Abstract

Diacritization for the World's Scripts

BSF START-UP Application 2022215

Kyle Gorman
Graduate Center, City University of New York

Yuval Pinter & Michael Elhadad
Ben-Gurion University of the Negev

In many writing systems, full knowledge about word identity and pronunciation is encoded by diacritic marks attached or adjacent to the core characters. Often, these marks are dropped either deliberately by the writer or during automatic text processing, leaving technologies like translation services and text-to-speech engines without information necessary for their operation. We propose to systematically quantify the role of diacritics in various types of scripts and languages, and to evaluate a wide array of systems proposed to this day whose goal is to restore diacritics to text lacking them. We will examine methods making use of linguistic knowledge, as well as those relying mostly on the power of vast computational power operating over large corpora, as well as a novel method inspired by recent advances in text-free processing of language through its rendered visual form. We will cover a range of language families and assess the importance of preserving or restoring diacritics across languages in multilingual applications.

1 Scientific Background

Many writing systems, both ancient and modern, systematically omit key information about the pronunciation of words. Such writing systems are sometimes referred to as *defective.* In Arabic, for example, consonants and long vowels are written, but short vowels are omitted except in some pedagogical and religious materials. Thus the word اليمن [ʔal.ja.man] ‘Yemen’ is written with five consonant characters ⟨'-l-y-m-n⟩ but no vowels are indicated. Similarly, in Latin texts—except for some pedagogical resources—there is no indication of vowel length, and ⟨I, V⟩ can either be read as short vowels [i, u], long vowels [iː, uː], or glides [w, j] depending on context. Therefore, the reader has little way to know that ⟨VVA⟩ is read [uː.wa] ‘grape’ and not, e.g., as *[vu.aː]. Similar issues arise—to a greater or lesser degree—in the writing systems of many hundreds of languages, including languages spoken in Europe, the Middle East, and South(east) Asia, not to mention many other elsewhere, particularly when the appropriate keyboard or text encoding support is unavailable.

Defective writing systems are a known issue for internationalizing speech technologies, since automatic speech recognition and text-to-speech synthesis demand more detailed pronunciation information than a defective writing system provides (Gorman et al. 2020, Ashby et al. 2021). Furthermore, because defectivity may introduce linguistic ambiguity, they also pose a problem for text understanding systems, including for machine translation. Indeed, defective writing systems have been cited as one major reason that speech and language technologies for Arabic and Hebrew seemingly lag behind other, similarly-resourced languages (e.g., Tsarfaty et al. 2019).

Diacritization procedures insert diacritics and other characters which are only rarely or irregularly written. Diacritizers are key components in speech and language technologies. They must incorporate local syntacto-semantic information, and in some cases deal with extensive ambiguity; for example, there may be more than a dozen possible ways to diacritize a given Arabic word (Belinkov and Glass 2015). Diacritization, particularly of Arabic and Hebrew, has been studied for several decades, and recent work with deep learning suggest that substantial improvements are possible over traditional machine learning techniques such as decision trees or linear classifiers (e.g., Arora et al. 2020, Shmidman et al. 2020) or dictionary-based methods (e.g., Cvetana et al. 2018). For modern deep learning practitioners, diacriticization is naturally expressed as a sequence processing task, either a sequence tagging task or as a sequence-to-sequence (generation) task.

Under the tagging formulation, the model can either predict word-level tags used to diacritize each word (e.g., Darwish et al. 2017, Shmidman et al. 2020, Zalmout and Habash 2020), or it can be framed as a character tagging task in which each tag gives the diacritics for a single character (e.g., Náplava et al. 2018, Gershuni and Pinter 2022). When text is lacking diacritics, but otherwise correct, the sequence-to-sequence formulation is “overkill” in the sense that the model must learn to regenerate the base characters. Sequence-to-sequence models can learn to perform this regeneration, but with a few exceptions, they lack the necessary inductive biases favoring this behavior, and this could introduce unnecessary errors or reduce data efficiency. Nevertheless, some previous work has opted for this technique (e.g., Belinkov and Glass 2015). With the exception of Gershuni and Pinter (2022), who compare LSTM character- and word-tagging models for Hebrew, there has been very little head-to-head comparison of these various problem formulations, and even less comparison of systems across different languages.

Language understanding systems are increasingly dominated by large-scale transformer based language models (LLMs), usually fine-tuned to specific tasks. In some cases, these models are **multilingual** in the sense that dozens of languages may share a single tokenizer, embedding layer, and encoder. Motivated by a belief that will allow for some degree of cross-lingual projection, it is common practice to remove all diacritics prior to tokenization, as was done, for example, by the widely-used mBERT model (according to Clark et al. 2022). At the same time, our pilot work suggests that neural machine translation systems produce radically different results depending on whether the input includes diacritics or not. For example, when translating Russian into Hebrew using the popular MarianNMT toolkit (Junczys-Dowmunt et al. 2018), adding or removing Russian accent marks—which are not commonly written—may result in incorrect translations at the word level. In sum, while mBERT itself may not be suitable for diacritization or related tasks, we hypothesize that related models which use diacritized text as input would result in substantial improvements in processing undiacritized text.

Research on diacritization has made use of rather different task formulations and computational models. Little is known about what tasks and models work in general, and next to nothing is known about what sorts of linguistic and orthographic properties provide affordances for particular modeling strategies, as most studies of diacritization focus on a single language. Despite its importance, diacritization has been left behind by current trends in natural language processing, such as the use of large pre-trained transformer language models.

1.1 Objectives and Impact

Our goal is to perform a **systematic study of diacritics and their restoration methods across languages**. The contributions of this project are threefold:

1. Development of a **unified theory of diacritics** currently absent from the literature, emanating from linguistically- and typologically-aware perspectives motivated by information theory. This theory will identify similarities and differences between languages and language-script pairs, allowing for data-driven model selection using these characteristics.

2. A principled application of multiple approaches to text diacritization, controlling for **data origin, size, and quality** and for **use of computational resources**. Considering some of these aspects for restoring diacritics is novel; one of the methods we propose (visual encoders) has not yet been applied to this task (or even its input-output configuration).

3. Actionable recommendations for application of different techniques for different levels of diacritization across different languages and scripts, conditioned on language typology developed in contribution (1) and on available computing and data resources. This will also contribute to the study of languages at large, through reaching insights on the role of diacritics in individual languages or across them, based on the applicability of different methods.

The design of our experiments and our choice of languages and models will support conclusions regarding various aspects of writing, language and linguistic variation. For example, they will measure the extent to which digital lexicons and other sources of **word-level information** can be used to restore diacritics. We hypothesize that this will vary as a factor of the linguistic function of the diacritics, the number of unique diacritics and their frequency, the extent to which there are “minimal pairs” with respect to diacritization (i.e., wordforms which are distinct only in undiacritized text), the richness of the language’s morphology, and the availability of appropriate lexical resources, among other factors. Complementary to this, the proposed experiments and languages will also help us to understand the **role of linguistic context** in restoring diacritics, and the ability of deep learning models to encode the contextual information needed to make these predictions. This duality of form vs. context is a longstanding area of interest in human language technologies (e.g., Pinter et al. 2020, Gorman et al. 2021). Our findings will also bear on questions about **cross-linguality and multilinguality** in human language technologies, by determining the degree to which various diacritic restoration models are transferable across languages. We ultimately seek a single approach which produces “reasonable” results across as many scripts and languages as possible, degrading gracefully when it fails, generalizing to related languages and scripts, and making efficient use of the available data. We will provide details for the level of different types of data necessary to perform well in each of the methods, including undiacritized text corpora, automatically- or manually-generated lexicons, texts diacritized semi-automatically or by human annotators. Such measures are helpful when considering languages and language varieties outside of the scope of the current project. We recognize that for many practitioners, diacritization is a problem rather than an end goal, and we hope to be able to point these practitioners in a good initial direction as they look for a solution, based on easy-to-measure properties of their corpora.

2 Task

In a diacritization task, one is provided with documents in a defective script. These texts may be totally undiacritized, or they may be incompletely diacritized. One then produces fully-diacritized version of these documents. Some examples are shown in Table 1. We first provide a cursory formal mechanism to quantify the phenomenon of diacritics from a pure orthographic perspective, the development and refinement of which will be carried out as part of our research plan, and follow up with details on the means by which we will evaluate the outputs of the computational diacritization models we will present in the next section.

2.1 A theory of diacritics

We present an initial formal theory of diacritics and diacritization using measures inspired by information theory.

Hebrew:
(a) איש היה בארץ עוץ איוב שמו
(b) אִישׁ הָיָה בְאֶרֶץ עוּץ אִיּוֹב שְׁמוֹ
‘In the land of Uz there lived a man whose name was Job.’
Latin:
(a) ...TROIAE QVI PRIMVS AB ORIS ITALIAM FATO PROFVGVS LAVINIAQVE VENIT LITORA...
(b) *...trojae quī prīmus ab ōris ītaliam fātō profugus lāvīniaque vēnit lītora...*
‘...first driven by fate from the coast of Troy to Italy and the Lavinian coast...’
Latvian:
(a) Vins tiecas klut par arabu pasaules austrumu dalas neoficialo lideri...
(b) *Viņš tiecās kļūt par arābu pasaules austrumu daļas neoficiālo līderi...*
‘He aspired to become the unofficial leader of the eastern part of the Arabic world...’
Vietnamese:
(a) Amphetamin duoc su dung de dieu tri roi loan tang dong giam chu y...
(b) *Amphetamin được sử dụng để điều trị rối loạn tăng động giảm chú ý...*
‘Amphetamine is used to treat attention deficit hyperactivity disorder...’

Table 1: The diacritization task, converting undiacritized text (a) to diacritized text (b). Diacritics for Hebrew and Latin were created during the Early Middle Ages and are mostly limited to pedagogical or religious texts. In Latvian and Vietnamese, however, these diacritics are generally used in modern texts, though undiacritized text may be produced when the language is typed without the benefit of keyboard support or appropriate encodings, or in informal communications between intimates.

2.1.1 Characters and diacritics

Let $\mathcal{C}$ be a vocabulary of “basic characters”, such as the set of alphabetical Latin characters used in words native to the English language (ASCII ranges 65–90 and 97–122), and let c denote a character in such a vocabulary. Let $\mathcal{D}$ be a set of diacritics **compatible** with $\mathcal{C}$ as used by language $\mathcal{L}$,[1] meaning that each $v \in \mathcal{D}$ may compose with at least one character in $\mathcal{C}$ to form a diacritized character, or a **rune**. For example, the acute accent marker forms the rune *á* when composed with *a*, as in Spanish. We refer to members of either the set of characters or diacritics as **glyphs** ($\mathcal{G} := \mathcal{C} \cup \mathcal{D}$).

Let $\delta : \mathcal{C} \rightarrow ((\mathcal{D})^*)^*$ be a mapping from basic characters to the set of possible diacritic assignments it can entertain within a language, each assignment being a tuple. For instance, the Vietnamese rune *ộ* is an instantiation of the character *o* being assigned the tuple $\langle$CIRCUMFLEX, DOT BELOW$\rangle$. Note that our definition of diacritics is designed to be maximally reductionist. Linguistically, the way in which the circumflexed rune is distinguished from the basic character is not systematic and mainly addresses a deficiency in vowel symbols of the Latin character inventory, whereas the dot below encodes (low rising) tone in a way affecting all vowel characters it adjoins in a consistent manner. We treat both symbols (and equivalent phenomena in other languages) as the same.[2]

Conversely, let $\sigma : (\mathcal{C} \times \mathcal{D}^*) \rightarrow \mathcal{C}$ be the function stripping a rune from its diacritics, returning the base character. Note that this formulation does not allow for “free diacritics”, which we do not expressly license. We now overload $\sigma(s)$ so that it can perform the action, rune-by-rune, on a diacritized string. For example, in French, $\sigma(\textit{déjà}) = \textit{deja}$.

[1]We only deal with language-specific properties in this work and thus the language remains implicit in the remainder.
[2]Despite the irrelevance of order among diacritics per character-wise assignment, we assume a canonical ordering—much like in Unicode—and denote the diacritic sets using angled brackets.

Definition 1 *A character c's **capacity** is the number of different diacritic sets it entertains:*

$$\mathrm{K}(c) := |\delta(c)|$$

Definition 2 *A diacritic mark v's **versatility** is the portion of basic characters which entertain it:*

$$\mathrm{P}(v) := \frac{|\{c \in \mathcal{C} : \exists d \in \delta(c).v \in d\}|}{|\mathcal{C}|}$$

2.1.2 Diacritization

Given a sequence of runes $D = [d_1 d_2 \ldots d_n]$ forming a text of $\mathcal{L}$, we define the **diacritization** operation φ from its stripped version $C := \sigma(D) =: [c_1 c_2 \ldots c_n]$ as our main focus of study. From a task-driven perspective, $\varphi(C) \mapsto D$ (where $d_i \in \delta(c_i) \forall 1 \leq i \leq n$) is the desired function for diacritization. Note that this mapping may not be unique: not only is it possible for individual words to be formed by selecting different runes for different characters, but such ambiguities may persevere beyond phrases, sentences, and even grounded context. In a slight abuse of notation, each d_i is seen as a set composed of c_i and the diacritics of the rune, a member of $\delta(c_i)$. The next definition follows from this note.

Definition 3 *Given a sequence $D = \varphi(C)$, we denote the* weight *of an individual character, or its corresponding rune, or its index, as the number of glyphs in the diacritized rune:*

$$|d_i| := |c_i| := |i| := 1 + |!\alpha \in \delta(c_i) \text{ s.t. } d_i = \{c_i\} \cup \alpha|$$

The next measure will be calculated over a large corpus in order to quantify corpus-, script- and language-level properties.

Definition 4 *Given a diacritized sequence D of length n, its* diacritic density Δ *is the average number of diacritic markers per rune:*

$$\Delta(D) := \frac{\sum_{i=1}^{n} |d_i|}{n} - 1$$

2.1.3 Information-theoretic statistics

Our intention is to obtain an information-theoretic idea of how diacritics interact with characters. One relaxing assumption that can be made is sort of a Markovian process for an hidden Markov model where "hidden states"—not in fact hidden—are the base characters and the "observations" are each character's corresponding diacritics. With C and D representing a sequence of the magnitude of a full corpus, we can estimate the emission probabilities of this model by decomposing the set of available diacritics into its components,

$$P_v(d|c) \approx \prod_{v \in d \setminus \{c\}} \frac{\#(d_i \in D \text{ s.t. } c_i = c \wedge v \in d_i)}{\#(c \in C)},$$

or we could compute individual probabilities for diacritic tuples,

$$P_t(d|c) \approx \frac{\#(d_i \in D \text{ s.t. } d_i = d)}{\#(c \in C)},$$

or even a type-based measure for tuples:

$$P_\delta(v|c) \approx \frac{|\{d \in \delta(c) : v \in d\}|}{\mathrm{K}(c)}.$$

In the first case, we might need an explicit estimate for the empty set as a protected case.

The transition probabilities will follow the usual bigram estimation $P(c_i|c_{i-1}) \approx \frac{\#_D(c_{i-1}c_i)}{\#_D(c_i)}$, with add-$\gamma$ smoothing assumed but not explicitly denoted for aesthetic purposes.

We obtain a measure along the lines of:

$$I_v(d_i) := - \sum_{v \in d_i \setminus \{\sigma(d_i)\}} \log P(\sigma(d_i)|\sigma(d_{i-1})) + \log P(v|\sigma(d_i))$$

$$I_\delta(d) := - \sum_{v \in d \setminus \{\sigma(d)\}} \log P(\sigma(d_i)|\sigma(d_{i-1})) + \log P_\delta(v|\sigma(d))$$

$$I_v(D) := \sum_{d \in D} I_v(d)$$

$$I_\delta(D) := \sum_{d \in D} I_\delta(d)$$

This model is certainly oversimplified: certain rune-pair combinations are definitely impossible, yet this model does not at all consider relationships between neighboring runes' diacritics. The model also does not allow forward-looking dependencies, which surely exist as well. Thus we use these measures solely as an information metric, rather than as models for diacritic restoration.

Having collected the statistics for various language corpora from Universal Dependencies treebanks,[3] we report our findings in Table 2, using the simple NFD-based decomposition method (subsection 3.8) as σ. Here, mI_α denotes the mean of the corresponding I_α measure. The data shows that languages using Latin script can show information patterns that do not trivially emanate from the frequency of diacritics in their corpora (shown under "Diacritic density"), as well as different sensitivities in character–rune relations, exemplified in the varying differences between mI_α and I_α measures. The data also demonstrates the inconsistent impact of domain change across languages, with a calibration datapoint pair for cross-lingual statistics.

Note finally that some symbols have more than one linguistic denotation in a given script. For instance, the Hebrew CENTRAL DOT might be treated as a single diacritic, but plays different grammatical roles as *dagesh* or *mappiq*, and this may affect the overall distributions across base characters, or necessitate an elaboration of our theory. As noted in subsection 3.8, the appropriate definitions of what are and are not diacritics may vary by language and script and may not match the Unicode notion in all cases.

[3] `https://universaldependencies.org`

Train	Eval (D) if different	$mI_\delta(D)$	$mI_t(D)$	Diacritic density (%)
Fra-FTB		2.5667	2.5325	3.71
Fra-GSD		2.6613	2.6261	3.76
Fra-FTB	Fra-GSD	2.7100	2.6752	
Fra-GSD	Fra-FTB	2.6113	2.5769	
Spa-Ancora		2.5563	2.5332	2.38
Spa-Ancora	Spa-GSD	2.6238	2.6007	
Deu-GSD		2.5890	2.5747	1.50
Nno-NDT		2.6233	2.6049	2.05
Nob-NDT		2.6068	2.5922	1.63
Nob-NDT	Nno-NDT	2.6963	2.6776	
Nno-NDT	Nob-NDT	2.6798	2.6654	
Lit-Alksnis		2.6862	2.6282	6.96
Lat-AEN		2.7479	2.6553	11.19
Nno-NDT	Fra-FTB	3.3732	3.3516	
Fra-FTB	Nno-NDT	3.5349	3.5011	

Table 2: Information statistics for various Latin-scripted corpora and their interactions. Diacritic density denotes the percentage of characters in the training corpus which have at least one diacritic mark ($c \neq d$). Smoothing value used for the calculations is $\gamma = 0.1$.

2.2 Intrinsic evaluation

The primary metrics we will use to evaluate diacritization models is word-level accuracy, with character-level accuracy used as a secondary metric.[4] We will also measure precision, recall and F-score for individual diacritic classes. In Hebrew, for instance, this might include classes like *dagesh* and the *sin* and *shin* dots. We will also collect statistics about model size, training, and inference latency. Finally we will conduct a pilot user study with Hebrew speakers investigating their sentiment towards automatically diacritized Hebrew text produced by our best models.

2.3 Extrinsic evaluation

Diacritization is a key early step in speech & language processing pipelines for languages written in defective scripts, and errors at this stage are likely to propagate downstream. Therefore extrinsic evaluation is the gold standard for evaluating diacritization. In each case, we will simply perform diacritization on the training data, use this to train an off-the-shelf downstream model, and measure the change in performance on a downstream task. Below we describe some downstream tasks we believe are likely to benefit from diacritization, focusing on Hebrew.

To measure the contribution of diacritization to morphosyntactic processing, we will use the UDPipe 2.0 prototype (Straka 2018, Straka et al. 2019, Straka and Straková 2020) and data from the CoNLL 2018 shared task on multilingual parsing (Zeman et al. 2018) and the SIGMORPHON 2019

[4]We note that other metrics have been suggested, such as the vocalization metric for Hebrew's superfluous pronunciation system (Gershuni and Pinter 2022), and that others can be used for intermediate evaluation, e.g. a binary indicator-level accuracy metric for whether or not a character requires diacritics, for sparsely-diacritized orthographic systems.

Method	lexicon	corpus	ortho	domain	xling	mling
Rule-based	✓		✓	✓		
Morphological	✓			✓		
RNNs			✓	✓	✓	✓
Transformers		✓	✓	✓	✓	✓
Visual encoders		✓	✓	✓	✓	✓
Ensembles	✓	✓	✓	✓	✓	✓

Table 3: Characteristics or various approaches for transferable diacritization. Columns, in order: lexicon-dependent, large-corpus-dependent, orthographically robust, domain-adaptable, applicable across languages, applicable as a single multilingual model.

shared task on morphological analysis (McCarthy et al. 2019). Together these shared tasks include sentence boundary detection, tokenization, POS tagging, morphological analysis, lemmatization, and labeled dependency parsing subtasks, each with task-specific baselines and metrics.

Hebrew marks gender on nouns, but many words are ambiguous with respect to gender in their ordinary undiacritized form; e.g., הָלַכְתְּ 'you.sg.fem. walked' vs. הָלַכְתָּ 'you.sg.masc. walked'. Stanovsky et al. (2019) find that both commercial and research English-to-Hebrew machine translation systems are poor at predicting the gender of Hebrew nouns, and are more accurate when source sentences are consistent with gender stereotypes. We conjecture that a MT systems trained using diacritized Hebrew text will be more sensitive to gender information in the source text. Using FairSeq (Ott et al. 2019), we will train English-to-Hebrew translation systems and test this hypothesis using Stanovsky et al.'s data and metrics.

Considering the fact that nearly all diacritization affects pronunciation, an important application of our research artifacts would be text-to-speech (TTS). At this time, we leave this avenue out of the official scope of the proposed project, to be pursued only following the allotted two years or if additional collaborators with appropriate experience join the team.

3 Methods

We intend to evaluate the utility of various classes of models for diacritization of defective text, matching both performance and error profiles to linguistic and textual variables as described in §4. We offer a taxonomy of our selected methods, based on whether each system is **lexicon-dependent**, **requires a large corpus**, **within-language orthographically robust**, **within-language domain-adaptable**, **applicable across languages**, **applicable as a single multilingual model**. Table 3 presents this taxonomy, with major model classes as detailed below.

In addition to the different models, we intend to look into additional design selections such as decoding strategies and the effects of corpus size and change of domain in training and inference. These experiments will present themselves as analysis of performance and errors dictate throughout the model evaluation processes. One such decision, namely the desired policy for handling the Unicode standard's differing treatments of different kinds of diacritics, is discussed in §3.8.

3.1 Naïve baselines

To calibrate expectation for system performance, we will implement a number of "dummy" baselines not requiring any form of complex modeling. These will include a baseline which never add diacritics (expected to score well for corpora that use diacritics sparsely), a baseline that selects the most common training-set diacritization for each word type, a baseline which selects the most common diacritic for each base character, and a baseline that stochastically emulates the per-base character diacritic distribution.

3.2 Decision lists and trees

As less naïve baselines, we will implement a system using word-level decisions lists (e.g., Yarowsky 1994, 1999). We will also implement a system using character-level decision trees conditioned on nearby base characters (e.g., Arora et al. 2020).

3.3 Sequence-to-sequence models

As mentioned earlier, some prior work (e.g., Belinkov and Glass 2015) has framed diacritization as a sequence-to-sequence generation problem. As a strong baseline, we will train attentive bidirectional LSTM and transformer character-based sequence-to-sequence models for diacritization.

3.4 Deep word tagging

Some prior work frames diacritization as a word tagging task (e.g., Shmidman et al. 2020). In this formulation, each tag is a pointer to instructions for deterministically diacriticizing that word, much like the traditional machine-learning approach to lemmatization (Chrupała et al. 2008). It is also similar in nature to morphological tagging and earlier dictionary-based methods, with machine learning used to resolve grammatical ambiguities and—possibly—generalize to out-of-vocabulary words. We hypothesize that this approach will be effective in languages with templatic morphology, since here the tags in some cases correspond to genuine natural classes posited by linguists. For example, nominal diminutives in Modern Standard Arabic are formed by mapping a CCC triliteral root onto a CuCayC template (McCarthy and Prince 1990), and much of this template is conveyed by diacritics. However, we hypothesize word tagging models will perform poorly on out-of-vocabulary words. We will experiment with word-based taggers, building the encoder from attentive bidirectional LSTM or transformer layers.

3.5 Deep character tagging

Other work has treated diacritization as a character tagging task, Each tag then gives the diacritic(s), if any, for the current character. In scripts which permit multiple diacritics on single characters, the tag set either treat these as a single tag (an "omnibus" setup) or multiple tagger layers—all sharing a single encoder—may be used for mutually independent classes of diacritics (a "multi-tag" setup).

This latter method is used by Gershuni and Pinter (2022) for Hebrew diacritization. We will experiment with both tagger strategies, using encoders with attentive bidirectional LSTM or transformer layers as the encoder, once again experimenting with greedy or conditional random fields decoding.

Whereas transformer models are increasingly dominant in NLP in general, they have proved less consistently effective for small-vocabulary tasks, including diacritic restoration. While Náplava et al. (2021) report that a transformer-based approach improves earlier results in Czech diacritization using an LSTM omnibus tagging model (Náplava et al. 2018), and Arabic diacritization with transformers has also recently shown promising results (Mubarak et al. 2019, AlKhamissi et al. 2020), Gershuni and Pinter (2022) obtain better results in Hebrew using a bidirectional LSTM multi-tagger. One limitation of transformer-based models is that they appear to perform best in transfer learning settings, which require a lengthy pre-training phase with a masked language modeling task. Since the resource costs of performing such a step are often prohibitive, transformer-based models often utilize pre-trained models available from popular repositories (e.g., Wolf et al. 2020). These are generally not extensible to character-level tasks because of their reliance on subword tokens which typically encompass multiple characters. Náplava et al. (2021) tailor their model to classify combinations of diacritics within multi-character sequences, a method that results in an enormous hypothesis space for decoding. Recent advancements in “tokenization-free” character- and byte-level transformers (e.g., Clark et al. 2022, Xue et al. 2022) and their application in this task (e.g. Stankevičius et al. 2022) suggest that we are near to resolving this limitation. TavBERT (Keren et al. 2022), a recently released character-level pre-trained transformer model for Hebrew, will be used to test this hypothesis in a much more challenging case than the European languages studied so far.

3.6 Deep visual networks

Rendering character sequences graphically and then embedding the resulting images can provide robustness to textual noise in encoding text for downstream tasks such as machine translation (Salesky et al. 2021) and language modeling (PIXEL; Rust et al. 2022). This novel method allows inputs containing typographical errors and deliberate or playful character obfuscation (such as replacing the letter *I* with the number *1*) to be represented similarly to the standard word forms, something unattainable under the dominant framework in which input text is presented in a discrete character mapping such as Unicode. We wish to extend the insights garnered from visual *encoding* to the *prediction* domain, visualized diagrammatically in Figure 1: in addition to representing the undiacritized input via an encoder similar to that in aforementioned prior art, we will also render candidate diacritization outcomes visually and embed them using a local PIXEL-based encoder, then use a softmax classifier to choose between diacritic combinations at the character or window level and backpropagate the cross-entropy loss into both encoders, generalizing training signals to cases where none of the contributing marks or their combination have been previously seen. In existing diacritization models’ approaches to the omnibus setting, where a single character can license several diacritic marks concurrently, encoding all possible mark combinations symbolically as separate classes inevitably loses much of the information brought from the individual marks being composed (e.g. in Vietnamese, where tone and vowel quality are independent but co-marked); while predicting each

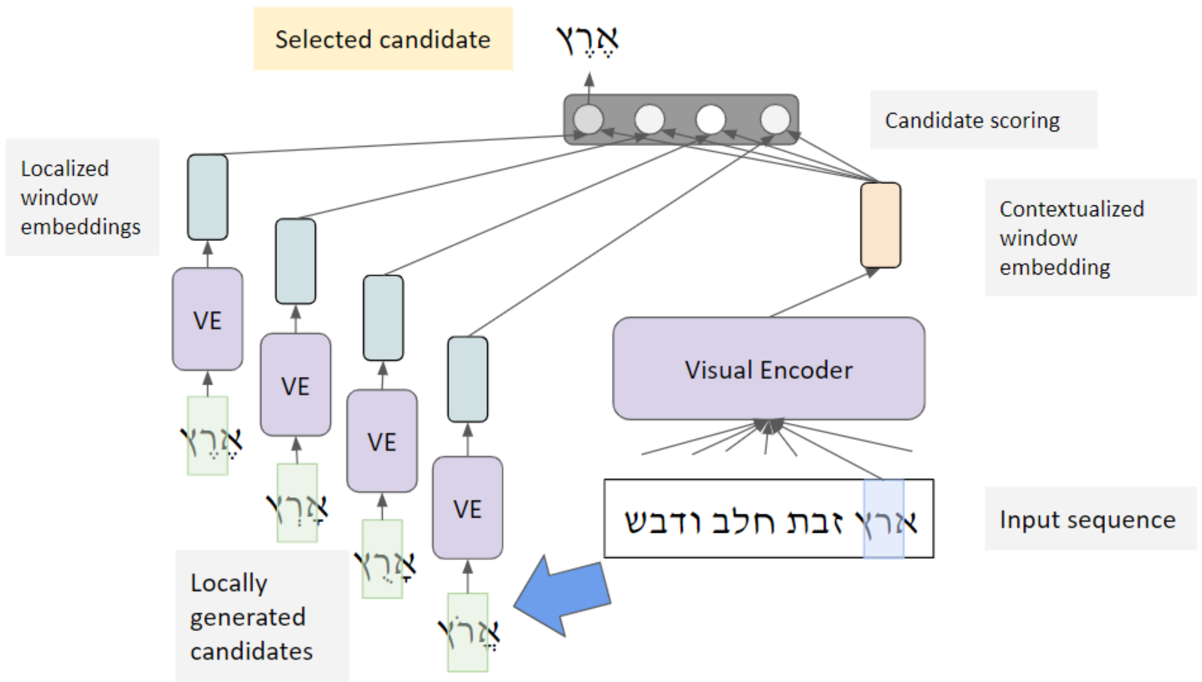

Figure 1: A schematic diagram of our initial design for a diacritization visual network. A visual encoder, built on models developed recently for language modeling and machine translation, is used for representing the input in a contextualized manner but also for local representation of diacritization candidates, a novel application for this technology. This architecture offers flexibility for unseen combinations of base characters and diacritics absent from other methods we survey.

component separately can lead to loss of compositional information obtained in the combination itself (e.g. in Hebrew, where *hataph* markings denote a templatic schwa that is vocalized in a specific manner, determined by its environment). Visual embeddings based on convolutional nets would surface both the individual markers and their combinations to the modeling layer, allowing all cases to be represented meaningfully.

This application would present a novel downstream setup for visual NLP models, which have so far been used either for fixed-set classification or for feeding a "classical" fixed-vocabulary decoder. The flexibility afforded by our setup can accommodate cases in which combinations of diacritics might be said to have compositional or relative meanings, like the double-acute accent in Hungarian; where visually-similar marks have similar usage (e.g. Hebrew's tsere and segol characters, דֵ / דֶ, both signifying the /ɛ/ vowel); and perhaps even capture cross-lingual, cross-script generalizations about the meaning of diacritics which could be used in zero- or few-shot transfer settings, paving the road for training diacritizers in languages with very little existing diacritic material.

3.7 Ensemble methods

Ensemble methods have the potential to yield greater performance than can be obtained from any of the constituent models, so long as the errors made by the constituent models alone are not closely correlated. In deep learning settings, one can often improve performance simply by ensembling several instances of the same model with different random initializations. Zalmout and Habash

(2017) argue for the utility of aggregating word- and character-level models, and demonstrate the value of morphological analysis for RNN-based methods. As we plan to implement to implement both kinds of networks, as well as a novel method making use of the visual modality in rendered text, we will examine the relevance of ensembling different classes of models for diacritization. This will include simple voting ensembles, as well as Zalmout and Habash's hybrid method, which uses morphological analyzers to constrain the hypothesis space for deep learning models.

3.8 The Unicode challenge

When developing models handling aspects of characters and diacritics, special care must be taken in order to correctly process the text in a manner conforming with encoding standards. The starting point for all text canonicalization efforts is the Unicode specification, which recognizes that there may be multiple ways—some of which are language-specific—to segment a text into a sequence of characters. For instance, in two of Unicode's four normal forms, *é* is analyzed as a single glyph, and in the other two as a glyph plus a diacritic. The majority of Unicode text is in the "Canonical Composition" normal form (NFC). In this form, Latin, Cyrillic, and Greek "accents" are largely composed with the characters they attach to, so *é* (like Cyrillic *ё*, and like polytonic Greek *ἐ*) are all single code points. To strip diacritics from these characters one can instead convert to the "Canonical Decomposition" normal form (NFD), in which diacritics are separate codepoints, then remove any codepoints that belong to the "Mark, nonspacing" (`Mn`) category.

In consonantal alphabets like (Perso-)Arabic and the "Jewish square script" used to write Modern Hebrew, as well as the Brahmic alphasyllabaries of south Asia, diacritics are rarely if ever composed with the consonants, regardless of normalization form. They are represented as separate codepoints of the `Mn` or `Mc` varieties. In scripts (like Arabic) which permit multiple points on a single consonant, there is a canonical order for the codepoints: e.g., *shaddah* comes before vowel points like *kasrah*. However, most rendering engines can correctly process text even if the diacritic codepoints are in a non-canonical ordering.

We aim to use insights regarding the representation of full text in the Unicode format when designing the input and output logic of our models. In some scripts, undiacritized text can be automatically generated by converting diacritized text to NFD, followed by removing any `Mn` and `Mc` codepoints. In others, however, additional information is needed to identify diacritics. Náplava et al. (2018), for instance, propose a more aggressive diacritic stripping strategy based on the automated analysis of Unicode character names, and additional strategies are required to produce undiacritized text in Bangla and Latin. These adaptations necessary for consistent diacritic modeling across scripts and languages will form part of our project's contributions, whether as research papers in appropriate venues and/or as an open-source package designed to expose the different possible decisions to text processing practitioners and offer recommendation for different use cases.

4 Languages

Our selection of languages for performing experiments on balances between several desiderata. First, our pan-linguistic goals dictate striving for diverse language and scripts across as many properties as possible: historical-genetic-oriented language families, relevant typological properties, especially those concerning morphological richness, script types (alphabets, alphasyllabaries, consonantal alphabets, etc.), the exponence and uniqueness of diacritical marks available in the script, and the extent to which those are used in practice based on domain, register, or other variables. The latter two properties can be quantified, for any given corpus, through the information-theoretic measures of diacritics-character relationships presented in §2.1. However, practical concerns are also a guiding factor given the limited duration of the proposed project: we are focused on languages in which team members are familiar with to expedite development, evaluation, and error analysis; availability of substantial quantities of high-quality diacritized text is also essential.

We therefore propose a list of *core languages*, for which we will guide our analysis and model development at the first stage. We will follow up with experiments on *extension languages*, which contribute less towards the diversity of the set and/or for which our team lacks deep linguistic expertise. For these extension languages, we rely on either prior studies or straightforward data collection strategies and judge quality based on the data alone. Each language added to our set also provides us with quadratic opportunities for measuring the language transfer capabilities of the various models.

4.1 Core languages

Arabic Modern Standard Arabic is a standardized, literary register of Arabic, a Semitic language with templatic morphology. It is written in a consonantal alphabet (or abjad) which omits short vowels, the presence of gemination, and word-final case markers. Following much prior work, we will use several resources, the Prague Arabic Dependency treebank (Smrž et al. 2008), the transcripts of the Arabic Broadcast News corpus (LDC2006T20), the ARL Arabic Dependency Treebank, and a filtered subset of the Tashkeela corpus (Abbad and Xiong 2021).

Hebrew Modern Hebrew is a Semitic-substrate language which has undergone revitalization in the late 19th century, leading to influence from many other languages. Still, insofar as its orthography is concerned, Hebrew exhibits templatic morphology and is written in an abjad with a very high prevalance of diacritics. Recent work on Hebrew diacritization (Shmidman et al. 2020, Rubinstein and Shmidman 2021, Gershuni and Pinter 2022) gives us access to generous amounts of diacritized Hebrew data spanning the modern era.

Latin Classical Latin is a richly inflected Italic language written in an alphabet derived from Etruscan and Greek. It is defective in that it fails to write the phonemic contrast between short and long monophthongs and between high vowels and glides. Winge (2015) investigates Latin diacritization using a statistical morphological tagger. Pedagogical editions of some Latin poetic texts provide the relevant diacritics needed to read the text with the appropriate meter. One of the PIs has digitized one such large text (Pharr 1964), and the proposed project will continue digitization efforts.

Bangla Bangla (also known as Bengali) is a moderately-inflected Indo-Aryan language. It is the most widely spoken language in Bangladesh and the second most widely spoken language in India. While it is the fifth most widely spoken native language worldwide, it is poorly resourced, with very few language processing resources available for it. It is primarily written with the Assamese alphasyllabary (or abugida). In this script the presence or absence of the two "inherent" vowels /ɔ, ɒ/ is not normally indicated after consonants, and there is no consistent indication of the contrast between /æ, e/. In pilot work, we find that the vast majority of Bangla grapheme-to-phoneme errors are caused by these these two ambiguities. The proposed project will generate a corpus of diacritized Bangla text by leveraging existing pronunciation dictionaries (Gutkin et al. 2016, Lee et al. 2020).

4.2 Extension languages

The diacritization data collected by Náplava et al. (2021) and extended by Stankevičius et al. (2022) should allow fairly painless extension of the proposed models to a wide array of Latin-based languages including Vietnamese (Austroasiatic), Hungarian (Finno-Ugric), Turkish (Turkic), Irish (Celtic), Latvian and Lithuanian (Baltic), Czech*, Polish, and Slovak (Slavic), and French*, Spanish*, Romanian (Italic).[5]

5 Risk Analysis

On the resource side of the project, we do not anticipate problems with the size and quality of the data we collect for each language we choose, considering we have already explored the available resources. However, if one or more core languages were to present difficulties, either in curation or in interaction with one or more models, we intend to fill the gap by using the extension languages provided in §4.2. If needed, the list can be further extended in a brief but concerted survey effort.

On the modeling side, the majority of methods we intend to apply have been tested on some form at the character and word level for more than one language. The main outlier is the visual encoder architecture, a novel framework not yet implemented or tested in the way we propose. Should this method fail to exceed the other proposed techniques on all languages—a plausible risk—we will nevertheless have contributed a negative result and a data point for a burgeoning field. Even under such circumstances, our implementation efforts can be used to find other applicable tasks of a similar form, i.e. visual candidate generation and scoring, to be further developed in our future work.

6 Resources and Equipment

The US PI has a lab equipped with a small cluster of GPUs. These computers will be used for data collection efforts and to implement some of the less resource-demanding models. The US PI is

[5]Asterisks mark languages in which at least one PI has at least medium-level proficiency.

requesting two laptops for the graduate research assistants. One will be for a MA student in year 1, who will lead the data collection and vetting efforts; the other for a PhD student working in years 1–2, who will implement and pilot the baselines and the two deep tagging models. Both students will be supervised by Dr. Gorman. The US PI has also requested funds for UNIX IT consulting to assist in making this cluster remotely accessible to support the data collection and archiving efforts across multiple sites.

The Israeli PIs have access to a departmental GPU cluster able to run all experiments described in the proposal, including pre-training moderately-sized character-level transformer models, aided by recently developed packages enabling training large language models on academic-scale resources (e.g., Izsak et al. 2021, Ciosici and Derczynski 2022). As other projects also rely on said cluster, we are requesting funding for credits on a cloud computing platform to use at times of high cluster stress. We plan to recruit an MSc. student, to be supervised under Dr. Pinter, who will work at half capacity on this project mostly on the theoretical underpinnings of diacritization, joining an MSc. student who has already begun preliminary exploration towards developing a PIXEL-based visual prediction model, an effort which will be facilitated through requisition of a high-end laptop computer.

References

1. Abbad, H. and Xiong, S. (2021), 'Simple extensible deep learning model for automatic Arabic diacritization', *ACM Transactions on Asian and Low-Resource Language Information Processing* **21**(2), 40.

2. AlKhamissi, B., ElNokrashy, M. and Gabr, M. (2020), Deep diacritization: efficient hierarchical recurrence for improved Arabic diacritization, *in* 'Proceedings of the Fifth Arabic Natural Language Processing Workshop', pp. 38–48.

3. Arora, A., Gessler, L. and Schneider, N. (2020), Supervised grapheme-to-phoneme conversion of orthographic schwas in Hindi and Punjabi, *in* 'Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics', pp. 7791–7795.

4. Ashby, L. F., Bartley, T. M., Clematide, S., Del Signore, L., Gibson, C., Gorman, K., Lee-Sikka, Y., Makarov, P., Malanoski, A., Miller, S., Ortiz, O., Raff, R., Sengupta, A., Seo, B., Spektor, Y. and Yan, W. (2021), Results of the second SIGMORPHON shared task on multilingual grapheme-to-phoneme conversion, *in* 'Proceedings of the 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology', pp. 115–125.

5. Belinkov, Y. and Glass, J. (2015), Arabic diacritization with recurrent neural networks, *in* 'Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing', pp. 2281–2285.

6. Chrupała, G., Dinu, G. and van Genabith, J. (2008), Learning morphology with Morfette, *in* 'Proceedings of the Sixth International Conference on Language Resources and Evaluation', pp. 2362–2367.

7. Ciosici, M. R. and Derczynski, L. (2022), 'Training a T5 using lab-sized resources', *arXiv preprint arXiv:2208.12097* .

8. Clark, J. H., Garrette, D., Turc, I. and Wieting, J. (2022), 'Canine: pre-training an efficient tokenization-free encoder for language representation', *Transactions of the Association for Computational Linguistics* **10**, 73–91.

9. Cvetana, K., Ranka, S. and Duško, V. (2018), Knowledge and rule-based diacritic restoration in Serbian, *in* 'Proceedings of the 3rd International Conference Computational Linguistics in Bulgaria', pp. 41–51.

10. Darwish, K., Mubarak, H. and Abdelali, A. (2017), Arabic diacritization: stats, rules, and hacks, *in* 'Proceedings of the Third Arabic Natural Language Processing Workshop', pp. 9–17.

11. Gershuni, E. and Pinter, Y. (2022), Restoring Hebrew diacritics without a dictionary, *in* 'Findings of the Association for Computational Linguistics: NAACL 2022', pp. 1010–1018.

12. Gorman, K., Ashby, L. F., Goyzueta, A., McCarthy, A. D., Wu, S. and You, D. (2020), The SIGMORPHON 2020 shared task on multilingual grapheme-to-phoneme conversion, *in* '17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology', pp. 40–50.

13. Gorman, K., Kirov, C., Roark, B. and Sproat, R. (2021), Structured abbreviation expansion in context, *in* 'Findings of the Association for Computational Linguistics: EMNLP 2021', pp. 995–1005.

14. Gutkin, A., Ha, L., Jansche, M., Kjartansson, O., Pipatsrisawat, K. and Sproat, R. (2016), Building statistical parametric multi-speaker synthesis for Bangladeshi Bangla, *in* '5th Workshop on Spoken Language Technology for Under-resourced Languages', pp. 194–200.

15. Izsak, P., Berchansky, M. and Levy, O. (2021), How to train BERT with an academic budget, *in* 'Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing', pp. 10644–10652.

16. Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K., Neckermann, T., Seide, F., Germann, U., Fikri Aji, A., Bogoychev, N., Martins, A. F. T. and Birch, A. (2018), Marian: Fast neural machine translation in C++, *in* 'Proceedings of ACL 2018, System Demonstrations', Association for Computational Linguistics, Melbourne, Australia, pp. 116–121. **URL:** *http://www.aclweb.org/anthology/P18-4020*

17. Keren, O., Avinari, T., Tsarfaty, R. and Levy, O. (2022), 'Breaking character: are subwords good enough for MRLs after all?', *arXiv preprint arXiv:2204.04748* .

18. Lee, J. L., Ashby, L. F., Garza, M. E., Lee-Sikka, Y., Miller, S., Wong, A., McCarthy, A. D. and Gorman, K. (2020), Massively multilingual pronunciation mining with wikipron, *in* 'Proceedings of the Twelfth Language Resources and Evaluation Conference', pp. 4223–4228.

19. McCarthy, A. D., Vylomova, E., Wu, S., Malaviya, C., Wolf-Sonkin, L., Nicolai, G., Kirov, C., Silfverberg, M., Mielke, S. J., Heinz, J., Cotterell, R. and Hulden, M. (2019), The SIGMORPHON 2019 shared task: morphological analysis in context and cross-lingual transfer for inflection, *in* 'Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology', pp. 229–244.

20. McCarthy, J. J. and Prince, A. S. (1990), 'Foot and word in prosodic morphology: the Arabic broken plural', *Natural Language & Linguistic Theory* **8**(2), 209–283.

21. Mubarak, H., Abdelali, A., Sajjad, H., Samih, Y. and Darwish, K. (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)', pp. 2390–2395.

22. Náplava, J., Straka, M. and Straková, J. (2021), 'Diacritics restoration using BERT with analysis on Czech language', *arXiv preprint arXiv:2105.11408* .

23. Náplava, J., Straka, M., Straňák, P. and Hajič, J. (2018), Diacritic restoration using neural networks, *in* 'Proceedings of the Eleventh International Conference on Language Resources and Evaluation', pp. 1566–1573.

24. Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D. and Auli, M. (2019), FAIRSEQ: a fast, extensible toolkit for sequence modeling, *in* 'Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)', pp. 48–53.

25. Pharr, C. (1964), *Vergil's Aeneid: Books I–IV*, revised edn, D.C. Heath & Co.

26. Pinter, Y., Jacobs, C. l. and Eisenstein, J. (2020), Will it unblend?, *in* 'Findings of the Association for Computational Linguistics: EMNLP 2020', pp. 1525–1535.

27. Rubinstein, A. and Shmidman, A. (2021), NLP in the DH pipeline: transfer-learning to a chronolect, *in* 'Proceedings of the Workshop on Natural Language Processing for Digital Humanities', pp. 106–110.

28. Rust, P., Lotz, J. F., Bugliarello, E., Salesky, E., de Lhoneux, M. and Elliott, D. (2022), 'Language modelling with pixels', *arXiv preprint arXiv:2207.06991* .

29. Salesky, E., Etter, D. and Post, M. (2021), Robust open-vocabulary translation from visual text representations, *in* 'Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing', pp. 7235–7252.

30. Shmidman, A., Shmidman, S., Koppel, M. and Goldberg, Y. (2020), Nakdan: professional Hebrew diacritizer, *in* 'Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations', pp. 197–203.

31. Smrž, O., Bielický, V., Kouřilová, I., Kráčmar, J., Hajič, J. and Zemánek, P. (2008), Prague Arabic Dependency Treebank: a word on the million words, *in* 'Proceedings of the Workshop on Arabic and Local Languages', pp. 16–23.

32. Stankevičius, L., Lukoševičius, M., Kapočiūtė-Dzikienė, J., Briedienė, M. and Krilavičius, T. (2022), 'Correcting diacritics and typos with ByT5 transformer model', *arXiv preprint arXiv:2201.13242* .

33. Stanovsky, G., Smith, N. A. and Zettlemoyer, L. (2019), Evaluating gender bias in machine translation, *in* 'Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics'.

34. Straka, M. (2018), UDPipe 2.0 prototype at conll 2018 UD shared task, *in* 'Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies', pp. 197–207.

35. Straka, M. and Straková, J. (2020), UDPipe at EvaLatin 2020: contextualized embeddings and treebank embeddings, *in* 'Proceedings of LT4HALA 2020: 1st Workshop on Language Technologies for Historical and Ancient Languages', pp. 124–129.

36. Straka, M., Straková, J. and Hajič, J. (2019), UDPipe at SIGMORPHON 2019: contextualized embeddings, regularization with morphological categories, corpora merging, *in* 'Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology', pp. 95–103.

37. Tsarfaty, R., Sadde, S., Klein, S. and Seker, A. (2019), What's wrong with Hebrew NLP? and how to make it right, *in* 'Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing: System Demonstrations', pp. 259–264.

38. Winge, J. (2015), Automated annotation of Latin vowel length, Bachelor's thesis, Uppsala University.

39. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q. and Rush, A. (2020), Transformers: state-of-the-art natural language processing, *in* 'Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations', pp. 38–45.

40. Xue, L., Barua, A., Constant, N., Al-Rfou, R., Narang, S., Kale, M., Roberts, A. and Raffel, C. (2022), 'Byt5: towards a token-free future with pre-trained byte-to-byte models', *Transactions of the Association for Computational Linguistics* **10**, 291–306.

41. Yarowsky, D. (1994), Decision lists for lexical ambiguity resolution: application to accent restoration in Spanish and French, *in* '32nd Annual Meeting of the Association for Computational Linguistics', pp. 88–95.

42. Yarowsky, D. (1999), A comparison of corpus-based techniques for restoring accents in Spanish and French text, *in* S. Armstrong, K. Church, P. Isabelle, S. Manzi, E. Tzoukermann and D. Yarowsky, eds, 'Natural Language Processing Using Very Large Corpora', Springer, pp. 99–120.

43. Zalmout, N. and Habash, N. (2017), Don't throw those morphological analyzers away just yet: Neural morphological disambiguation for Arabic, *in* 'Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing', pp. 704–713.

44. Zalmout, N. and Habash, N. (2020), Joint diacritization, lemmatization, normalization and fine-grained morphological tagging, *in* 'Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics', pp. 8297–8307.

45. Zeman, D., Hajič, J., Popel, M., Potthast, M., Straka, M., Ginter, F., Nivre, J. and Petrov, S. (2018), CoNLL 2018 shared task: multilingual parsing from raw text to Universal Dependencies, *in* 'Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies', pp. 1–21.