Restoring Hebrew Diacritics Without a Dictionary

Elazar Gershuni
Technion – Israel Institute of Technology
Haifa, Israel
elazarg@gmail.com

Yuval Pinter
Department of Computer Science
Ben-Gurion University of the Negev
Beer Sheva, Israel
uvp@cs.bgu.ac.il

Abstract

We demonstrate that it is feasible to accurately diacritize Hebrew script without any human-curated resources other than plain diacritized text.

We present NAKDIMON, a two-layer character-level LSTM, that performs on par with much more complicated curation-dependent systems, across a diverse array of modern Hebrew sources. The model is accompanied by a training set and a test set, collected from diverse sources.

1 Introduction

The vast majority of modern Hebrew texts are written in a letter-only version of the Hebrew script, one which omits the diacritics present in the full diacritized, or *dotted* variant.[1] Since most vowels are encoded via diacritics, the pronunciation of words in the text is left underspecified, and a considerable mass of tokens becomes ambiguous. This ambiguity forces readers and learners to infer the intended reading using syntactic and semantic context, as well as common sense (Bentin and Frost, 1987; Abu-Rabia, 2001). In NLP systems, recovering such signals is difficult, and indeed their performance on Hebrew tasks is adversely affected by the presence of undotted text (Shacham and Wintner, 2007; Goldberg and Elhadad, 2010; Tsarfaty et al., 2019).

As an example, the sentence in Table 1 (a) will be resolved by a typical reader as (b) in most reasonable contexts, knowing that the word "softly" may characterize landings. In contrast, an automatic system processing Hebrew text may not be as sensitive to this kind of grammatical knowledge and instead interpret the undotted token as the more frequent word in (c), harming downstream performance.

	המטוס נחת ברכות
(a)	hamatos naxat ????
	'The plane landed (unspecified)'
	הַמָּטוֹס נָחַת בְּרִכּוּת
(b)	hamatos naxat b-rakut
	'The plane landed softly'
	הַמָּטוֹס נָחַת בְּרָכוֹת
(c)	hamatos naxat braxot
	'The plane landed congratulations'

Table 1: An example of an undotted Hebrew text (a) (written right to left) which can be interpreted in at least two different ways (b,c), dotted and pronounced differently, but only (b) makes grammatical sense.

One possible way to overcome this problem is by adding diacritics to undotted text, or *dotting*, implemented using data-driven algorithms trained on dotted text. Obtaining such data is not trivial, even given correct pronunciation: the standard Tiberian diacritic system contains several sets of identically-vocalized forms, so while most Hebrew speakers easily read dotted text, they are unable to produce it. Moreover, the process of manually adding diacritics in either handwritten script or through digital input devices is mechanically cumbersome. Thus, the overwhelming majority of modern Hebrew text is undotted, and manually dotting it requires expertise. The resulting scarcity of available dotted text in modern Hebrew contrasts with Biblical and Rabbinical texts which, while dotted, manifest a very different language register. This state of affairs allows individuals and companies to offer dotting as paid services, either by experts or automatically, e.g. the Morfix engine by Melingo.[2] Such usage practices also force a disconnect in the NLP pipeline, requiring an API call into an external

[1] Also known as *pointed* text, or via the Hebrew term for the diacritic marks, *nikkud/niqqud*.

[2] https://nakdan.morfix.co.il/

Findings of the Association for Computational Linguistics: NAACL 2022, pages 1010 - 1018

service whose parameters cannot be updated.

Existing computational approaches to dotting are manifested as complex, multi-resourced systems which perform morphological analysis on the undotted text and look undotted words up in handcrafted dictionaries as part of the dotting process. Dicta's Nakdan (Shmidman et al., 2020), the current state-of-the-art, applies such methods in addition to applying multiple neural networks over different levels of the text, requiring manual annotation not only for dotting but also for morphology. Among the resources it uses are a diacritized corpus of 3M tokens and a POS-tagged corpus of 300K tokens. Training the model takes several weeks.[3]

In this work, we set out to simplify the dotting task as much as possible to standard modules. We introduce a large corpus of semi-automatically dotted Hebrew, collected from various sources, and use it to train an RNN-based model. Our system, NAKDIMON, accepts the undotted character sequence as its input, consults no external resources or lexical components, and produces diacritics for each character, resulting in dotted text whose quality is comparable to that of the commercial Morfix, on both character-level and word-level accuracy. Our model is easy to integrate within larger systems that perform end-to-end Hebrew processing tasks, as opposed to the existing proprietary dotters. To our knowledge, this is the first attempt at a "light" model for Hebrew dotting since early HMM-based systems (Kontorovich, 2001; Gal, 2002).

We introduce a novel test set for Modern Hebrew dotting, derived from larger and more diverse sources than existing datasets. In experiments over our dataset, we show that our system is particularly useful in the main use case of modern dotting, which is to convey the desired pronunciation to a reader, and that the errors it makes should be more easily detectable by non-professionals than Dicta's.[4]

2 Task and Datasets

2.1 Dotting as Sequence Labeling

The input to the dotting task consists of a sequence of characters. Each of the characters is assigned three values, from three separate diacritic categories: one category for the dot distinguishing *shin* (שׁ) from *sin* (שׂ), two consonants sharing a base character ש; another for the presence of *dagesh/mappiq*, a central dot affecting pronunciation of some consonants, e.g. פּ /p/ from פ /f/, but also present elsewhere; and one for all other diacritic marks, which mostly determine vocalization, e.g. דָ /da/ vs. דֶ /de/. Diacritics of different categories may co-occur on single letters, e.g. שָׁ, or may be absent altogether.

Full script Hebrew script written without intention of dotting typically employs a compensatory variant known colloquially as full script (*ktiv male*, כתיב מלא), which adds instances of the letters י and ו in some places where they can aid pronunciation, but are incompatible with the rules for dotted script. In our formulation of dotting as a sequence tagging problem, and in collecting our test set from raw text, these added letters may conflict with the dotting standard. For the sake of input integrity, and unlike some other systems, we opt not to remove these characters, but instead employ a dotting policy consistent with full script. See Appendix A for further details.

2.2 Training corpora

Dotted modern Hebrew text is scarce, since speakers usually read and write undotted text, with the occasional diacritic added for disambiguation when context does not suffice. As we are unaware of legally-obtainable dotted modern corpora, we use a combination of dotted pre-modern texts as well as automatically and semi-automatically dotted modern sources to train NAKDIMON:

The PRE-MODERN portion is obtained from two main sources: A combination of late pre-modern text from Project Ben-Yehuda, mostly texts from the late 19th century and the early 20th century;[5] rabbinical texts from the medieval period, the most important of which is Mishneh Torah (obtained from Project Mamre);[6] and 23 short stories from the short story project.[7] This portion contains roughly 1.81M Hebrew tokens, most of which are dotted, with a varying level of accuracy, varying dotting styles, and varying degree of similarity to Modern Hebrew.

The AUTOMATIC portion contains 547 short stories taken from the short story project. The stories are dotted using Dicta *without* manual validation.

[3] Private communication.

[4] The system is available at https://nakdimon.org, and the source code is a available at https://github.com/elazarg/nakdimon.

[5] https://benyehuda.org

[6] https://mechon-mamre.org

[7] https://shortstoryproject.com/he/

	Genre	Sources	# Docs	# Tokens
	Wiki	Dicta test set	22	5,862
	News	Yanshuf	78	11,323
†	Literary	Books, forums	129	73,770
*	Official	gov.il	24	20,181
*	News / Mag	Online outlets	137	92,151
*	User-gen.	Blogs, forums	63	60,673
*	Wiki	he.wikipedia	40	62,723
	Total		493	326,683

Table 2: Data sources for our MODERN Hebrew training set. Rows marked with * were automatically dotted via the Dicta API and corrected manually. Rows with † were dotted at low quality, requiring manual correction. The rest were available with professional dotting.

The corpus contains roughly 1.27M Hebrew tokens.

Lastly, the MODERN portion contains manually collected text in Modern Hebrew, mostly from undotted sources, which we dot using Dicta and follow up by manually fixing errors, either using Dicta's API or via automated scripts which catch common mistakes. We made an effort to collect a diverse set of sources: news, opinion columns, paragraphs from books, short stories, Wikipedia articles, governmental publications, blog posts and forums expressing various domains and voices, and more. Our MODERN corpus contains roughly 326K Hebrew tokens, and is much more consistent and similar to the expectation of a native Hebrew speaker than the PRE-MODERN or the AUTOMATIC corpora, and more accurately dotted than the AUTOMATIC corpus. The sources and statistics of this dataset are presented in Table 2.

2.3 New test set

Shmidman et al. (2020) provide a benchmark dataset for dotting modern Hebrew documents. However, it is relatively small and non-diverse: all 22 documents in the dataset originate in a single source, namely Hebrew Wikipedia articles.

Therefore, we created a new test set[8] from a larger variety of texts, including high-quality Wikipedia articles and edited news stories, as well as user-generated blog posts. This set consists of ten documents from each of eleven sources (5x Dicta's test set), and totals 20,474 Hebrew tokens, roughly 3.5x Dicta's. We use the same technique and style for dotting this corpus as we do for the MODERN corpus (§2.2), but the documents were collected in different ways.

3 Nakdimon

NAKDIMON embeds the input characters and passes them through a two-layer Bi-LSTM (Hochreiter and Schmidhuber, 1997). The LSTM output is fed into a single linear layer, which then feeds three linear layers, one for each diacritic category (see §2). Each character then receives a prediction for each category independently and all predicted marks are added to it as output.

Decoding is performed greedily, with no validation of readability or any other dependence between character-level decisions.

The input is pre-processed by removing all but Hebrew characters, spaces and punctuation; digits are converted to a dedicated symbol, as are Latin characters. All existing diacritic marks are stripped, and each document is split into chunks bounded at whitespace, ignoring sentence boundaries.

We train NAKDIMON first over PRE-MODERN, then over the AUTOMATIC corpus, and then by over the MODERN corpus. During training, the loss is the sum of the cross-entropy loss from all three categories. Trivial decisions, such as the label for the *shin/sin* diacritic for any non-שׁ letter, are masked.

Tuning experiments are detailed in Appendix B; an evaluation of a preliminary version of NAKDIMON over the Dicta test set is in Appendix C, and Hyperparameters are detailed in Appendix D.

4 Experiments

We compare the performance of NAKDIMON on our new test set (§2.3) against Dicta,[9] Snopi,[10] and Morfix (Kamir et al., 2002). as well as a MAJORITY baseline which returns the most common dotting for each word seen in our full training set.

Metrics We report four metrics: **decision accuracy (DEC)** is computed over the entire set of individual possible decisions: *dagesh/mappiq* for letters that allow it, *sin/shin* dot for the letter שׁ, and all other diacritics for letters that allow them; **character accuracy (CHA)** is the portion of characters in the text that end up in their intended final form (which may combine two or three decisions, e.g. *dagesh* + vowel); **word accuracy (WOR)** is the portion of words with no mistakes; and **vocalization**

[8] https://github.com/elazarg/hebrew_diacritized/tree/master/test_modern

[9] Version 4.0, wordlist version 43.

[10] http://nakdan.com/Nakdan.aspx

System	DEC	CHA	WOR	VOC
MAJORITY	93.79	90.01	84.87	86.19
SNOPI	91.29	85.84	76.45	78.91
MORFIX	96.84	94.92	90.38	92.39
DICTA	**97.95**	**96.77**	**94.11**	**94.92**
NAKDIMON	97.91	96.37	89.75	91.64

Table 3: Document-level macro % accuracy.

Context	Correct	Incorrect
... וצריך להסתכל לה בעיניים.	בְּעֵינַיִים	בְּעֵינַיִם
'...and we need to look her **in the eyes (/in eyes)**.'		
... יענו לך בסבלנות ...	לָךְ	לְךָ
'...**you.sg.f (/unreadable)** will be answered patiently...'		
... משחמשי האייפון הראשונים ...	הָאַיְיפוֹן	הָאִייפוֹן
'...the first **iPhone (/*ee-pon*)** users...'		

Table 4: Examples of words dotted incorrectly (top) or correctly (bottom) only by NAKDIMON.

accuracy (VOC) is the portion of words where any dotting errors do not cause incorrect pronunciation among mainstream Israeli Hebrew speakers.[11]

4.1 Results

We provide document-level macro-averaged accuracy percentage results for a single run over our test set in Table 3. All systems, except Snopi, substantially outperform the majority-dotting baseline on all metrics. NAKDIMON outperforms Morfix on character-level metrics but not on word-level metrics, mostly since Morfix ignores certain words altogether, incurring errors on multiple characters.

We note the substantial improvement our model achieves on the VOC metric compared to the WOR metric: 18.43% of word-level errors are attributable to vocalization-agnostic dotting, compared to 13.80% for Dicta and 10.41% for Snopi (but 20.91% for Morfix). Considering that the central use case for dotting modern Hebrew text is to facilitate pronunciation to learners and for reading, and that undotted homograph ambiguity typically comes with pronunciation differences, we believe this measure to be no less important than WOR.

Results on Dicta's test set (Shmidman et al., 2020) are presented in Appendix C.

4.2 Error analysis

In Table 4 we present examples of words dotted incorrectly, or correctly, only by NAKDIMON, compared with Morfix and Dicta. The largest category for NAKDIMON-only errors (∼18% of 90 sampled) are ones where a fused preposition+determiner character is dotted to only include the preposition, perhaps due to its inability to detect the explicit determiner clitic ה in neighboring words, on which the complex systems apply morphological segmentation. In other cases (∼15%), NAKDIMON creates unreadable vocalization sequences, as it has no lexical component and is decoded greedily. These types of errors are more friendly to the typical use cases of a dotting system, as they are likely to stand out to a reader. In contrast, a large portion of cases where only NAKDIMON was correct (∼13% of 152) are foreign names and terms. This may be the result of such words not yet appearing in dictionaries, or not being easily separable from an adjoining clitic, while character-level information can capture pronunciation patterns from similar words (e.g. טֶלֶפוֹן 'telephone', for the example האייפון).

OOVs To further quantify the strengths of NAKDIMON's architecture and training abilities, we evaluate the systems' results pertaining only to those words in the test set which do not appear in our training sets. We follow common practice by calling them OOVs ("out of vocabulary"), but emphasize that NAKDIMON does not consult an explicit vocabulary, and the other systems are not evaluated against their own vocabularies (which are unknown to us).

We find that NAKDIMON's performance on this subset is substantially **worse** compared with the other systems than on the full set: 15 percentage points below Dicta and seven below Morfix on the VOC metric (see full results in Appendix C).

These results might be counter-intuitive considering the proven utility of character-level models in OOV contexts (e.g., Plank et al., 2016), and so we offer several possible explanations: First, many "OOVs" consist in fact of known words coupled with an unseen combination of prefix clitics and/or suffix possessive markers, which other systems explicitly remove using morphological analyzers before dotting. Second, mirroring the last finding from the overall analysis, some "OOVs" are proper names which appear in dictionaries but are absent from the training set, due to corpus effects such as time and domain, or simply chance.

[11] These are: the *sin*/*shin* dot, vowel distinctions across the a/e/i/o/u/null sets, and *dagesh* in the פ/כ/ב characters. We do not distinguish between *kamatz gadol*/*kamatz katan*, and *schwa* is assumed to always be null.

5 Related Work

Existing work on diacritizing Hebrew is not common, and all efforts build on word-level features.

Kontorovich (2001) trains an HMM on a vocalized and morphologically-tagged portion of the Hebrew Bible containing 30,743 words, and evaluates the result on a test set containing 2,852 words, achieving 81% WOR accuracy. Note that Biblical Hebrew is very different from Modern Hebrew in both vocabulary, grammatical structure, and diacritization, and also has many words with unique diacritization. In our system, we exclude the Bible altogether from the training set, as its inclusion actively hurts performance on the validation set, which consists of Modern Hebrew.

Tomer (2012) designs a diacritization system for Hebrew verbs consisting of a combination of a verb inflection system, a syllable boundary detector, and an SVM model for classifying verb inflection paradigms. The focus on verbs in a type-level setup makes this work incomparable to ours or to others in this survey.

In Arabic, diacritization serves a comparable purpose to that in Hebrew, but not exclusively: most diacritic marks differentiate consonantal phonemes from each other, e.g. ب /b/ vs. ت /t/ (which only the sin/shin dot does in Hebrew), whereas vocalization marks are in a one-to-one relationship with their phonetic realizations, e.g. only the *fatha* as in بَ /ba/ encodes the /a/ vowel.

Dictionary-less Arabic diacritization has been attempted using a 3-layer Bi-LSTM (Belinkov and Glass, 2015). Abandah et al. (2015) use a Bi-LSTM where characters are assigned either one or more diacritic symbols. Our system differs from theirs by virtue of separating the diacritization categories. Mubarak et al. (2019) tackled Arabic diacritization as a sequence-to-sequence problem, tasking the model with reproducing the characters as well as the marks.

Zalmout and Habash (2017) have made the case against RNN-only systems, arguing for the importance of morphological analyzers in Arabic NLP systems. We concede that well-curated systems may perform better than uncurated ones, particularly on low-resource languages such as Hebrew, but we note that they are difficult to train for individual use cases and are burdensome to incorporate within larger systems.

Diacritics restoration in Latin-based scripts, applicable mostly to European languages, forms a substantially different problem from the one in Hebrew given the highly lexicalized nature of diacritic usage in these languages and the very low rate of characters requiring diacritics. The state-of-the-art systems in such languages employ transformer models in a sequence-to-sequence setup (Náplava et al., 2021; Stankevičius et al., 2022), supplanting character-RNN sequence prediction architectures reminiscent of ours (Náplava et al., 2018). Indeed, the authors of this latter work note the only non-European in their dataset, Vietnamese, as a special outlier.

6 Conclusion

Learning directly from plain diacritized text can go a long way, even with relatively limited resources. NAKDIMON demonstrates that a simple architecture for diacritizing Hebrew text as a sequence tagging problem can achieve performance on par with much more complex systems. We also introduce and release a corpus of dotted Hebrew text, as well as a source-balanced test set.

In the future, we wish to evaluate the utility of dotting as a feature for downstream tasks such as question answering, machine translation, and speech generation, taking advantage of the fact that our simplified model can be easily integrated in an end-to-end Hebrew processing system.

Ethical Considerations

We collected the data for our training set and test sets from open online sources, while making sure their terms allow research application and privacy is not impugned. NAKDIMON's architecture does not encourage memorization of training data and the system is not trained for generating text.

We consider a main use case for our system to be assisting Hebrew learners in reading. We therefore expect NAKDIMON to facilitate life in Israel for immigrants still struggling with Hebrew, among other underprivileged groups. Automatic dotting can increase inclusion in Hebrew-prominent societies for literacy-challenged individuals, and derivative

improvements in text-to-speech applications can assist those with impaired vision. Lastly, dotting can help researchers with limited understanding of Hebrew access resources in the language.

Hebrew is a gendered language. Orthographically, in many cases the lack of dots masks gender ambiguity, allowing both masculine and feminine readings for a given word (e.g. שְׁלַחְתְּ / שְׁלַחְתָּ 'you.fem sent' / 'you.masc sent'). While well-performing automatic dotting can help alleviate these ambiguities and reduce the amount of potentially prejudiced readings, we recognize the large body of work on gender bias in NLP (Blodgett et al., 2020), including in Hebrew NLP (Moryossef et al., 2019), and the findings that an imbalanced training set may result in an even more skewed distribution of gender bias in applications (Zhao et al., 2017). We believe our unlexicalized approach is more robust to such bias compared with other systems, and have already started quantifying and addressing these issues as we find them in ongoing work. In the meantime, we offer this paragraph as a disclaimer.

Acknowledgments

We would like to thank Avi Shmidman for details about Dicta's Nakdan and other suggestions. We thank Sara Gershuni for lengthy and fruitful discussions, and for her linguistic insights and advice. We thank Yoav Goldberg, Reut Tsarfaty, Ian Stewart, Sarah Wiegreffe, Kyle Gorman and many anonymous reviewers for their comments and suggestions in discussions and on earlier drafts.

References

Gheith 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)*, 18:183–197.

Salim Abu-Rabia. 2001. The role of vowels in reading semitic scripts: Data from Arabic and Hebrew. *Reading and Writing*, 14(1-2):39–59.

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

Shlomo Bentin and Ram Frost. 1987. Processing lexical ambiguity and visual word recognition in a deep orthography. *Memory & Cognition*, 15(1):13–23.

Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (technology) is power: A critical survey of "bias" in NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5454–5476, Online. Association for Computational Linguistics.

Ya'akov Gal. 2002. An HMM approach to vowel restoration in Arabic and Hebrew. In *Proceedings of the ACL-02 Workshop on Computational Approaches to Semitic Languages*, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Yoav Goldberg and Michael Elhadad. 2010. Easy-first dependency parsing of modern Hebrew. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 103–107, Los Angeles, CA, USA. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Dror Kamir, Naama Soreq, and Yoni Neeman. 2002. A comprehensive NLP system for modern standard Arabic and modern Hebrew. In *Proceedings of the ACL-02 Workshop on Computational Approaches to Semitic Languages*, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Leonid Kontorovich. 2001. Problems in Semitic NLP: Hebrew vocalization using HMMs. In *Problems in Semitic NLP, NIPS Workshop on Machine Learning Methods for Text and Images*.

Amit Moryossef, Roee Aharoni, and Yoav Goldberg. 2019. Filling gender & number gaps in neural machine translation with black-box context injection. In *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*, pages 49–54, Florence, Italy. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Jakub Náplava, Milan Straka, and Jana Straková. 2021. Diacritics restoration using bert with analysis on czech language. *arXiv preprint arXiv:2105.11408*.

Jakub Náplava, Milan Straka, Pavel Straňák, and Jan Hajič. 2018. Diacritics restoration using neural networks. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418, Berlin, Germany. Association for Computational Linguistics.

Danny Shacham and Shuly Wintner. 2007. Morphological disambiguation of Hebrew: A case study in classifier combination. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 439–447, Prague, Czech Republic. Association for Computational Linguistics.

Avi Shmidman, Shaltiel Shmidman, Moshe Koppel, and Yoav Goldberg. 2020. Nakdan: Professional Hebrew diacritizer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 197–203, Online. Association for Computational Linguistics.

Leslie N Smith. 2017. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. IEEE.

Lukas Stankevičius, Mantas Lukoševičius, Jurgita Kapočiūtė-Dzikienė, Monika Briedienė, and Tomas Krilavičius. 2022. Correcting diacritics and typos with ByT5 transformer model. *arXiv e-prints*.

Eran Tomer. 2012. *Automatic Hebrew Text Vocalization*. Ben-Gurion University of the Negev, Faculty of Natural Sciences, Department of Computer Science.

Reut Tsarfaty, Shoval Sadde, Stav Klein, and Amit Seker. 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 (EMNLP-IJCNLP): System Demonstrations*, pages 259–264, Hong Kong, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Nasser Zalmout and Nizar Habash. 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*, pages 704–713, Copenhagen, Denmark. Association for Computational Linguistics.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2979–2989, Copenhagen, Denmark. Association for Computational Linguistics.

System	Dicta – reported / reproduced				New test set (§2.3)					
									OOV	
	DEC	CHA	WOR	VOC	DEC	CHA	WOR	VOC	WOR	VOC
Baselines										
MAJMOD	84.93	75.94	68.10	69.63	88.04	81.22	76.14	77.10	N/A	N/A
MAJALL	91.67	86.29	79.43	81.19	93.79	90.01	84.87	86.19	N/A	N/A
Lexicalized										
SNOPI	87.81	78.96 / 79.92	66.41 / 66.57	70.35	91.29	85.84	76.45	78.91	40.83	42.39
MORFIX	94.91	90.32 / 91.29	80.90 / 82.24	86.48	96.84	94.92	90.38	92.39	63.91	69.20
DICTA	**97.53**	**95.12 / 95.71**	**88.23 / 89.23**	**90.66**	**97.95**	**96.77**	**94.11**	**94.92**	**76.21**	**77.66**
Unlexicalized										
NAKDIMON$_0$	95.78	92.59	79.00	83.01	94.59	91.70	84.94	87.54	47.05	50.96
NAKDIMON					97.91	96.37	89.75	91.64	57.46	62.06

Table 5: Document-level macro % accuracy on the test set from Shmidman et al. (2020) and on our new test set. We cannot report our full NAKDIMON's performance on the former, as we use the test set for parts of its training. MAJALL is reported as MAJORITY in the main text; MAJMOD only considers text in the MODERN portion of our training set.

A Full Script Reconciliation

We apply the following resolution tactics for added letters in undotted text: (a) We almost never remove or add letters to the original text (unless it is completely undiacritizable). (b) We keep *dagesh* in letters that follow a *shuruk* which replaces a *kubuts*, and similarly for yod (*hirik male* replacing *hirik haser*). (c) When we have double *vav* or double *yod*, the second letter is usually left undotted, except when it is impossible to have the correct vocalization this way.

Resolving *ktiv haser* discrepancies from Morfix outputs is done by adding missing vowel letters, or removing superfluous vowel letters, in such a way that would not count as an error if it is correct according to Academy regulations.

B Development Experiments

We tried to further improve NAKDIMON by initializing its parameters from a language model trained to predict masked characters in a large undotted Wikipedia corpus (440MB, 30% mask rate), but were only able to achieve an improvement of 0.07%. Attempted architectural modifications, including substituting a Transformer (Vaswani et al., 2017) for the LSTM; adding a CRF layer to the decoding process; and adding a residual connection between the character LSTM layers, yielded no substantial benefits in these experiments. Similarly, varying the number of LSTM layers between 2 and 5 (keeping the total number of parameters roughly constant, close to the 5,313,223 parameters of our final model) had little to no impact on the accuracy on the validation set.

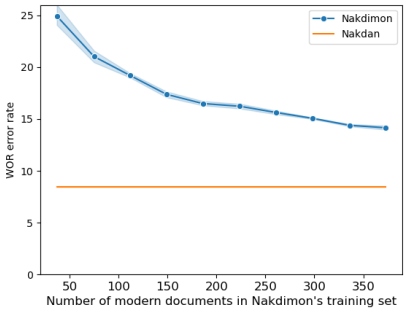

Figure 1: WOR error rate on validation set as a function of training set size vs. Dicta, over five runs. Other metrics show similar trends.

Figure 1 shows the favorable effect of training NAKDIMON over an increasing amount of MODERN text.

C Dicta Test Set

We present results for the Dicta test set in Table 5. In order to provide fair comparison and to preempt overfitting on this test data, we ran this test in a preliminary setup on a variant of NAKDIMON which was not tuned or otherwise unfairly trained. This system, NAKDIMON$_0$, differs from our final variant in three main aspects: it is not trained on the Dicta portion of our training corpus (§2.2), it is not trained on the AUTOMATIC corpus, and it employs a residual connection between the two character Bi-LSTM layers. Testing on the Dicta test set required some minimal evaluation adaptations resulting from encoding constraints (for example, we do not distinguish between *kamatz katan* and *kamatz gadol*). Thus, we copy the results reported in

Shmidman et al. (2020) as well as our replication.

We see that the untuned NAKDIMON$_0$ performs on par with the proprietary Morfix, which uses word-level dictionary data, consistent with our main results on our novel test set.

D Hyperparameters

We tuned hyperparameters and architecture over a held-out validation set of 40 documents with 27,681 tokens, on which Dicta performs at 91.56% WOR accuracy.

In our chosen setup, we train NAKDIMON over PRE-MODERN for a single epoch, followed by two epochs over the AUTOMATIC corpus, and then by three epochs over the MODERN corpus. We optimize using Adam (Kingma and Ba, 2014). For the PRE-MODERN corpus we use a cyclical learning rate schedule (Smith, 2017), varying linearly from $3 \cdot 10^{-3}$ through $8 \cdot 10^{-3}$ and down to 10^{-4}, which we found to be more useful than a constant learning rate. For each of AUTOMATIC and MODERN corpora we use epoch-wise decreasing learning rate: $(3 \cdot 10^{-3}, 10^{-3})$ and $(10^{-3}, 10^{-3}, 3 \cdot 10^{-4})$ respectively. We set maximum chunk size to 80 characters, and use batch size of 128. We set both character embedding and LSTM hidden dimensions to 400, and apply a dropout rate of 0.1.