Restoring Missing Spaces in Scraped Hebrew Social Media

Avi Shmidman,[1,2,†] Shaltiel Shmidman[1,‡]
[1]DICTA, Jerusalem, Israel
[2]Bar-Ilan University, Ramat Gan, Israel
[†]avi.shmidman@biu.ac.il
[‡]shaltieltzion@gmail.com

Abstract

A formidable challenge regarding scraped corpora of social media is the omission of spaces, causing pairs of words to be conflated together as one. In order for the text to be properly parsed and analyzed, these missing spaces must be detected and restored. However, it is particularly hard to restore whitespace in languages such as Hebrew which are written without vowels, because a conflated form can often be split into multiple different pairs of valid words. Thus, a simple dictionary lookup is not feasible. In this paper, we present and evaluate a series of neural approaches to restore missing spaces in scraped Hebrew social media. Our best all-around method involved pretraining a new character-based BERT model for Hebrew, and then fine-tuning a space restoration model on top of this new BERT model. This method is blazing fast, high-performing, and open for unrestricted use, providing a practical solution to process huge Hebrew social media corpora with nothing more than a consumer-grade GPU. We release the new BERT model and the fine-tuned space-restoration model to the NLP community.

1 Introduction

Scraped corpora of social media tend to contain many instances of missing spaces, where two or more words have been run together as one. This phenomenon is likely due to the fact that the HTML source of internet pages often encodes different parts of the text in distinct HTML tags, without explicit indication of whether two consecutive tags contain a single word or two separate words. Scraping algorithms exercise various heuristics to decide whether to add a space or not; however, these heuristics do not always succeed. In practice, the NLP researcher is often faced with digital corpora of scraped social media in which a substantial number of lines are corrupted with conflated words. These missing spaces can impair downstream tasks such as parsing, segmentation, and information retrieval. Additionally, when these corpora are used to train language models, the errors are propagated forward into the model. Thus, it is essential to restore missing spaces wherever possible.

The problem of missing spaces is particularly acute in languages such as Hebrew, in which words are generally written as consonants alone. The omission of vowels results in extreme ambiguity, such that a given sequence of letters can generally be interpreted as multiple different words (Tsarfaty et al., 2019). Crucially, this means that when two words are run together, they generally cannot be separated by means of a simple dictionary lookup, because there are multiples ways of splitting the conflated sequence into two valid Hebrew words.

For instance, here is an actual line contained in the Hebrew section of the public OSCAR corpus (Ortiz Suárez et al., 2020): רמת קושיכל אחד יכול ("Level of difficulty Everyone can do it"). The words קושי ("difficulty") and כל ("every") are conflated together in the corpus as a single string. However, there is more than one way to split this string; if we were to apply a dictionary lookup, we could also split it into the two words קו ("line") and שיכל ("transposed"). Another line in the same corpus contains the conflated string קריאהבמה, which can be split into קריאה ("reading") and במה ("in what"), or into קריא ("readable") and הבמה ("the stage"). Hundreds of thousands of additional sentences within the Hebrew portion of the OSCAR corpus are similarly corrupted. Yet, the OSCAR internet crawl is the primary component of virtually all publically-available Hebrew BERT models, including heBERT (Chriqui and Yahav, 2021), AlephBERT (Seker et al., 2022), and AlephBertGimmel (Gueta et al., 2023).

An efficient context-aware method is needed to fix this. In this paper, we present and evaluate a series of neural approaches for the restoration of the missing spaces within social-media corpora.

2 Task Definition

We formalize the space-restoration task as follows: given an input string s with characters $c_1 ... c_n$ where $|s| = n$, our goal is to predict a binary label for each c_i, indicating whether a space should appear before the character at that position. This formulation treats the problem as a character-level sequence labeling task, which is particularly suitable for languages like Hebrew where subword boundaries must be handled carefully.

3 Neural Models for Space Restoration

In this study, we develop and evaluate a series of neural models for the restoration of the missing spaces.

3.1 Existing Encoder Models for Hebrew

Existing Hebrew encoder-based models such as mBERT (Devlin et al., 2018), AlephBertGimmel (Gueta et al., 2023), HeRo (Shalumov and Haskey, 2023), and DictaBERT (Shmidman et al., 2024b) are trained with a wordpiece tokenizer, which obscures character-level information and impairs their ability to perform well on character-level labeling tasks. In contrast, TavBERT (Group, 2023) adopts a character-based representation for Hebrew, preserving full granularity over all character positions. TavBERT thus opens the door to the possibility of training an encoder-based model to perform char-level predictions for whitespace restoration.

3.2 A New Character-based Encoder Model

As noted, TavBERT provides a possible basis for training a model to provide character-level predictions for Hebrew words. Nevertheless, at its core, TavBERT was designed with word prediction in mind; accordingly, it was trained with a SpanBERT-style objective, wherein the model is trained to predict a series of consecutive masked characters, rather than just a single character. Indeed, as we will see below (Section 5), fine-tuning TavBERT for this task results in a low-performing model.

Therefore, as part of this study, we pretrain a new character-level BERT model for Hebrew, dubbed `DictaBERT-char`. Our new model is pretrained on the standard BERT masked-language-modeling objective at the character level; that is, it is trained to predict single masked characters, rather than spans or wordpieces. We hypothesize that this will produce a model that is more robustly tuned to the fine-grained requirements of character-level tasks, such as the space-restoration task.[1]

In order to pretrain this new model, we adopt the same essential training setup and corpus used in the training of the Hebrew BERT model DictaBERT, which has been shown to be highly successful on a wide variety of NLP tasks (Shmidman et al., 2024b). We make two key modifications: (1) We use a purely character-level tokenizer to fully capture potential space boundaries, and (2) we set a consistent context length of 2048 throughout training (rather than gradually scaling from 256 to 512), in order to address the lower compression ratio when working at the character level.

The model was trained on a DGX Workstation with 4xA100 40GB GPUs for a total of 31,600 steps. Each step included a batch size of 4,096 examples, where each example had a context length of (up to) 2,048 tokens in order to accommodate the character-level tokenizer. The rest of the parameters, including the training corpus, are the same as DictaBERT, detailed by Shmidman et al. (2024b).

3.3 Decoder-based Models (LLMs)

Generative large language models (LLMs) have demonstrated remarkable capabilities across many NLP tasks, including sequence-to-sequence problems. As these models are generative, we can leverage their ability to generate free-form text to solve char-level tasks such as our space-restoration task. We explore two avenues regarding LLMs.

First, we fine-tune an open-weight LLM. We use Dicta-LM 2.0 (Shmidman et al., 2024a), a 7B parameter LLM continuously trained in Hebrew (based on Mistral-7B (Jiang et al., 2023)). This model is particularly strong regarding Hebrew tasks, as indicated by its position on the Hebrew LLM Leaderboard.[2]

Second, we evaluate a prompt engineering approach with two state-of-the-art proprietary LLMs: GPT-4o and GPT-4o-Mini (OpenAI, 2024).

4 Experimental Setup

4.1 Training Corpus

The training data set was created automatically by augmenting texts and removing spaces randomly.

[1] We release this model to the public on HuggingFace under the CC-BY-4.0 license: https://huggingface.co/dicta-il/dictabert-char

[2] https://huggingface.co/spaces/hebrew-llm-leaderboard/leaderboard

We start with a collection of 150,000 Hebrew sentences from high-quality Hebrew corpora.[3] Next, in 20% of the sentences, we randomly remove between 1 and 4 spaces.

4.2 Test Corpus

The test data set contains 6,000 sentences, and was created similarly to the training data, with three important caveats:

1. We wish to minimize the likelihood of the models having previously seen any of these sentences. Therefore, we collect the test corpus sentences from the newly-released FineWeb2 corpus (Penedo et al., 2024), after removing any documents that appeared in previous Hebrew corpora (such as OSCAR (Ortiz Suárez et al., 2020) and mC4 (Xue et al., 2021)).

2. In order to focus the evaluation metrics on the ability of the models to handle missing spaces, we removed 1-4 random spaces from each of the sentences in the test corpus.

3. To ensure that the test data reflects real-world challenging cases, we only remove spaces between two words, rather than before or after punctuation marks. Missing spaces next to punctuation can easily be fixed using rule-based methods; the word-conflation errors are where we need a neural model.

We release the test corpus to the NLP community so that future studies can reproduce and compare to the results of this paper.[4]

4.3 Training Details

4.3.1 Training Encoder Models

We train the encoder models on the sequence labeling task described above (Section 2). For each character, the models are fine-tuned to predict a binary label indicating whether a space should appear before it or not.

The BERT encoders generate contextualized character representations, followed by a linear layer that maps these representations to logits. We train by minimizing the cross-entropy loss between the predicted logits and the true labels.

During initial fine-tuning, we observed that the model almost exclusively predicted the negative label, likely due to the the class imbalance (the average sentence had 115 characters but fewer than 4 missing spaces - a ratio of roughly 99:1). To address this, we trained the encoder models only on the 20% of examples where spaces were removed. Within each example, we also downsampled the negative labels, keeping only 10% of the $l_i = 0$ labels, ensuring a more balanced ratio.

The hyperparameters and loss graphs are detailed in Appendix B.

4.3.2 Training The Decoder-based Model

For the decoder-based fine-tuned model, we train on the full training data, where 80% of the examples had no spaces removed. We use a supervised fine-tuning (SFT) approach, similar to instruction tuning in large language models (Zhang et al., 2024).

Each example was formatted as:

```
[SRC] {input sentence} [/SRC]
{output sentence}</s>
```

We compute the loss only on the completion (i.e., the output sentence), ensuring that the model focuses on predicting the correct restoration of spaces. Since most of the training examples already contained correctly spaced text, this setup allowed the model to learn both how to copy well-formed sentences and also how to correct corrupted ones without being biased toward over-inserting spaces.

The hyperparameters and loss graphs are detailed in Appendix C.

During testing, we constrained the model's output by using guided decoding in the inference engine, in order to prevent any alterations other than additional spaces. This allowed for a reliable evaluation of its ability to restore proper spacing, without the need to worry that the generative model might otherwise modify the text.

4.3.3 Prompting General-Purpose LLMs

To craft an ideal prompt, we used OpenAI's o1 model (OpenAI, 2024). We provided the model with a definition of the task as a character-level sequence labeling task, and we emphasized that the prompt should clearly instruct the model to modify only spaces, without altering any other characters. The final prompt can be found in Appendix D.

[3] The sentences are gathered from high-quality Hebrew corpora such as newspapers and ebooks, rather than from scraped social media, to avoid the possibility that these sentences themselves might already be corrupted with missing spaces.

[4] https://huggingface.co/datasets/dicta-il/hebrew-space-restoration-corpus

When testing, we verify that the model's output is "valid", that is, identical to the input except for the addition of spaces. If the model makes any other modifications, we treat the sentence as unchanged, since we cannot reliably evaluate an output that differed beyond spacing adjustments. GPT-4o produced valid outputs 95.2% of the time, while GPT-4o-Mini produced valid outputs only 83.5% of the time.

5 Results

We evaluate each model's ability to accurately restore all missing spaces across the sentences in our test corpus. We compute precision, recall, and F1 score for restoring a space (a positive label). Results are presented in Table 1.

Model	**Precision**	**Recall**	**F1**
Our Model (T=0.5)	90.1%	**99.0%**	.943
Our Model (T=0.9)	97.0%	96.7%	.968
tau/TavBERT	13.0%	13.4%	.131
DictaLM2.0 (FT)	97.5%	97.9%	**.976**
DictaLM2.0-AWQ (FT)	97.2%	97.7%	.974
GPT-4o	**98.9%**	93.6%	.961
GPT-4o-Mini	97.7%	84.5%	.906

Table 1: Performance on the space restoration task, measured in terms of precision, recall, F1 for positive labels (i.e., correctly adding a missing space). For our model we presented results with two different thresholds.

The fine-tuned 7B-parameter decoder model (Dicta-LM 2.0) outperforms the other methods, with our new character-based model not far behind, with both of these models being lightweight enough to run on consumer hardware. Additionally, we evaluate a 4-bit quantized version of the model using AWQ quantization (Lin et al., 2023). This version requires only 5GB of VRAM to run efficiently and performs nearly as well as the full-precision model.

To provide a more realistic view of practical usage, we compared the performance of the 7B models and the encoder model, as shown in Table 2. Both were evaluated on an RTX 4090 GPU; we ran our char-based BERT model using the standard HuggingFace implementation, and we ran the DictaLM model via vLLM.

Our char-based BERT model outputs logit values, which are then transformed using softmax into normalized scores between 0 and 1, allowing us to set a confidence threshold. Based on this, we present a graph of its metrics across different thresholds in Figure 1. Notably, when setting the threshold to 0.9, the F1 score surpasses that of GPT-4o, as shown in Table 1, and is only slightly shy of the F1 score achieved by model based on DictaLM 2.0 model (0.968 vs. 0.976). Furthermore, our char-based BERT model runs nearly 30 times faster than DictaLM2.0 with guidance, making it a highly efficient method for real-world corpora. In Appendix A we present examples of output from the model when run on real-world Hebrew sentences, with a qualitative analysis of its successes and failures.

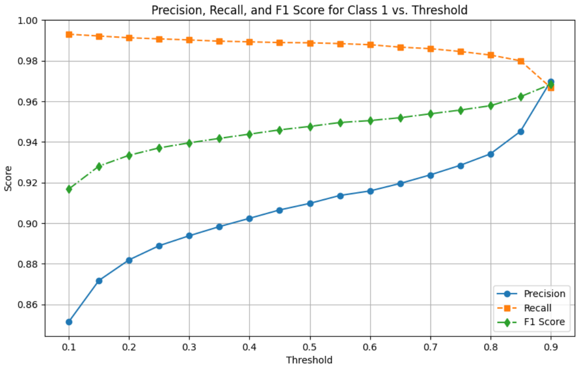


Figure 1: Precision, Recall, and F1 score of our new char-based BERT model across different confidence thresholds, when run on the test set.

Model	**Time (s)**	**Invalid**
New char-based Hebrew BERT	23.46	0
DictaLM2.0 (not guided)	616.8	3.6%
DictaLM2.0 (guided)	676.8	0
DictaLM2.0-AWQ (not guided)	437.3	16.9%
DictaLM2.0-AWQ (guided)	1081.5	0

Table 2: Inference time comparison of the models running on 12,000 sentences on an RTX 4090. The "guided"/"not guided" label indicates whether the model was run with or without the guided backend enforcing valid output (i.e., restricting modifications to only adding spaces. This increases runtime since the engine has to construct a new predictions tree for each input). The third column notes the percentage of outputs that were invalid, where the model altered more than just spaces.

6 Conclusion

Almost all of the methods presented here provide decent accuracy on the space restoration task. However, because scraped social media corpora tend to be huge, the decoder-based methods are largely impractical, due to issues of speed (thus for the open Dicta-LM model), or cost (thus regarding the commercial LLM models). Fortunately, the fine-tuned character-BERT model which we presented here provides a practical solution: it is blazing fast, free for unrestricted use, and achieves accuracy which rivals the other methods. We are thus pleased to hereby release this model to the NLP community.

Furthermore, the character-based Hebrew BERT model which we pretrained as part of this project is released here as well, so that NLP researchers can continue to fine-tune it for other character-level NLP tasks as well.

Acknowledgements

This work has been funded by the Israel Science Foundation (Grant No. 2617/22) and by the European Union (ERC, MiDRASH, Project No. 101071829; Principal investigators: Nachum Dershowitz, Tel-Aviv University; Judith Olszowy-Schlanger, EPHE-PSL; Avi Shmidman, Bar-Ilan University, and Daniel Stoekl Ben Ezra, EPHE-PSL), for which we are grateful. Views and opinions expressed are, however, those of the authors only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

References

Avihay Chriqui and Inbal Yahav. 2021. Hebert & hebemo: a hebrew bert model and a tool for polarity analysis and emotion recognition. *Preprint*, arXiv:2102.01909.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Tau NLP Group. 2023. Tavbert: Hebrew character-based bert model. Accessed: 2025-01-21.

Eylon Gueta, Avi Shmidman, Shaltiel Shmidman, Cheyn Shmuel Shmidman, Joshua Guedalia, Moshe Koppel, Dan Bareket, Amit Seker, and Reut Tsarfaty. 2023. Large pre-trained models with extra-large vocabularies: A contrastive analysis of hebrew bert models and a new one to outperform them all. *Preprint*, arXiv:2211.15199.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *arXiv preprint*, arXiv:2310.06825.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2023. AWQ: Activation-aware weight quantization for LLM compression and acceleration. *arXiv preprint arXiv:2306.00978*.

OpenAI. 2024. GPT-4o system card. *arXiv preprint arXiv:2410.21276*.

OpenAI. 2024. OpenAI o1 System Card. *arXiv preprint arXiv:2412.16720*.

Pedro Javier Ortiz Suárez, Laurent Romary, and Benoît Sagot. 2020. A monolingual approach to contextualized word embeddings for mid-resource languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1703–1714.

Guilherme Penedo, Hynek Kydlíček, Vinko Sabolčec, Bettina Messmer, Negar Foroutan, Martin Jaggi, Leandro von Werra, and Thomas Wolf. 2024. Fineweb2: A sparkling update with 1000s of languages.

Amit Seker, Elron Bandel, Dan Bareket, Idan Brusilovsky, Refael Greenfeld, and Reut Tsarfaty. 2022. AlephBERT: Language model pre-training and evaluation from sub-word to sentence level. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 46–56, Dublin, Ireland. Association for Computational Linguistics.

Vitaly Shalumov and Harel Haskey. 2023. Hero: Roberta and longformer hebrew language models. *arXiv:2304.11077*.

Shaltiel Shmidman, Avi Shmidman, Amir DN Cohen, and Moshe Koppel. 2024a. Adapting llms to hebrew: Unveiling dictalm 2.0 with enhanced vocabulary and instruction capabilities. *Preprint*, arXiv:2407.07080.

Shaltiel Shmidman, Avi Shmidman, Moshe Koppel, and Reut Tsarfaty. 2024b. MRL parsing without tears: The case of Hebrew. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 4537–4550, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

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.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. 2024. Instruction tuning for large language models: A survey. *Preprint*, arXiv:2308.10792.

A Appendix: Qualitative Analysis

We survey here a set of representative examples of the successes and failures of the best all-around model presented in the paper (that is, the model based upon the new Hebrew character-based BERT released with this paper), with the threshold set to 0.9 (the optimal threshold, as per Figure 1 in the paper).

All input examples in this section are taken from the publically-available OSCAR internet crawl (Ortiz Suárez et al., 2020).

A.1 Successes

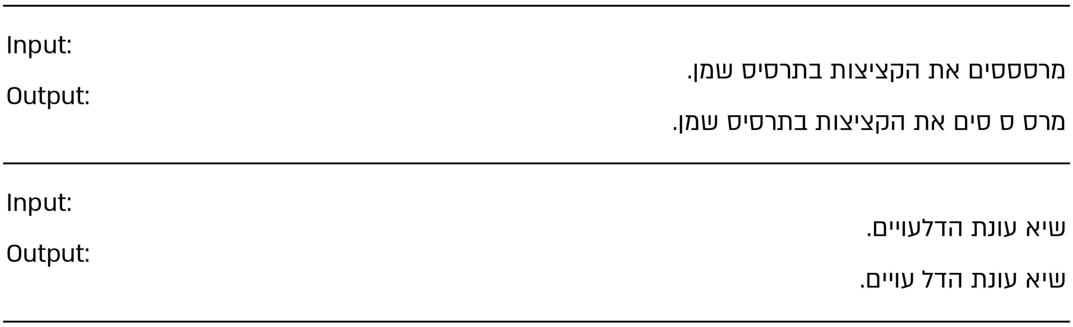
We first present a series of cases where the OSCAR lines are missing one or more spaces and our model successfully restores the spaces in the proper places. These cases demonstrate the strengths and capabilities of the model:

Input:	תשובותהלכתיות – הרבישראליוסףהכהןשיחי’ הנדל, רבקהילתחב”דמגדלהעמק.
Output:	תשובות הלכתיות – הרב ישראל יוסף הכהן שיחי’ הנדל, רב קהילת חב”ד מגדל העמק.
Input:	משה אליסקבלן מפתח
Output:	משה אליס קבלן מפתח
Input:	פלפלשחורגרוס
Output:	פלפל שחור גרוס
Input:	כדאי לנו מאוד לשים לב לכלל הזה, כי הוא דיחשוב, ואנחנו ממש עשויים לעבור עליו מעת לעת.
Output:	כדאי לנו מאוד לשים לב לכלל הזה, כי הוא די חשוב, ואנחנו ממש עשויים לעבור עליו מעת לעת.
Input:	יש לוודא כי הקבלן יהיה אחראי לתיקון רישום הבית המשותףוהדירות החדשות .
Output:	יש לוודא כי הקבלן יהיה אחראי לתיקון רישום הבית המשותף והדירות החדשות .
Input:	הכולטלוויזיהמוזיקהמחולקולנועתיאטרון
Output:	הכול טלוויזיה מוזיקה מחול קולנוע תיאטרון
Input:	אז תמשיך להצליחואני אמשיך להנות מכך.
Output:	אז תמשיך להצליח ואני אמשיך להנות מכך.
Input:	זמן הכנהשלושת רבעי שעה
Output:	זמן הכנה שלושת רבעי שעה

A.2 Failures

Next, we identify three categories where our model tends to fail:

Failures due to additional typos: When the input text contains additional typographical errors beyond the missing spaces, our model will sometimes attempt to add spaces in the middle of misspelled words words, as in the following examples:



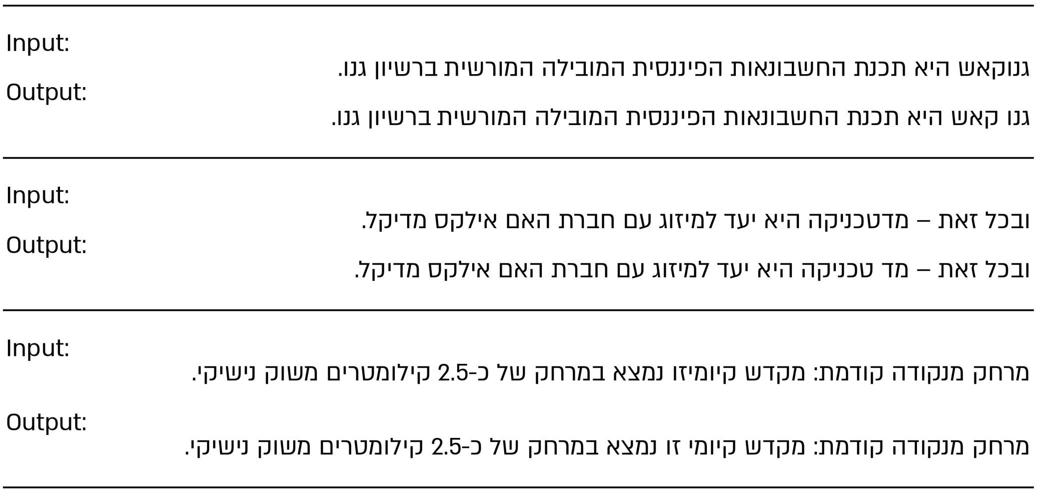
Input: מרסססים את הקציצות בתרסיס שמן.

Output: מרס ס סים את הקציצות בתרסיס שמן.

Input: שיא עונת הדלעויים.

Output: שיא עונת הדל עויים.

Failures due to proper nouns: Our model does not always recognize proper nouns for what they are, and sometimes attempts to divide them into two, especially when one (or both) of the resulting pieces is a common Hebrew word.



Input: גנוקאש היא תכנת החשבונאות הפיננסית המובילה המורשית ברשיון גנו.

Output: גנו קאש היא תכנת החשבונאות הפיננסית המובילה המורשית ברשיון גנו.

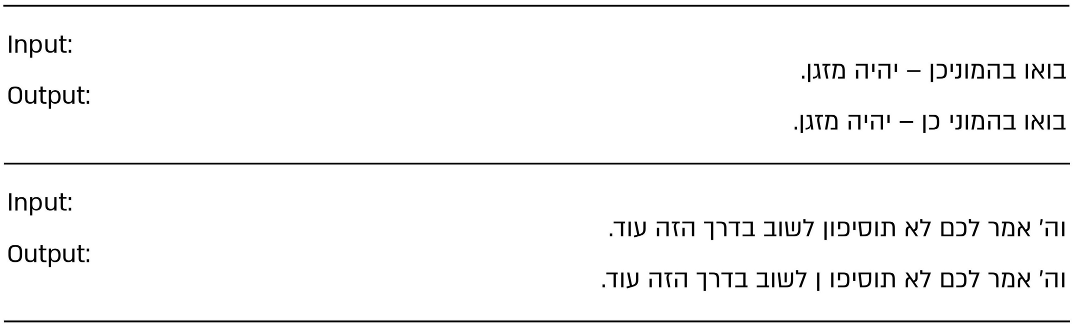
Input: ובכל זאת – מדטכניקה היא יעד למיזוג עם חברת האם אילקס מדיקל.

Output: ובכל זאת – מד טכניקה היא יעד למיזוג עם חברת האם אילקס מדיקל.

Input: מרחק מנקודה קודמת: מקדש קיומיזו נמצא במרחק של כ-2.5 קילומטרים משוק נישיקי.

Output: מרחק מנקודה קודמת: מקדש קיומי זו נמצא במרחק של כ-2.5 קילומטרים משוק נישיקי.

Failures due to unusual grammatical suffixes: When the input text contains relatively long words which also contain a relatively rare grammatical suffix, our model is sometimes fooled and attempts to add a space before the grammatical suffix, as in the following examples:



Input: בואו בהמוניכן – יהיה מזגן.

Output: בואו בהמוני כן – יהיה מזגן.

Input: וה' אמר לכם לא תוסיפון לשוב בדרך הזה עוד.

Output: וה' אמר לכם לא תוסיפו ן לשוב בדרך הזה עוד.

In light of these failures, practical use of the model would entail use of additional filters in order to restrain the model from splitting too eagerly. For instance, an NER model could be used to identify proper names in the text, and to restrain the space restoration model from splitting those names. Similarly, in order to avoid the issue with grammatical suffixes, a script could check whether the letters after a word-split form of the few dozen sequences of letters which comprise Hebrew grammatical suffixes; in such cases, it would be wise to ignore the additional space predicted by the model.

B Appendix: Encoder Training Details

The models were fine-tuned on a single NVIDIA A10G GPU. We used a learning rate of $2e-6$, and a batch size of 16. We trained using mixed BF16 precision, with 500 warmup steps (27%). You can view the loss graph from the fine-tuning of both `tau/tavbert-he` and of our new char-based Hebrew BERT model in Figure 2. Total train runtime was 350 seconds for 30,000 training examples.

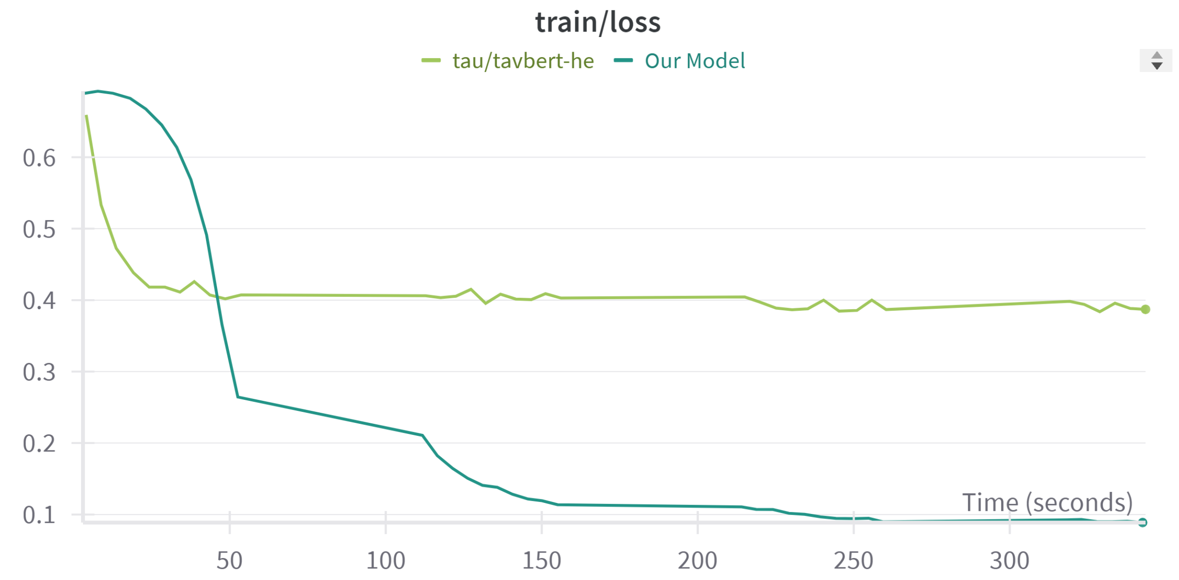


Figure 2: Training loss graph when fine-tuning our new char-based Hebrew BERT model and tau/tavbert-he

C Appendix: Decoder Training Details

The fine-tuning of Dicta-LM 2.0 was done on an NVIDIA-DGX with 4xA100 40GB. The training was done using the HuggingFace TRL library together with DeepSpeed. We set the initial learning rate to 5e-6, with a global batch size of 128 (per device batch size of 4, with 8 gradient accumulation steps). We used the `adamw_8bit` optimizer provided by the bitsandbytes library. Total training time was 110 minutes for 150,000 examples.

D Appendix: Prompt for General Purpose LLMs

Below is the entire prompt used with GPT-4o to complete the missing-spaces task:

```
You are a specialized tool for detecting and correcting missing spaces in Hebrew
↪   text. In the next message, I will provide you with a single Hebrew sentence.
↪   Your task is:

1. Identify any places in the sentence where spaces between words have been omitted.
2. Reinsert these missing spaces so that the sentence becomes correctly spaced.
3. Preserve all other text exactly as it appears in the input. This means:
    • Do not alter any words beyond adding missing spaces.
    • Do not, under any circumstance, add any letters!
    • Do not change or add punctuation.
    • Do not correct spelling or grammar (unless it solely involves inserting
    ↪   spaces).
    • Do not rearrange or remove any words or letters.
    • Do not add or modify diacritics (niqqud).

Your output must be the exact same sentence, in Hebrew, with the only difference
↪  being the addition of the missing spaces. If there are no missing spaces,
↪  return the exact input sentence verbatim.

{input sentence}
```