their *combining class* property (Whistler, 2023a). UNFs are important for string matching and lexicographic sorting despite diacritic order variability. These robustness-supporting mechanisms inadvertently allow inconsistent uses to coexist freely in the wild.

	(a)	(b)	(c)	(d)	(e)	(f)
Arabic	اليوم	أشرقت	الشمس	الساطعة	من	الغرب
English	*today*	*rose*	*the-sun*	*the-bright*	*from*	*the-west*
Undiacritized	*Alywm*	*Âšrqt*	*Alšms*	*AlsATςħ*	*mn*	*Alγrb*
Partial Diacritization	*Alyawm*	*Âšraqt*	*Alšmsu*	*AlsATςħ*	*min*	*Alγrb*
ATB Diacritization	*Alyaw.ma*	*Âaš.raqat*	*Alš~am.su*	*Als~ATiςaħu*	*min*	*Alγar.bi*
WikiNews Diacritization	*Al.*[1]*yaw.ma*	*Âaš.raqat.*[2]	*Alš~am.su*	*Als~a*[3]*ATiςaħu*	*min.*[2]	*Al.*[1]*γar.bi*
Maximal Diacritization	*Aa*[4]*l.yaw.ma*	*Âaš.raqati*[5]	*Alš~am.su*	*Als~aATiςaħu*	*mina*[5]	*Al.γar.bi*
Phonology	*'al yaw ma*	*'aš ra qa ti-*	*-š šam su-*	*-s sā Ti ςa tu*	*mi na-*	*-l γar bi*

Figure 2: An example in different levels of diacritization. The red underlined diacritics highlight changes from row to row. ATB is the Arabic Treebank diacritization standard (Maamouri et al., 2004), an essential full diacritization. WikiNews (Darwish et al., 2017) addresses some ATB gaps like missing Sukuns (1,2) and long vowel marking with *aA* (3). Maximal diacritization adds contextual diacritics in word-initial (4) and inter-word contexts (5).

are phonologically elided or assimilated are marked by leaving them bare of diacritics (Fig. 1 (b)'s grey letters).[5,6]

Dimensions of Disambiguation We can categorize the functional use of diacritics based on the disambiguating information they provide to the reader, which includes lexical, morphosyntactic, and contextual phonological liaison (*sandhi*). For example, the word من *mn* can be diacritized in different ways: مِنْ *min.* 'from', مَنْ *man.* 'who', مَنَّ *man~a* 'he granted', مَنٌّ *man~ũ* 'a favor [indef. nom]', among many others. All four instances show lexical diacritics. The last two words' final diacritics indicate morphosyntactic features such as verb aspect-person-gender-number, and nominal case and state. In specific phonological contexts, some helping epenthetic vowels are introduced, and others may be elided. The typical epenthetic vowel is ◌ِ *i*, but there are other special cases: For example, *min.* 'from', has two additional forms: *mina* before words starting with the definite article (see Fig. 2), and *mini* in general epenthesis, e.g., مِنِ ابْنِهِ *mini Ab.nihi* 'from his son'.

[5] A very common example is the ل *l* of the definite article ال *Al* when followed by a *Sun Letter* – a coronal consonant with which the *l* assimilates, e.g., *š* and *s* in Fig. 1(b). A Shadda on the following letter indicates assimilated gemination.

[6] The Alif (ا *A*) letter is used in word-initial positions as a vowel carrier (i.e., Alif Wasla, aka Hamzat Wasl). When the vowel is elided in context, the Alif is retained but kept bare, e.g., cases of grey ا *A* in Fig. 1(b). Quranic Arabic retains the Wasla diacritic ٱ *Ä* to mark this absence, but MSA does not use it any more, which unavoidably leads to a minor ambiguity between silence and elongation. We internally model the Alif Wasla as it interacts with other diacritization decisions, but convert it to a bare Alif in the output. See Appendices E and F.

2.3 Arabic Diacritics in the Wild

The two main challenges concerning Arabic diacritics in NLP are *incompleteness* and *inconsistency*.

The Challenge of Incompleteness Arabic diacritics are quite often omitted, with around 1-2% of words in news text having at least one diacritic. Arabic's rich templatic morphology, common default syntactic orders, and contextual semantic resolutions explain why educated Arabic readers can read Arabic with such a signal deficit. Not all texts and genres are equally devoid of diacritics. For Quranic Arabic, and to a lesser extent poetry, diacritics are almost always included to avoid any misreading of the texts. Similarly for children's educational materials, diacritics are included to assist in learning. In this paper we study a number of genres to help understand how our approach will function under different conditions.

We define the terms **undiacritized** and **dediacritized** to refer to words with no diacritics, or stripped of all diacritics, respectively. We will use the term **fully diacritized** to refer to a number of standards that have been used to evaluate automatic diacritization processes (Fig. 2 ATB and WikiNews). We reserve the term **maximally diacritized** to the version we target in this paper as a higher standard of completeness (Appendix D). **Partially diacritized** refers to a state of in-betweenness where some diacritics are provided. **WildDiacs** are typically partial diacritizations.

The Challenge of Inconsistency We note two types of inconsistency in diacritic use. First, there are a number of acceptable variations that reflect different styles. These appear in the wild, as well as in commonly used evaluation datasets. Examples include (i) foreign names whose diacrtization reflect local pronunciation such as بريطانيا 'Britain' as *b.riTaAniyaA* or *biriTaAniyaA*; (ii) the inclusion of the Sukun (silence diacritic) at the end of utterance-final words; and (iii) the position of the Tanwiyn Fath before or after a word-final silent Alif or Alif-Maqsura such كِتَابًا *kitaAbãA* or كِتَاباً *kitaAbAã* 'a book'.

The second type are simply errors. Examples include (i) the Shadda appearing after the vowel diacritic, e.g., *kata∼b* instead of the correct *kat∼ab* 'he dictated'; (ii) multiple incompatible diacritics on the same letter, e.g., *ktAbuu* instead of *ktAbũ*; (iii) diacritics in impossible positions such as word initial *ĩktAb*; and (iv) the correct diacritic is on the incorrect letter, e.g., *ktiAb* for *kitAb*.

To catch some of these inconsistencies, we developed a well-formedness check script and used it as part of this paper. Details are in Appendix B.

Despite their imperfections, **WildDiacs** are useful human annotations that not only aid other human readers, but can be exploited automatically.

3 Related Work

3.1 Roles of Diacritics in Arabic NLP

Diacritics play a significant role in a wide variety of NLP tasks, such as text-to-speech synthesis (Ungurean et al., 2008; Halabi, 2016), automatic speech recognition (Aldarmaki and Ghannam, 2023), machine translation (Diab et al., 2007; Alqahtani et al., 2016; Fadel et al., 2019b; Thompson and Alshehri, 2022), morphological annotation (Habash et al., 2016), homograph disambiguation (Alqahtani et al., 2019), linking lemmas across different lexical resources (Jarrar et al., 2018), language proficiency assessment (Hamed and Zesch, 2018), and improving text readability (Esmail et al., 2022; ElNokrashy and AlKhamissi, 2024).

Several attempts have explored the impact of varying degrees of diacritization in downstream tasks. Diab et al. (2007) and their follow-up work (Alqahtani et al., 2016) investigate the impact of various degrees of diacritization to identify the optimal amount of diacritics for better machine translation performance. Habash et al. (2016) observe a positive correlation between the degree of diacritization in the input text and the performance in the morphological annotation task.

Using naturally occurring diacritics in the input text is shown to be effective in the diacritization task itself. AlKhamissi et al. (2020) propose a model that accepts partially diacritized text during decoding, demonstrating improved diacritization performance. With a similar motivation, Bahar et al. (2023) introduce a bi-source model that takes both characters and optional diacritics available in the input sequence. Our work differs from theirs in that no training is required to leverage the presence of diacritics in the input text, making our approach computationally efficient.

3.2 Datasets for Diacritization

Numerous datasets have been developed for diacritization in different variants of Arabic, such as MSA (Maamouri et al., 2004; Darwish et al., 2017), classical Arabic (Zerrouki and Balla, 2017; Yousef et al., 2019; Fadel et al., 2019a), and dialectal Arabic (Jarrar et al., 2016; Abdelali et al., 2019; El-Haj, 2020; Alabbasi et al., 2022). The source of datasets varies, including news, e.g., the Penn Arabic Treebank (ATB) (Maamouri et al., 2004) and the WikiNews dataset (Darwish et al., 2017), classical books (Tashkeela; Zerrouki and Balla, 2017), and poetry (APCD; Yousef et al., 2019). Among these resources, ATB and WikiNews are widely used as the standard benchmark datasets for diacritization in MSA.

Annotation conventions vary across datasets due to the lack of consensus in definitions, and may even be inconsistent within a dataset (Darwish et al., 2017). In this work, we thoroughly analyze and compare diacritization patterns in widely used datasets, highlighting the need for an evaluation set based on *maximal diacritization*—a refined definition of diacritization. We also provide a new annotated dataset based on this paradigm comprising 6,000 words across six different genres.

3.3 Automatic Diacritization

Approaches to automatic diacritization vary in task formulation, architecture choice, and the use of external resources. One line of work formulates diacritization as a single isolated task, e.g., a character-level sequence labeling problem (Zitouni et al., 2006, among others) and a sequence-to-sequence problem (Mubarak et al., 2019). Early efforts employ traditional machine learning models such as the maximum entropy classifier (Zitouni et al., 2006) and SVMs (Darwish et al., 2017). Recently, neural models have shown significant progress, such as LSTMs and their extensions (Abandah et al., 2015; Belinkov and Glass, 2015; Fadel et al., 2019b; Madhfar and Qamar, 2021; Darwish et al., 2021; Elmallah et al., 2024) and Transformer-based models (Mubarak et al., 2019; Qin et al., 2021; Elmadany et al., 2023).

Another line of work addresses diacritization within a multi-task setup, jointly modeling diacritization with relevant tasks such as POS tagging (Habash and Rambow, 2005; Alqahtani et al., 2020, among others) and machine translation (Thompson and Alshehri, 2022). A large body of this kind has demonstrated the value of using external resources such as a morphological analyzer. They adopt an *analyze-and-disambiguate* strategy, where they generate possible analyses for each word with a morphological analyzer (*analyze*), then rank the analyses based on separately trained feature classifiers (*disambiguate*). This approach has been extensively explored using various architectures, including SVMs (Habash and Rambow, 2005, 2007; Roth et al., 2008; Pasha et al., 2014; Shahrour et al., 2015), LSTMs (Zalmout and Habash, 2017, 2019, 2020), and Transformer-based models (Inoue et al., 2022; Obeid et al., 2022). In this work, we present an extension to this approach where we utilize the presence of naturally occurring diacritics in the input text to improve the re-ranking of the analyses without additional training.

4 Diacritics: Data and Statistics

4.1 Datasets

We use eight pre-existing datasets (two fully diacritized and six partially diacritized), and we introduce two new maximally diacritized datasets that we make publicly available.

Fully Diacritized Datasets We include the WikiNews (Darwish et al., 2017) and ATB (Maamouri et al., 2004) datasets, which have been used extensively in past recording on Arabic diacritization (Darwish et al., 2017; Pasha et al., 2014; Mubarak et al., 2019). As mentioned in Sec. 2.3 and Fig. 2, WikiNews' annotation guideline lead to a fuller diacritization than those of ATB's essential guidelines. In Table 1, we use all the text of WikiNews, and the *training* data portion of ATB Parts 1-3 (Diab et al., 2010).

Partially Diacritized Datasets To study the variation in **WildDiacs** patterns in different genres, we targeted six datasets covering News, Poetry, Novels, Children's books, UN, and ChatGPT. The **Children** and **Novels** set are from the Hindawi website[7] (Al Khalil et al., 2018). The **Poetry** samples are from the Arabic Poem Comprehensive Dataset (APCD) (Yousef et al., 2019). The **UN** data is from the Arabic portion of the UN Corpus (Ziemski et al., 2016). The **News** data is from Arabic Gigaword (Parker et al., 2011), specifically from Asharq Alawsat (aaw) and AlHayat (hyt). We intentionally did not use the sources used in building the ATB corpus (afp, umh, nhr, asb) since ATB was used to train the baseline model we used in our experiments, and we did not chose Ahram (ahr) or Xinhua (xin) because of their very low percentage of **WildDiacs**. Finally, for **ChatGPT**, we use the dataset released by Alhafni et al. (2023) as part of their work on grammatical error correction, where they prompted ChatGPT-3.5 to correct undiacritized raw Arabic text and ChatGPT interestingly added seemingly random partial diacritizations. We randomly sampled 50,000 lines from Poetry, UN, and News due to their large size, while we used the whole datasets for the rest of the genres.[8]

Maximally Diacritized Datasets We introduce two new datasets. The first is the **Multi-genre Wild Diacritization to Maximal Diacritization** (**Wild2MaxDiacs**) dataset. **Wild2MaxDiacs** is composed of randomly chosen 6,000 words with wild partial diacritization, 1,000 from each of the six genres discussed above. For each selected word, given its full context, we provide a maximally diacritized version of it. The annotations were carried out by two Arabic native speakers. The annotators closely followed our definition of in-context full diacritization well-formedness. The annotations were passed through our well-formedness checks as an extra pass of validation and the words that failed our check were corrected. **Wild2MaxDiacs** is split into equal 3,000-word development (Dev) and Test sets.

The second dataset is **WikiNewsMaxDiacs**, a new version of WikiNews that we manually extended to our maximal diacritization, affecting 3.5% of its tokens. The main extensions are adding contextual diacritics and Dagger Alifs. We also, for the first time to our knowledge, annotated the dataset with valid alternative diacritizations (2.1% of tokens), such in the case of foreign names discussed above.

4.2 Statistics

Table 1 presents different diacritic usage statistics.

[7] https://www.hindawi.org/

[8] All texts are processed with CAMeL Tools' simple word tokenizer (Obeid et al., 2020).

		Full Diacritization		Partial Diacritization					
		WikiNews	ATB	Children	Poetry	Novels	UN	News	ChatGPT
(a)	# Lines	400	19,738	30,468	50,000	165,005	50,000	50,000	4,485
(b)	# Arabic Words	16,215	540,329	533,524	464,764	4,737,226	869,119	1,416,681	268,081
(c)	% Lines with any Diac	100.0	99.6	81.4	81.2	50.8	15.6	13.9	58.1
(d)	% Words with any Diac	100.0	96.9	82.6	53.8	5.6	1.4	1.3	5.3
(e)	# Diac per Diacritized Word	4.0	3.4	3.2	2.1	1.4	1.2	1.1	1.9
(f)	% Max Diac Words	97.0	41.1	53.1	7.4	0.2	0.0	0.0	1.0
(g)	% Tanwiyn Fath (◌ً ã)	0.8	1.1	1.6	2.8	37.0	39.1	62.8	18.8
(h)	% Tanwiyn Dam (◌ٌ ũ)	0.4	0.4	0.7	1.8	1.5	0.2	0.6	1.8
(i)	% Tanwiyn Kasr (◌ٍ ĩ)	1.5	1.6	1.2	2.4	2.6	1.2	2.4	2.2
(j)	% Fatha (◌َ a)	39.9	35.9	42.4	43.2	19.0	8.5	3.6	30.1
(k)	% Damma (◌ُ u)	9.4	11.0	11.7	16.7	11.3	22.3	5.7	13.2
(l)	% Kasra (◌ِ i)	24.5	29.3	20.3	19.7	7.8	5.2	2.0	15.8
(m)	% Shadda (◌ّ ~)	6.9	9.3	6.5	8.5	17.0	23.2	22.1	9.6
(n)	% Sukun (◌ْ .)	16.7	10.9	15.6	4.9	3.7	0.4	0.8	8.5
(o)	% Dagger Alif (◌ٰ á)	0.0	0.4	0.0	0.0	0.0	0.0	0.0	0.1
(p)	*% Correlation with WikiNews*		*97.3*	*99.0*	*92.2*	*12.4*	*-15.5*	*-28.6*	*78.5*
(q)	% Canonical Diac Clusters	100.0	100.0	99.4	100.0	99.9	99.8	99.1	97.3
(r)	% Tanwiyn Alif Order	99.7	0.0	99.9	1.0	93.3	0.4	0.5	59.8
(s)	% Shadda Vowel Order	100.0	100.0	91.4	99.9	99.4	99.9	59.6	65.1

Table 1: Statistics of diacritic usage in fully diacritized and partially diacritized datasets.

Diacritization Completeness Across Genres Table 1 (c-d) highlight the great variation in use of diacritics across different genres. The fully diacritized datasets have almost complete diacritization by design. The columns in Table 1 are ordered to reflect the pattern of completeness (from left to right), except that ChatGPT, being artificial, is separated at the end. As expected, the degree of completeness in naturally occurring partially diacritized datasets is highest for **Children** text, followed by **Poetry**, **Novels**, **UN** documents and finally **News**. **ChatGPT** has an interestingly high level of **WildDiacs** comparable to Novels. We also observe that the number of diacritics per diacritized word correlates highly with the percentage of words with diacritics and the percentage of fully diacritized words—Table 1 (d-f). The low number of fully diacritized words for ATB and lower than perfect for WikiNews reflect the systematically missing diacritics in their annotations (see Section 2.3).[9]

Diacritization Patterns Across Genres Table 1 (g-o) shows the distributions of the nine diacritics (in Unicode order) across the datasets relative to the total number diacritics. Table 1 (p) presents the correlation of the diacritic distributions of each genre against the WikiNews distribution (fullest in diacritization). Interestingly, we note that the genres with high percentages of diacritics are highly correlated with full diacritization; but as percentages drop, the ratio of lower frequency diacritics are more common (and the correlation becomes negative). This makes sense as it suggests **WildDiacs** fill in the gaps for the low frequency events. This is most evident when comparing **News** to **WikiNews** (same genre but different degrees of completeness): Sukuns drop in News to 1/20 of their WikiNews distribution; and Tanwiyn Fath increases by over 70 times. We also note that the ChatGPT correlation pattern with WikiNews is oddly different from all naturally occurring partial diacritization datasets. Finally, we note that the use of Dagger Alif is almost extinct across MSA; and it seems to mostly be present in the ATB tokenization. This is the only case where ATB has more details than WikiNews. We include Dagger Alif in our Maximal Diacritization.

Wild Diacritization Failures Table 1 (q) reports that almost all of the diacritized words pass our well-formedness check, with ChatGPT performing the worst by far. Table 1 (r) shows that the Alif-Tanwiyn order is split between two schools: *ãA* in WikiNews, Children and Novels; and *Aã* in ATB, Poetry, UN and News. ChatGPT produces a mix of the two approaches. And finally, according to Table 1 (s) the Shadda-Vowel order is almost perfectly stable, except in News and ChatGPT, where there is a lot of noise.

[9] The main issues in ATB diacritization are (i) no *a* diacritic before Alif when indicating long vowel /ā/ , (ii) no Sukun to mark the definite article with moon letters and some foreign name consonant clusters, and (iii) no contextual diacritics.

	Children	Poetry	Novels	UN	News	ChatGPT	All
Lexical	99.4	75.2	31.6	29.4	25.4	56.4	52.9
CasStt	44.6	47.4	57.6	46.6	67.8	53.8	53.0
PGNMA	48.2	22.2	18.8	0.8	2.6	16.2	18.1
VPass	0.4	1.2	3.4	24.8	4.2	2.0	6.0
Context	2.2	0.4	0.0	0.0	0.0	0.6	0.5
Error	0.4	3.8	2.8	1.0	1.2	7.0	2.7

Table 2: Percentage of words where **WildDiacs** indicate Lexical specification, CasStt (case, state), PGNMA (person, gender, number, mood, aspect), VPass (passive voice), and contextual diacritics, or are errors.

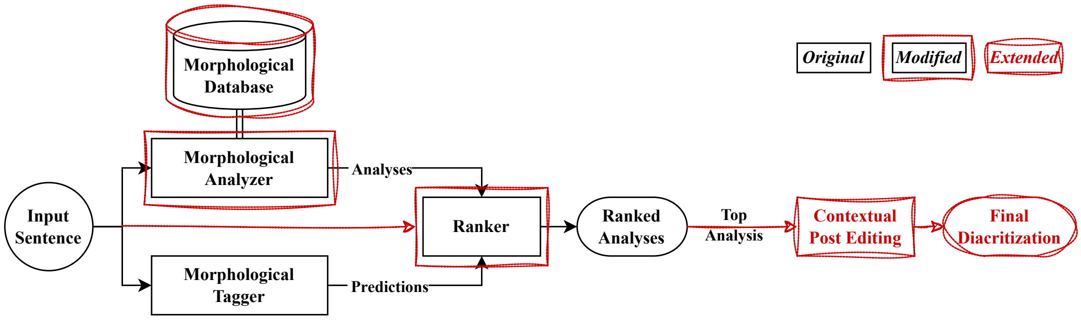

Figure 3: The original morphological analysis and disambiguation process with modifications and extensions.

Functional Use of Wild Diacritization Finally, we report on the distribution of functional uses of **WildDiacs** in Table 2. We annotated the 3,000 words in the **Wild2MaxDiacs** Dev set across all six genres for six categories: lexical, case/state, person/gender/number/mood/aspect (PGNMA), passive verb, context diacritics, and errors. Lexical and case/state are the highest in frequency overall, followed by PGNMA then voice. The most dominant genres per category are: lexical, PGNMA and context (Children), case/state (News, 94% Tanwiyn Fath), passive voice (UN), and Errors (ChatGPT). The diversity and patterns of functional use focus are consistent with our expectations, although not previously reported to our knowledge.

5 Exploiting Diacritics in the Wild

Our basic approach to exploiting **WildDiacs** is based on the intuition that they are simply *hints* that we need to *take* (Bahar et al., 2023). We build on a hybrid (neuro-symbolic) *analyze-and-disambiguate* solution for Arabic diacritization implemented in CAMeL Tools (Obeid et al., 2020). While this open-source toolkit reports competitive results (Obeid et al., 2020, 2022), it has a number of limitations connected to its dependence on the-less-than-ideal ATB diacritization (training and tools). We present the basic approach followed by our modifications and extensions.

5.1 The Analysis & Disambiguation Process

Figure 3 shows the basic CAMeL Tools disambiguation process (in black components) and our extensions and modifications of it (in red components). First, for each word in the input sentence, a list of analyses is generated by a symbolic morphological analyzer. These analyses are comprised of a number of morphological features (e.g., part-of-speech, gender, case), a number of lexical features (e.g., lemma, diacritized orthography), and a number of pre-computed statistical features (e.g., log probability of lemma). In parallel, a neural morphological tagger predicts a set of morphological features (a subset of features provided in analysis) for each word in context.[10] Finally, the analyses are ranked using the tagger's predictions by sorting on the number of matching morphological features per analysis, then breaking analysis ties with the following analysis features in order: the joint POS-lemma log probability, the lemma log probability, and finally sorting lexicographically on the analysis diacritization, to ensure a final stable ranking.

5.2 Modifications & Extensions

Re-ranking with WildDiacs We modified the basic ranker to use Levenshtein insertion, deletion,

[10]We report here on the Unfactored BERT Disambiguator model (Inoue et al., 2022).

	Algorithm	Context	Database	Children	Poetry	Novels	UN	News	ChatGPT	ALL
(a) Development	**Oracle**	**None**	**Current**	47.8	46.2	62.0	62.4	64.8	57.2	56.7
	Oracle	**Solo**	**Extended**	77.6	68.2	87.8	83.0	89.2	80.6	81.1
	Oracle	**Full**	**Extended**	93.2	84.6	94.0	92.0	96.0	96.6	92.7
	CT	**None**	**Current**	39.8	35.8	52.2	52.6	59.8	50.8	48.5
	CT	**Solo**	**Extended**	67.2	49.6	73.8	70.8	83.4	72.8	69.6
	CT	**Full**	**Extended**	81.8	63.0	78.8	77.2	89.6	88.2	79.8
	CT++	**Solo**	**Extended**	76.4	64.4	82.4	76.6	86.8	76.2	77.1
	CT++	**Full**	**Extended**	**91.8**	**80.0**	**88.4**	**84.8**	**93.2**	**91.0**	**88.2**
			OOV	2.2	4.4	2.4	1.4	0.8	0.4	1.9
(b) Test	**Oracle**	**Full**	**Extended**	94.8	87.2	93.6	93.8	95.8	95.4	93.4
	CT	**None**	**Current**	42.8	35.8	54.4	51.0	59.2	51.8	49.2
	CT++	**Full**	**Extended**	**93.8**	**81.2**	**88.2**	**89.4**	**91.2**	**89.6**	**88.9**
			OOV	1.8	2.4	0.6	0.6	0.6	1.4	1.2

Table 3: Percentage of correctly maximally diacritized words from **Wild2MaxDiacs** dataset: (a) Dev and (b) Test. For Dev, all possible system combinations are presented, except for impossible combinations, e.g., the Current DB with the Context Full is meaningless since the Context Full requires information not in the Current DB. For Test, only the best setup identified by the Dev set, baseline and oracle are presented.

and substitution edits between input word and analysis diacritization. Empirically, our best setup is sorted with substitutions+deletions, followed by morphological features, substitution alone, deletion alone, POS+lemma log probability, lemma log probability, insertions, then diacritization lexicographic order. This order guarantees ideal behavior when the input is fully diacritized, and is backward compatible with the original ranker when the input is not diacritized. See more details in Appendix C.

Morphological Analyzer Modifications Our approach depends heavily on the choices produced by the morphological analyzers. The baseline analyzer we use is based on SAMA/Calima-Star (Graff et al., 2009; Taji et al., 2018) which follow the ATB full diacritization standard. As such, we make a number of changes to the morphological analyzer's algorithm and database to accommodate maximal diacritization (Footnote 9). Examples include (i) adding a missing Fatha before Alif for long vowel representation, (ii) fixing the position in A/ý word-final Tanwiyn Fath, (iii) adding missing Sukuns in word final and medial locations, (iv) enhancing the database's Alif Wasla coverage to include unmodeled cases, e.g., the determiner آل *Äal*, and (v) adding string flags to mark cases with allomorphic variants, e.g., مِن *min* is marked as *min%n* (Section 2.3). For more details, see Appendix E.

Contextual Post Edit Extensions After analysis re-ranking, the top analysis for each token is selected, and an array of all top analyses are passed through a new component that handles contextual post edits. The sequence of regex-based edits perform inter-word changes that depend on surrounding contexts and allomorphic flags introduced in the morphological analyzer. For example, the edits will transform the sequence هُمm% آلْحُبّ *hum%m Äal.Hub∼u* 'they are love' to هُمُ الْحُبّ *humu Al.Hub∼u*. For more details, see Appendix F.

6 Evaluation

6.1 Experimental Setup

We evaluate our enhanced model in various settings to assess the impact of different extensions. Regarding the **ranking algorithm**, we compare the CAMeL Tools baseline (**CT**) with our extended re-ranking approach (**CT++**). Additionally, we conduct an **Oracle** evaluation to determine the upper limits of both versions by selecting the analysis closest to the gold diacritization for each word. For **context modeling**, we examine three scenarios: no modeling (**None**), modeling as standalone utterances (**Solo**), and full context (**Full**). Regarding the **morphological analyzer database**, we compare the **Current** CAMeL Tools database with our extended version (**Extended**). We assess performance across the Dev and Test sets of **Wild2MaxDiacs**'s six genres, without using any **Wild2MaxDiacs** data for training. Additionally, we evaluate the full system on **WikiNewsMaxDi-**

Error Type	Ratio	Post-context	Input Word	Pre-context	Gold Ref	Best Model
Missing Analysis	55%	.	العَرمَرَمِ *Alςarmarami*	إِذا غِبتَ أَو صَدرُ الخَميسِ	الْعَرَمْرَمِ *Al.ςaram.rami*	الْعرمرم *Al.ςrmrm*
Tagging Failure	22%	زينَةُ الحَفلِ الَّذي أَنتَ زَينُهُ	وَمَن *waman*		وَمَنْ *waman.*	وَمَنِّ *waman∼i*
Wrong Input	11%	في البلقاء فاستقدمهم	أهلهُ *Âhlhu*	نخاف على	أَهْلِهِ *Âah.lihi*	أَهْلَهُ *Âah.lahu*
Gold Ref Error	11%	أيضا من عائلة صفا	يُطلب *yuTlb*	تمرّ دقائق قبل أن	يُطْلَبُ *yuT.labu*	يُطْلَبَ *yuT.laba*
Valid Alternative	1%	إلا القشور وغير ذاك دفين	السّقم *Als∼qm*	تالله ما عرف الطّبيب من	السَّقَمِ *Als∼aqami*	السُّقْمِ *Als∼uq.mi*

Table 4: Analysis of different error types in order of severity. Alongside each error type, we include a typical example from the Dev set. The *missing analysis* error example is also classifiable as a *wrong input* due to the extra Fatha on its fifth letter; however, we only consider it here as a missing analysis error since the system could still recover from this wrong input error if an analysis was present.

acs without partial diacritization to demonstrate the added value of our extensions. Our metric is strict word diacritization matching accuracy.

6.2 Results and Discussion

Wild2MaxDiacs Results Table 3 shows a result breakdown on **Wild2MaxDiacs** dataset. The results are consistent across genres, and both Dev and Test. Under All Dev, the morphological database modifications improved accuracy by 21.1% over the baseline model, and the contextual post-editing improved accuracy further by 10.2%. With all modifications applied, we achieve a 39.7% improvement in total, and we achieve close to 95.1% accuracy relative to Oracle.

We analyzed all the errors in the Dev set (n=354). As shown in Table 4, 55% are due to missing analyses in the morphological analyzers, and 22% are due to failures in morphological tagging. The rest of the errors are either wrong **WildDiacs** input (11%) or gold reference error (11%). There were very few cases of multiple different valid diacritizations (1%). These patterns were comparable across genres.

WikiNewsMaxDiacs Results Our modified system increases the strict matching accuracy of word diacritization from dediacritized input on original **WikiNews** (Darwish et al., 2017) from 47.1% (CAMeL Tools baseline) to 84.5% (our best system). The corresponding results on our newly annotated multi-reference **WikiNewsMaxDiacs** improve from 47.5% (CAMeL Tools) to 89.7% (our best system). 64% of the increase in the best system is due to improved basic gold reference in WikiNews, and the rest is from the additional references. The improvements over the baseline show the added value of our modified components, although still below reported state-of-the-art systems (Elmallah et al., 2024); however, a direct comparison is not possible due to differences in training datasets and target diacritization standards. The increase in accuracy in the WikiNewsMaxDiacs set suggests more work is needed in the space of proper Arabic diacritization evaluation.

7 Conclusions and Future Work

In this paper, we conducted a detailed analysis of **WildDiacs** patterns across six different genres of Arabic text. We developed **Wild2MaxDiacs**, a new annotated dataset designed for evaluating models to exploit **WildDiacs** for maximal diacritization. Additionally, we extended the **WikiNews** dataset to **WikiNewsMaxDiacs**, aligning it more closely with our definition of maximal diacritization. We also enhanced an open-source toolkit, **CAMeL Tools**, to effectively utilize partial diacritization, resulting in improved performance.

In the future, we plan to enhance the morphological analyzer's coverage, expand our research to include less explored genres like Classical Arabic texts, and investigate other *wild* text signals such as Hamza usage and dialectal spelling variations. All of the modifications and extensions will be integrated in a future version of CAMeL Tools. We also plan to study the effect of our system on downstream applications such as text-to-speech (Halabi, 2016; Abdelali et al., 2022) and Arabic romanization (Eryani and Habash, 2021).

Limitations

- We acknowledge that our downstream application evaluation focused solely on automatic diacritization, an important task in Arabic NLP with potential benefits for other applications.
- We acknowledge that the observations are limited by the selection of genres and sample sizes we studied.
- We acknowledge that the definition of *maximal diacritization*, as with *full diacritization*, is an open question and there may be different views on what is essential or not that we did not include or included, respectively.

Ethics Statement

- We do not violate any preconditions on the pre-existing resources we use to our knowledge.
- The texts we release are either already out of copyright, creative commons, or allowable within fair use laws (short snippets).
- All new manual annotations were done by the paper authors and were compensated fairly.
- Like all NLP models, there is a chance that mistakes could be made by the system leading to unintended consequences. However, in this particular setup and problem, we are not aware of any sources of systematic bias or harm.

Acknowledgements

We thank the anonymous reviewers for their insightful comments, and Kareem Darwish and Hamdy Mubarak for helpful conversations.

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.

Ahmed Abdelali, Mohammed Attia, Younes Samih, Kareem Darwish, and Hamdy Mubarak. 2019. Diacritization of Maghrebi Arabic sub-dialects.

Ahmed Abdelali, Nadir Durrani, Cenk Demiroglu, Fahim Dalvi, Hamdy Mubarak, and Kareem Darwish. 2022. NatiQ: An end-to-end text-to-speech system for Arabic. In *Proceedings of the The Seventh Arabic Natural Language Processing Workshop (WANLP)*, pages 394–398, Abu Dhabi, United Arab Emirates (Hybrid).

Muhamed Al Khalil, Hind Saddiki, Nizar Habash, and Latifa Alfalasi. 2018. A Leveled Reading Corpus of Modern Standard Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Miyazaki, Japan.

Nouf Alabbasi, Mohamed Al-Badrashiny, Maryam Aldahmani, Ahmed AlDhanhani, Abdullah Saleh Alhashmi, Fawaghy Ahmed Alhashmi, Khalid Al Hashemi, Rama Emad Alkhobbi, Shamma T Al Maazmi, Mohammed Ali Alyafeai, Mariam M Alzaabi, Mohamed Saqer Alzaabi, Fatma Khalid Badri, Kareem Darwish, Ehab Mansour Diab, Muhammad Morsy Elmallah, Amira Ayman Elnashar, Ashraf Hatim Elneima, MHD Tameem Kabbani, Nour Rabih, Ahmad Saad, and Ammar Mamoun Sousou. 2022. Gulf Arabic diacritization: Guidelines, initial dataset, and results. In *Proceedings of the The Seventh Arabic Natural Language Processing Workshop (WANLP)*, pages 356–360, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Hanan Aldarmaki and Ahmad Ghannam. 2023. Diacritic Recognition Performance in Arabic ASR. In *Proc. INTERSPEECH 2023*, pages 361–365.

Bashar Alhafni, Go Inoue, Christian Khairallah, and Nizar Habash. 2023. Advancements in Arabic grammatical error detection and correction: An empirical investigation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6430–6448, Singapore.

Badr AlKhamissi, Muhammad ElNokrashy, and Mohamed Gabr. 2020. Deep diacritization: Efficient hierarchical recurrence for improved Arabic diacritization. In *Proceedings of the Fifth Arabic Natural Language Processing Workshop*, pages 38–48, Barcelona, Spain (Online).

Sawsan Alqahtani, Hanan Aldarmaki, and Mona Diab. 2019. Homograph disambiguation through selective diacritic restoration. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 49–59, Florence, Italy.

Sawsan Alqahtani, Mahmoud Ghoneim, and Mona Diab. 2016. Investigating the impact of various partial diacritization schemes on Arabic-English statistical machine translation. In *Conferences of the Association for Machine Translation in the Americas: MT Researchers' Track*, pages 191–204, Austin, TX, USA. The Association for Machine Translation in the Americas.

Sawsan Alqahtani, Ajay Mishra, and Mona Diab. 2020. A multitask learning approach for diacritic restoration. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8238–8247, Online.

Parnia Bahar, Mattia Di Gangi, Nick Rossenbach, and Mohammad Zeineldeen. 2023. Take the Hint: Improving Arabic Diacritization with Partially-Diacritized Text. In *Proc. INTERSPEECH 2023*, pages 3949–3953.

Yonatan Belinkov and James Glass. 2015. Arabic diacritization with recurrent neural networks. In *Pro-*

ceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2281–2285, Lisbon, Portugal.

Kareem Darwish, Ahmed Abdelali, Hamdy Mubarak, and Mohamed Eldesouki. 2021. Arabic diacritic recovery using a feature-rich bilstm model. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 20(2).

Kareem Darwish, Hamdy Mubarak, and Ahmed Abdelali. 2017. Arabic diacritization: Stats, rules, and hacks. In *Proceedings of the Third Arabic Natural Language Processing Workshop*, pages 9–17.

Mona Diab, Mahmoud Ghoneim, and Nizar Habash. 2007. Arabic diacritization in the context of statistical machine translation. In *Proceedings of Machine Translation Summit XI: Papers*, Copenhagen, Denmark.

Mona Diab, Nizar Habash, Reem Faraj, and May Ahmar. 2010. Guidelines for the creation of resources for Arabic dialects. In *Proceedings of the Workshop on Language Resources and Human Language Technology for Semitic Languages*.

Mahmoud El-Haj. 2020. Habibi - a multi dialect multi national Arabic song lyrics corpus. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 1318–1326, Marseille, France. European Language Resources Association.

AbdelRahim Elmadany, El Moatez Billah Nagoudi, and Muhammad Abdul-Mageed. 2023. Octopus: A multitask model and toolkit for Arabic natural language generation. In *Proceedings of ArabicNLP 2023*, pages 232–243, Singapore (Hybrid).

Muhammad Morsy Elmallah, Mahmoud Reda, Kareem Darwish, Abdelrahman El-Sheikh, Ashraf Hatim Elneima, Murtadha Aljubran, Nouf Alsaeed, Reem Mohammed, and Mohamed Al-Badrashiny. 2024. Arabic diacritization using morphologically informed character-level model. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 1446–1454, Torino, Italia. ELRA and ICCL.

Muhammad ElNokrashy and Badr AlKhamissi. 2024. Partial diacritization: A context-contrastive inference approach.

Fadhl Eryani and Nizar Habash. 2021. Automatic Romanization of Arabic bibliographic records. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, pages 213–218, Kyiv, Ukraine (Virtual).

Saeed Esmail, Kfir Bar, and Nachum Dershowitz. 2022. How much does lookahead matter for disambiguation? partial Arabic diacritization case study. *Computational Linguistics*, 48(4):1103–1123.

Ali Fadel, Ibraheem Tuffaha, Bara' Al-Jawarneh, and Mahmoud Al-Ayyoub. 2019a. Arabic text diacritization using deep neural networks.

Ali Fadel, Ibraheem Tuffaha, Bara' Al-Jawarneh, and Mahmoud Al-Ayyoub. 2019b. Neural Arabic text diacritization: State of the art results and a novel approach for machine translation. In *Proceedings of the 6th Workshop on Asian Translation*, pages 215–225, Hong Kong, China.

David Graff, Mohamed Maamouri, Basma Bouziri, Sondos Krouna, Seth Kulick, and Tim Buckwalter. 2009. Standard Arabic Morphological Analyzer (SAMA) Version 3.1. Linguistic Data Consortium LDC2009E73.

Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*, pages 573–580, Ann Arbor, Michigan.

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*, pages 53–56, Rochester, New York.

Nizar Habash, Anas Shahrour, and Muhamed Al-Khalil. 2016. Exploiting Arabic diacritization for high quality automatic annotation. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4298–4304, Portorož, Slovenia. European Language Resources Association (ELRA).

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*, pages 15–22. Springer, Netherlands.

Nizar Y Habash. 2010. *Introduction to Arabic natural language processing*, volume 3. Morgan & Claypool Publishers.

Nawar Halabi. 2016. *Modern standard Arabic phonetics for speech synthesis*. Ph.D. thesis, University of Southampton.

Osama Hamed and Torsten Zesch. 2018. The role of diacritics in increasing the difficulty of Arabic lexical recognition tests. In *Proceedings of the 7th workshop on NLP for Computer Assisted Language Learning*, pages 23–31, Stockholm, Sweden. LiU Electronic Press.

Go Inoue, Salam Khalifa, and Nizar Habash. 2022. Morphosyntactic tagging with pre-trained language models for Arabic and its dialects. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1708–1719, Dublin, Ireland.

Mustafa Jarrar, Nizar Habash, Faeq Alrimawi, Diyam Akra, and Nasser Zalmout. 2016. Curras: an annotated corpus for the Palestinian Arabic dialect. *Language Resources and Evaluation*, pages 1–31.

Mustafa Jarrar, Fadi Zaraket, Rami Asia, and Hamzeh Amayreh. 2018. Diacritic-based matching of Arabic words. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 18(2).

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. In *Proceedings of the International Conference on*

Arabic Language Resources and Tools, pages 102–109, Cairo, Egypt.

Mokthar Ali Hasan Madhfar and Ali Mustafa Qamar. 2021. Effective deep learning models for automatic diacritization of Arabic text. *IEEE Access*, 9:273–288.

Hamdy Mubarak, Ahmed Abdelali, Hassan Sajjad, Younes Samih, and Kareem Darwish. 2019. Highly effective Arabic diacritization using sequence to sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2390–2395, Minneapolis, Minnesota.

Ossama Obeid, Go Inoue, and Nizar Habash. 2022. Camelira: An Arabic multi-dialect morphological disambiguator. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 319–326, Abu Dhabi, UAE.

Ossama Obeid, Nasser Zalmout, Salam Khalifa, Dima Taji, Mai Oudah, Bashar Alhafni, Go Inoue, Fadhl Eryani, Alexander Erdmann, and Nizar Habash. 2020. CAMeL tools: An open source python toolkit for Arabic natural language processing. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 7022–7032, Marseille, France.

Robert Parker, David Graff, Ke Chen, Junbo Kong, and Kazuaki Maeda. 2011. Arabic Gigaword Fifth Edition. LDC catalog number No. LDC2011T11, ISBN 1-58563-595-2.

Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, pages 1094–1101, Reykjavik, Iceland.

Roozbeh Pournader, Bob Hallissy, and Lorna Evans. 2021. Unicode Arabic Mark Rendering.

Han Qin, Guimin Chen, Yuanhe Tian, and Yan Song. 2021. Improving Arabic diacritization with regularized decoding and adversarial training. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 534–542, Online.

Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab, and Cynthia Rudin. 2008. Arabic morphological tagging, diacritization, and lemmatization using lexeme models and feature ranking. In *Proceedings of ACL-08: HLT, Short Papers*, pages 117–120, Columbus, Ohio.

Anas Shahrour, Salam Khalifa, and Nizar Habash. 2015. Improving Arabic diacritization through syntactic analysis. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1309–1315, Lisbon, Portugal.

Dima Taji, Salam Khalifa, Ossama Obeid, Fadhl Eryani, and Nizar Habash. 2018. An Arabic morphological analyzer and generator with copious features. In *Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 140–150, Brussels, Belgium.

Brian Thompson and Ali Alshehri. 2022. Improving Arabic diacritization by learning to diacritize and translate. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 11–21, Dublin, Ireland (in-person and online). Association for Computational Linguistics.

Catalin Ungurean, Dragos Burileanu, Vladimir Popescu, Cristian Negrescu, and Aurelian Dervis. 2008. Automatic diacritic restoration for a TTS-based e-mail reader application. *UPB Scientific Bulletin, Series C*, 70(4):3–12.

Ken Whistler. 2023a. Unicode Character Database.

Ken Whistler. 2023b. Unicode Normalization Forms.

Waleed A. Yousef, Omar M. Ibrahime, Taha M. Madbouly, and Moustafa A. Mahmoud. 2019. Learning meters of Arabic and English poems with recurrent neural networks: a step forward for language understanding and synthesis.

Nasser Zalmout and Nizar Habash. 2017. Don't throw those morphological analyzers away just yet: Neural morphological disambiguation for Arabic. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 704–713, Copenhagen, Denmark.

Nasser Zalmout and Nizar Habash. 2019. Adversarial multitask learning for joint multi-feature and multi-dialect morphological modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1775–1786, Florence, Italy.

Nasser Zalmout and Nizar Habash. 2020. Joint diacritization, lemmatization, normalization, and fine-grained morphological tagging. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8297–8307, Online.

Taha Zerrouki and Amar Balla. 2017. Tashkeela: Novel corpus of Arabic vocalized texts, data for auto-diacritization systems. *Data in brief*, 11:147–151.

Michal Ziemski, Marcin Junczys-Dowmunt, and Bruno Pouliquen. 2016. The united nations parallel corpus v1.0. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Portorož, Slovenia.

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 44th Annual Meeting of the Association for Computational Linguistics*, pages 577–584, Sydney, Australia.

A Experimental Details

Computational Resources All the models were run on a MacBook Pro with a 2.3 GHz Quad-Core Intel Core i7 Processor and 32 GB RAM. All the experiments were done within ∼11 CPU hours.

B Well-formedness Check

Scope The well-formedness checking rules presented below focus on the validity of use of sequences of diacritics and specific letters. They do not guarantee that a word is correct linguistically, only that it has plausible maximal diacritization, e.g., the non-word كُكَّاكًا *kuk~aAkãA* would pass but the real word كُتَّاباً *kuta~AbAã* would fail.

Word Diacritization Patterns A word is formed of an optional starting pattern, required recurring middle letter pattern and an optional ending pattern.

Starting patterns are ordered combinations of:

1. Conjunction: وَ *wa* or فَ *fa*
2. Preposition: بِ *bi*, كَ *ka* or لِ *li*
3. Definite article: اَلْ *Aal.* with Moon letters, or اَل◌ّ *Aal*○∼ with Sun letters (○)
4. Alif Wasla (ا A) followed by a contextually dependent short vowel (see **Context** below)

The **recurring middle pattern**, repeated one or more times, is either of the following:

1. Any letter (except ا *A*) followed by (◌ّ? ((◌َ| ◌ِ| ◌ُ) | ◌َٰ) | ◌ْ) *∼?((a|i|u)|aá)|\.)*
2. A long vowel: ◌َا *aA*, ◌ُو *uw*, or ◌ِي *iy*.

Ending pattern is one of the following:

1. A valid Tanwiyn cluster: any letter (except ا *A*) followed by ◌ّ?(◌ً| ◌ٍ| ◌ٌ) *∼?(ã|ĩ|ũ)*. A bare Alif follows ◌ً *ã* for all letters except the following final sequences: أ *Â*, اء *A'*, ة *ħ*.
2. Waw-of-Plurality: ◌ُوا *uwA* or ◌َوْا *w.A*

Double Consonants Two consecutive letters with a Sukun each or a letter with Sukun followed by a letter with a Shadda are not allowed, unless at the end of the context.

Exceptions There is a small number of allowable exceptions, such as the name 'Amr' ending with silent و *w*: عَمْرَو *ςam.raw* and عَمْرٍو *ςam.rĩw*.

Context The rules apply per word, but have access to surrounding context words to validate contextual diacritics. Punctuation marks and sentence boundaries constitute context delimiters. We assume that any unhamzated Alif ا *A* at the beginning of an input word is an Alif Wasla. There are two context diacritization well-formedness rules: (i) All words must end with a vowel representation (short or long) if followed by a word starting with an Alif Wasla; and (ii) Context-initial Alif Wasla, e.g., after punctuation or at beginning of sentence, must be followed by a short vowel diacritic.

C Ranking and Re-Ranking

Ranking in Analysis and Disambiguation The ranking process consists of sorting the analyses using a ranking tuple for each analysis as a sorting key. For an analysis a, the ranking tuple $\boldsymbol{r_a}$ is defined as $\boldsymbol{r_a} = (M_a, P_a, L_a, W_a)$ used to sort the analyses where M_a is the number of morphological features matching the tagger's prediction, P_a is the joint part-of-speech and lemma log probability of the analysis, L_a is the lemma log probability of the analysis and W_a is diacritized orthography of the analysis. While sorting analyses, their respective ranking tuples are compared item-wise only comparing subsequent items when the previous items are equal. W_a is used to sort analyses lexicographically when all the prior items are equal to ensure a stable ranking.

Re-ranking Modifications We extend this tuple with four new features. We first compute the Levenshtein distance between the original input word and W_a, both normalized to a UNF (see Footnote 4) with consistent Tanwiyn Alif order. We count the number of insert operations (I_a), substitution operations (S_a) and deletion operations (D_a). We then create a new ranking tuple $\boldsymbol{r'_a} = (S_a + D_a, M_a, S_a, D_a, P_a, L_a, I_a, W_a)$ which we use to re-rank the analyses. This new ranking tuple has the following properties:

- When an input word contains any diacritic, the analyses that change these diacritics the least are ranked higher.
- When an input word has no diacritics, $\boldsymbol{r_a}$ and $\boldsymbol{r'_a}$ produce very similar rankings.
- When an input word is fully diacritized, any exact matching analysis will have the top rank.
- Deletions are preferred over substitutions since some diacritics can be removed while substitutions change the word meaning.

D Comparison of Different Diacritization Schemes

Phenomenon	Diacritization Scheme: ATB	Diacritization Scheme: WikiNews	Diacritization Scheme: Maximal	Examples	
Alif Wasla (aka Hamzat Wasl)	A | Ai | Au (lexical)	A	A | Ai | Au | Aa (lexical, contextual)	Aib.nihi | waAb.nihi Aux.ruj. | waAx.ruj. Aal.bay.tu | waAl.bay.tu	اِبْنِهِ | وَابْنِهِ اُخْرُجْ | وَاخْرُجْ اَلْبَيْتُ | وَالْبَيْتُ
Word-final Sukun	. | ∅ (lexical)	.	. | i | u | a (lexical, contextual)	katabat. | katabati min. | mina | mini (%n flag) hum. | humu (%m flag)	كَتَبَتْ | كَتَبَتِ مِنْ | مِنَ | مِنِ هُمْ | هُمُ
Long Vowel /ā/	A	aA	aA	kitaAbũ	كِتَابٌ
Tanwiyn Alif Order	Aã	ãA	ãA	kitaAbãA	كِتَابًا
Tanwiyn Alif Maqsura Order	ýã	ãý	ãý	fatãý	فَتًى
Dagger Alif	á	a	aá	haáðaA	هَٰذَا

Table 5: A comparison of different full diacritization schemes used in the literature.

E Morphological Analyzer Modifications

	#	Description	From		To	
(a)	1	Change bare Alif to Alif Wasla Fatha in the determiner Al	Al#+	ال#+	Äal#+	ٱَل#+
	2	Change bare Alif to Alif Wasla Fatha in relative pronouns	Al~aðiy	الَّذِي	Äal~aðiy	ٱَلَّذِي
	3	Add Sukun on Alif Fariqa (to mark it internally)	+aw.A	+◌َوْا	+aw.A.	+◌َوْاْ
			+uwA	+◌ُوا	+uwA.	+◌ُواْ
	4	Add missing Fatha before Alif	ςimAd	عِماد	ςimaAd	عِمَاد
	5	Add missing Fatha before Dagger Alif	háðaA	هٰذَا	haáðaA	هَٰذَا
	6	Add Missing Sukuns	+at	+◌َت	+at.	+◌َتْ
			briyTAniyA	بِرِيطانِيا	b.riyTaAniyaA	بْرِيطَانِيَا
	7	Flag the word مِن *min*	min	مِن	min%n	مِن%n
	8	Flag 2MP and 3MP suffixes ending with م *m*	lahum	لَهُم	lahum%m	لَهُم%m
(b)	1	Change Sukun before Alif Wasla to Kasra	biÄl.Äs.mi	بِٱلْٱسْمِ	biÄliÄs.mi	بِٱلِٱسْمِ
	2	Dediacritize Alif Waslas in the middle of the word	biÄis.mi	بِٱِسْمِ	biÄs.mi	بِٱسْمِ
	3	Adjust Tanwiyn Fath on Alif	jid~aAã	جِدَّاً	jid~ãA	جِدًّا
	4	Adjust Tanwiyn Fath on Alif Maqsura	fataýã	فَتَىً	fatãý	فَتًى
	5	Remove Sukun on Alif Fariqa	ramaw.A.	رَمَوْاْ	ramaw.A	رَمَوْا

Table 6: The most important morphological analyzer modifications with examples: (a) lists database modifications on various prefixes, stems and suffixes; and (b) lists online edits to analyses after prefix-stem-suffix concatenation.

F Contextual Edits

#	Description	From		To		XID
1	Change Sukun before Alif Wasla to Kasra	man. Äib.nuka	مَنْ ٱِبْنُكَ	mani Äib.nuka	مَنِ ٱِبْنُكَ	E1.a
2	Change %m flag to Damma before Alif Wasla, and to Sukun elsewhere	hum%m Äal.Hub~u	هُم%m ٱَلْحُبُّ	humu Äal.Hub~u	هُمُ ٱَلْحُبُّ	E2.a
		lahum%m salaAmũ	لَهُم%m سَلَامٌ	lahum. salaAmũ	لَهُمْ سَلَامٌ	**E3**
3	Change %n flag before Alif Wasla to mimic the Alif Wasla's diacritic, and change to Sukun elsewhere	min%n Äal.Hub~i	مِن%n ٱَلْحُبِّ	mina Äal.Hub~i	مِنَ ٱَلْحُبِّ	E4.a
		min%n Äib.nihi	مِن%n ٱِبْنِهِ	mini Äib.nihi	مِنِ ٱِبْنِهِ	E5.a
		min%n miS.ra	مِن%n مِصْرَ	min. miS.ra	مِنْ مِصْرَ	**E6**
4	Dediacritize Alif Waslas mid-context	mani Äib.nuka	مَنِ ٱِبْنُكَ	mani Äb.nuka	مَنِ ٱبْنُكَ	E1.b
		humu Äal.Hub~u	هُمُ ٱَلْحُبُّ	humu Äl.Hub~u	هُمُ ٱلْحُبُّ	E2.b
		mina Äal.Hub~i	مِنَ ٱَلْحُبِّ	mina Äl.Hub~i	مِنَ ٱلْحُبِّ	E4.b
		mini Äib.nihi	مِنِ ٱِبْنِهِ	mini Äb.nihi	مِنِ ٱبْنِهِ	E5.b
		qaAla Äib.nuka	قَالَ ٱِبْنُكَ	qaAla Äb.nuka	قَالَ ٱبْنُكَ	E7.a
5	Normalize all Alif Waslas to Alifs	mani Äb.nuka	مَنِ ٱبْنُكَ	mani Ab.nuka	مَنِ ابْنُكَ	**E1.c**
		humu Äl.Hub~u	هُمُ ٱلْحُبُّ	humu Al.Hub~u	هُمُ الْحُبُّ	**E2.c**
		mina Äl.Hub~i	مِنَ ٱلْحُبِّ	mina Al.Hub~i	مِنَ الْحُبِّ	**E4.c**
		mini Äb.nihi	مِنِ ٱبْنِهِ	mini Ab.nihi	مِنِ ابْنِهِ	**E5.c**
		qaAla Äb.nuka	قَالَ ٱبْنُكَ	qaAla Ab.nuka	قَالَ ابْنُكَ	**E7.b**
		Äis.mu	ٱِسْمُ	Ais.mu	اِسْمُ	**E8**

Table 7: Our system's contextual edits with examples. XID serves as the running example identifier, demonstrating the changes as rules are successively applied to the example. The green bolded cells mark the final forms.