Tokemon: A Root Aware Lossless Morphological Tokenizer for Hebrew and Aramaic

Technical Report, Public Version

Ilan Ohayon David Busbib

La Boétie

July 2026

Abstract

Subword tokenizers such as byte pair encoding (BPE) and unigram language models were shaped by the statistics of English and similar languages. On Hebrew and Aramaic, two root and pattern languages, they cut words at boundaries unrelated to the morphology: the token count rises and the consonantal root, which the writing system marks, is split across fragments. We present results for Tokemon, a root aware, fully lossless morphological tokenizer for Hebrew and Aramaic with a 16,384 token vocabulary. Tokemon segments each word into an interpretable sequence of prefix, root, pattern mark, and suffix tokens, so a single root recurs as one token across the inflected forms it appears in. On the Tanakh it spends **1.30** tokens per morphological word and recovers **98.07%** of gold roots as intact tokens, with exact round trip reconstruction including final letter forms. Round trip is verified with zero failures across roughly 1.8 million words spanning biblical, mishnaic, halakhic, and held out talmudic text, and at corpus scale across 286 million words. On the same running text and word basis it is 1.3 to 1.6 times tighter than o200k, the production tokenizer behind current ChatGPT models, and about three times tighter than the earlier cl100k, while being competitive with a Hebrew trained BPE at an equal vocabulary budget. This report states the design principles and the measured results; the segmentation algorithm, cost model, vocabulary construction procedure, and mark encoding are documented internally and are not disclosed here.

Scope of this report. Tokemon is a proprietary system. This document reports what the tokenizer does and how well it does it: the problem it addresses, the properties it guarantees, the evaluation protocol, and the measured results on the frozen release artifact. It intentionally does not specify how the system is built. The minimum cost segmentation procedure and its cost model, the vocabulary construction recipe and layer budgets, the encoding of the reversible morphological marks, the grammar rule inventory, and the override generation machinery are covered in an internal companion paper and are withheld from this version. Every number reported below is measured, and the evaluation basis is stated precisely so the claims can be checked against the released artifact.

1 Introduction

For English, subword tokenization is largely settled. Byte pair encoding [1, 2] learns frequent character sequences and the resulting fragments approximately follow morpheme and word boundaries. For Hebrew and Aramaic this approximation breaks down.

Hebrew and Aramaic are root and pattern (templatic) languages. A consonantal root, usually three letters, is set into a vocalic pattern and wrapped in prefixes and suffixes to form a family of related words. The root

כתב (*k-t-b*, "to write") surfaces as כָּתַב (*katav*, "he wrote"), כְּתִיבָה (*ketivah*, "writing"), הִכְתִּיב (*hikhtiv*, "he dictated"), and מִכְתָּב (*mikhtav*, "a letter"). A fluent reader perceives one lexical family. A byte pair tokenizer sees four unrelated strings and splits each into whichever fragments were frequent in training, so the root never appears as a stable unit.

This has two consequences. First, fertility, the mean number of tokens per word, is high: measured on the same normalized text and word basis, o200k spends about 2.3 tokens per word on unvocalized Hebrew and the earlier cl100k about 5 to 6, so a fixed context window fills with encoding overhead. Second, the model must reconstruct from co-occurrence statistics that the inflected forms share a root, a relation the orthography already encodes. Tokemon supplies that relation directly, keeping the root visible in the token stream, which concentrates the training signal on real morphemes instead of frequency artifacts.

A recurring worry about morphologically motivated segmentation is fidelity: Hebrew has five letters whose form changes at word end (final or *sofit* letters), and a tokenizer that cuts a word incorrectly can leave a medial letter where a final belongs, silently altering the text. Tokemon is designed so that decoding is the exact inverse of encoding on consonantal Hebrew and Aramaic, final forms included, which we verify by round tripping entire corpora word by word.

Contributions of this report.

1. A statement of the design principles of a root aware, lossless morphological tokenizer for Hebrew and Aramaic with a 16,384 token vocabulary, at the level of guaranteed properties rather than construction (Section 4).
2. An empirical evaluation on the Tanakh, Mishnah, Mishneh Torah, Midrash Rabbah, and held out Talmud, reporting fertility, gold root preservation for Hebrew and Aramaic, and exact round trip, together with a head to head comparison against a Hebrew trained BPE, a Hebrew trained unigram model, and the tiktoken encodings cl100k and o200k (Sections 5 and 6).
3. Corpus scale throughput and integrity results: the full 286 million word Sefaria corpus tokenizes in minutes and round trips with zero failures (Section 7).
4. A short account of the design space, including two instrumented designs (gematria token identities and factored embeddings) that were built, measured, and abandoned in favor of the shipped tokenizer (Section 8).

2 Background: Hebrew and Aramaic morphology

This section states the descriptive linguistics the design relies on. It is drawn from standard reference grammars and from the OSHB morphology codes used to tag the Tanakh.

Roots. The lexical core of a word is a consonantal root (שֹׁרֶשׁ, *shoresh*), almost always three letters, occasionally two or four. The root carries the semantic field. It never appears in isolation; it is always realized through a pattern.

Patterns. A pattern (a בִּנְיָן, *binyan*, for verbs; a מִשְׁקָל, *mishkal*, for nouns) is a template of vowels and affixal slots into which the root consonants are placed. The seven Hebrew verbal binyanim (*qal*, *nifal*, *piel*, *pual*, *hifil*, *hufal*, *hitpael*) and the noun mishkalim yield different but related meanings from one root: causative, reflexive, intensive, agentive, instrumental.

Affixes. Proclitic prefixes mark conjunctions, prepositions, the definite article, and the relative particle; suffixes mark person, number, gender, and pronominal objects. A single orthographic word can therefore bundle what other languages spread across several words, for example וּכְשֶׁמַּלְכְּתוֹ (*ukhshemalkhuto*, "and when his kingship").

Weak roots. Patterns deform predictably when the root contains weak letters. Initial נ assimilates; middle ו/י (hollow roots) elide; final ה/י shift; geminate roots collapse a doubled radical. The surface therefore often lacks a radical of the underlying root, or carries an inserted mater lectionis (ה/א/י/ו) that is not part of the root. A tokenizer that wants the root visible must model these regular deformations.

Aramaic. Talmudic, Targumic, and Biblical Aramaic use the same architecture with their own patterns and function words: the relative and genitive proclitic ד (*d-*), the emphatic state suffix א (*-a*) in place of the Hebrew article, masculine plural ין (*-in*) rather than ים (*-im*), the causative *afel*/*haphel* and reflexive *itpeel*/*itpaal* stems, and metathesis or voicing of the *itpeel* ת after sibilants (for example אִזְדַּמִּן, *izdamen*, where the ת surfaces as ד after ז). Hebrew and Aramaic are interleaved within single Talmudic sentences, so one tokenizer must cover both.

3 Related work

Subword tokenization. Byte pair encoding was introduced as a compression algorithm [1] and adapted to open vocabulary neural machine translation [2]. WordPiece [3] and the unigram language model [4] are the other common families; SentencePiece [5] packages BPE and unigram with byte fallback and language independent preprocessing, and underlies most current tokenizers. These methods optimize corpus compression, not linguistic structure, and their segments align only loosely with morphemes.

Tokenization and morphology. A body of work shows that the segments these methods produce are a poor fit for morphologically rich languages. Bostrom and Durrett [6] find that unigram segmentation aligns better with morphology than BPE and improves pretraining. Klein and Tsarfaty [7] show directly that word pieces are inadequate for modelling the complex morphology of Hebrew. Hofmann et al. [8] show that injecting derivational morphology into BPE improves segmentation. For number tokenization, Singh and Strouse [11] show that how digits are grouped changes arithmetic accuracy substantially, a further sign that segmentation choices carry downstream signal.

Efficiency and fairness. Rust et al. [9] quantify how much a language's tokenizer matters for a multilingual model, and Petrov et al. [10] show that shared tokenizers charge speakers of underserved languages many more tokens for the same content, a fairness and cost gap that is acute for Hebrew and Aramaic under English-centric encodings. Fertility, the tokens per word measure we adopt, is the standard proxy for this overhead.

Hebrew and Aramaic NLP. Hebrew NLP has long treated morphological segmentation as a first class problem [12, 13]. Recent Hebrew language models, AlephBERT [14] and, for rabbinic Hebrew and Aramaic, BEREL [15], are trained on standard subword tokenizers and inherit their fragmentation. For the wider Semitic family, Arabic NLP relies on dedicated morphological analysis and segmentation [16]. Tokemon differs from all of these in that segmentation is morphological by construction, the root is a token rather than a latent variable, and the mapping from ids back to text is exact.

4 Design principles and system properties

Tokemon has three design goals: the root is a recurring token (*root aware*); decoding reproduces the input exactly on consonantal Hebrew and Aramaic, final letter forms included (*lossless*); and the vocabulary is small, with frequent whole words promoted to single tokens (*compact*). This section states the properties the system guarantees and the components it comprises. The construction of each component is withheld.

Token stream shape. A word decodes from an interpretable sequence of token classes: zero or more prefix tokens, a core that is either a promoted whole word token, a root token, or a root token accompanied by morphological mark tokens, and optional suffix tokens. Every inflected form of a regular root therefore contains the same root token, and the affixes that other tokenizers smear across arbitrary fragments appear as dedicated function tokens. Function morphemes are function tagged: a prefix that renders the same glyph as a bare letter occupies a different id, because the two play different roles and should receive different embeddings.

Vocabulary. The vocabulary is a fixed inventory of 16,384 tokens, frozen with stable ids. It layers special and byte fallback tokens, Hebrew letters including independent final forms, canonical root tokens for triliteral and quadriliteral roots, function tagged prefix and suffix tokens covering Hebrew and Aramaic, a small inventory of reversible morphological mark tokens, and frequent whole word forms stored with exact spelling. The layer budgets, the ordering, and the selection procedure that fills them are not disclosed. Ids fit comfortably under a two byte ceiling, so tokenized corpora are stored as `uint16` and memory mapped for training.

Segmentation. Segmentation is per word, deterministic, and context free: a surface form receives the same analysis wherever it occurs, so a corpus tokenizes identically on every run and tokenization parallelizes trivially. Analyses are chosen by a minimum cost search over a per word morphological lattice; the cost model and the lattice construction are not disclosed. When no morphological analysis reconstructs the surface exactly, a guaranteed fallback covers the word with native Hebrew units, so Hebrew text never degrades to byte fragments.

Reversible morphological marks. A compact inventory of mark tokens lets one canonical root token stand in for the many surface realizations of a weak, hollow, or geminate root. The marks cover the regular deformations described in Section 2: plene spellings that insert a mater lectionis, radicals absent from the surface, gemination and its contraction, the *hitpael* and *itpeel* sibilant metathesis, and the Aramaic voicing and emphatic alternations. Each mark names an honest, literal transformation and is emitted only when its condition actually holds on the surface form; at decode time each mark is inverted exactly. For example, the passive שׁוּלַם (*shulam*, "was paid") is analyzed as the root שלמ together with a recorded mater insertion, and decodes back to its exact spelling. The positional encoding of the marks is not disclosed. Deformations the mark inventory cannot express honestly fall back to a longer but still lossless analysis.

Losslessness. Decoding is the exact inverse of encoding on consonantal Hebrew and Aramaic. Final letter orthography, including scriptural exceptions such as the mid word ם of לְמַרְבֵּה (Isaiah 9:6) and regular forms inside abbreviations, is handled deterministically at decode time. Word boundaries are implicit wherever the grammar permits and explicit otherwise; on the Tanakh, explicit boundary tokens add only about 0.01 tokens per word over the content stream. The binding mark *maqaf* and verse punctuation are preserved reversibly. These guarantees are verified empirically by round tripping entire corpora word by word (Section 6).

Overrides. A curated override layer maps a small set of surface forms, mostly proper names and rare irregulars, to explicit token sequences that expose the gold root where the automatic analysis would not. Every override is validated against the live decoder before admission, so the override layer cannot break losslessness. The machinery that generates and selects overrides is not disclosed.

Freezing, testing, and streaming. Once ids are married to embedding rows, the vocabulary must not be renumbered. The tokenizer is frozen into a single checksummed bundle; rebuilds after freeze are append only. A test suite of 38 checks covers golden token vectors, edge case round trips, vocabulary layout, override integrity, streaming equivalence, a seeded 400 case fuzz pass, and NFC and NFD normalization equivalence; all 38 pass on the frozen artifact. For generation, an incremental streaming decoder withholds a word until its boundary settles, since a word's final letter can flip when its boundary arrives, and emits only stable text.

5 Experimental setup

Corpora. We evaluate on five corpora spanning the registers of classical Hebrew and Aramaic: the Tanakh (Hebrew Bible), the Mishnah, Mishneh Torah (Maimonides' halakhic code), Midrash Rabbah, and the tractate Chagigah of the Babylonian Talmud. Chagigah is held out entirely from vocabulary construction and tuning, so its numbers measure generalization to unseen text rather than fit to a known corpus. All text is normalized to consonantal form (niqqud and cantillation stripped), which is how Hebrew and Aramaic are normally written.

Metrics. *Fertility* is the mean number of tokens per word, lower being better. We report it on two bases, kept distinct throughout: the *morphological* basis counts tokens per morphological word over the OSHB gold word list weighted by corpus frequency, and the *running text* basis counts tokens per whitespace word on the raw normalized corpus, the basis on which all tokenizers are compared head to head. *Root preservation* is the share of gold-root-bearing words whose gold root is recovered as an intact token (a word that is a single whole word token is credited, since it is one semantic unit; otherwise the gold root token must appear in the stream). *Round trip* reports whether decoding reproduces the input exactly.

Baselines. The baselines are the tokenizers actually applied to Hebrew in practice: the tiktoken byte level BPE encodings cl100k (earlier GPT-4) and o200k (current ChatGPT models), and a byte fallback BPE and a unigram model trained with SentencePiece on Hebrew at the same 16,384 token budget. Every tokenizer is measured on the same normalized text with the same whitespace word denominator.

6 Results

6.1 Fertility and round trip across registers

Table 1 reports running text fertility and round trip across the five corpora. Fertility rises smoothly from biblical through halakhic to talmudic text, tracking lexical diversity, and round trip is exact everywhere. The held out tractate Chagigah sits at 1.759, close to the in-distribution registers, so the behavior generalizes rather than fitting a memorized corpus.

6.2 Root preservation

On the Tanakh, scored against OSHB gold morphological lemmas, Tokemon preserves **98.07%** of roots as intact units. Of the root scored words, **80.6%** are a single whole word token (the word is one semantic

Corpus	Words	Fertility	Round trip
Tanakh (canonical text)	266,100	1.757	lossless
Mishneh Torah (Halakhah)	785,706	1.523	lossless
Midrash Rabbah	539,708	1.653	lossless
Mishnah	187,830	1.633	lossless
Chagigah, Talmud Bavli (held out)	18,635	1.759	lossless

Table 1: Running text fertility (tokens per whitespace word) and round trip fidelity across registers, measured on normalized consonantal text with the frozen bundle. On the morphological basis (tokens per OSHB word, frequency weighted) the Tanakh figure is 1.297.

unit), **18.6%** are multi token with the gold root token present in the stream, and only **0.8%** fall back without exposing the root. The morphological fertility that this preservation is achieved at is 1.297 tokens per word.

Root preservation and fertility pull against each other: forcing a gold root into a word that the minimum cost analysis would otherwise spell out costs extra tokens. The shipped configuration holds morphological fertility at 1.297; internal measurements show that preservation toward 98.4% is reachable at a fertility of 1.298, a trade the release declines in favor of the tighter encoding.

6.3 Aramaic

The same tokenizer, unchanged, handles Aramaic. Table 2 reports root preservation on two gold sets: a hand tagged Babylonian Talmudic set and the full Biblical Aramaic of Daniel and Ezra extracted from OSHB. On the Talmudic set, preservation is 99.85% by token mass (the set covers about 63% of the Bavli by mass), and per type it is 95.03%; on Biblical Aramaic it is 96.38% by mass. Round trip is exact on both, with zero failures. The two sets weight the credit differently: on the frequency dominated Talmudic set, 72% of scored types are credited through a single whole word token and 25% through an explicit root token, whereas on the type diverse Biblical set 72% carry an explicit root token. Misses concentrate in Persian loanwords and proper names (for example אֲחַשְׁדַּרְפְּנַיָּא, אַרְתַּחְשַׁשְׁתְּא) and in prefixed forms whose minimum cost cut crosses the root.

Gold set	Scored types	By mass	Per type	RT fails
Talmudic Aramaic (hand tagged)	865	99.85%	95.03%	0
Biblical Aramaic (Daniel, Ezra)	1,609	96.38%	94.72%	0

Table 2: Aramaic root preservation, scored with the same matcher used for Hebrew. Hebrew and Aramaic are interleaved within single Talmudic sentences and one tokenizer covers both.

6.4 Head to head comparison

Table 3 compares all five tokenizers on the same normalized running text and the same whitespace word denominator. The picture is honest and consistent. Against the production tokenizers that would otherwise be used on Hebrew, Tokemon is decisively tighter: 1.3 to 1.6 times tighter than o200k and about three times tighter than cl100k across registers, while also keeping the root visible and round tripping losslessly, neither of which the tiktoken encodings do. Against a subword tokenizer trained on Hebrew at the same budget, Tokemon is competitive but not the tightest: it essentially ties the Hebrew BPE on the Tanakh and beats the Hebrew unigram there, while on lexically diverse later text the Hebrew BPE compresses about 12 to

35% tighter, because it spends its whole budget on frequency optimal fragments whereas Tokemon reserves part of the vocabulary for grammar units and pays a token per punctuation mark. That gap is the measured price of the semantic and lossless constraints, and we consider it well spent: the compression only baseline exposes no roots and does not round trip exactly on every corpus.

Corpus	Tokemon	BPE 16k	Unigram 16k	cl100k	o200k
Tanakh (canonical)	**1.757**	1.748	1.808	6.013	2.763
Chagigah (held out)	1.759	**1.553**	1.580	5.153	2.357
Mishnah	1.633	**1.485**	1.510	5.097	2.218
Mishneh Torah	1.523	**1.322**	1.354	4.854	2.011
Root visible	yes	no	no	no	no
Lossless round trip	yes	partial	partial	yes	yes

Table 3: Tokens per word on normalized consonantal running text, same word basis for all five. cl100k and o200k are the tiktoken encodings behind ChatGPT; the Hebrew BPE and Hebrew unigram are trained on the same corpus at the same 16,384 budget. Tokemon is the only tokenizer that keeps the root as a token and round trips losslessly across every register (the SentencePiece models reconstructed most but not all of Mishneh Torah under a strict character level check).

7 Throughput and corpus scale ingestion

Tokemon runs at corpus scale. Warm throughput is about 2.9 MB/s, comparable to SentencePiece. With eight worker processes the parallel emitter tokenizes the full Talmud Bavli in 6.4 seconds, and the full Sefaria corpus of about 286 million words (6,205 texts, 515.8 million tokens) in roughly 17 minutes. Tokens are written as `uint16`, two bytes each, so the output is compact and memory mappable for training, well under the 65,536 id ceiling. Tokenization is deterministic and context free, so the corpus tokenizes identically on every run, and the full 286 million word corpus round trips with **zero** failures across all 6,205 texts, leaving no silent corruption to reconcile downstream.

8 Design space: instrumented alternatives

The shipped tokenizer is the survivor of an explicit search over the design space, and two discarded branches are informative enough to report.

Gematria token identities. An earlier line encoded each word as the numerical value (*gematria*) of its prefix, root, and suffix, so that inflections of one root would share a root value. Across five iterations on Genesis, this reached 100% root consistency and a fertility of 1.506 to 1.517 tokens per word with a verified bijection between surface forms and token sequences, using order sensitive gematria variants to break collisions. It was abandoned for two reasons: a Hebrew trained BPE at vocabulary 4,000 compressed Genesis tighter (1.304), so gematria did not win on fertility; and a numerical value is opaque, discarding the very morpheme structure the current design keeps explicit.

Factored embeddings. A further variant pushed all structure out of the token stream into parallel embedding channels, giving a fertility of exactly 1.0. At full Sefaria scale (449,692 unique Hebrew and Aramaic forms) the channels were unique for all but ten anagram pairs, a 99.996% uniqueness. It was abandoned

because collapsing a word to one token removes the morpheme boundaries from the sequence the model attends over, the opposite of the goal.

These branches motivate the shipped choice: native morpheme tokens are collision free by construction (two roots never share a token, whatever their gematria), keep the boundaries in the stream, and stay lossless, at a fertility close to the gematria variants and competitive with BPE.

9 Limitations

Fertility on later, more lexically diverse registers is higher than on the Tanakh, as Table 1 shows; this is expected, since richer vocabulary admits fewer high frequency whole word forms. Root preservation is reported against OSHB gold, which is available for the Tanakh and for Biblical Aramaic and hand extended to a Talmudic sample; comparable gold annotation is scarcer for later corpora, and the Talmudic long tail below the sampled frequency stratum is unmeasured. The override layer covers attested forms and does not generalize to unseen surface forms.

Tokemon targets classical Hebrew and Aramaic. On text that leans on borrowed modern terminology, foreign words written in Hebrew script that have no Semitic root to recover, a general subword tokenizer with a large multilingual vocabulary can compress more tightly, without the root level structure Tokemon keeps; dedicated loanword handling is left to future work. Finally, the larger experiment this design enables, a controlled comparison of a morphological against a standard BPE tokenizer under an identical language model at equal epochs, is specified but not yet run, and is not claimed here.

10 Conclusion

Tokemon is a root aware, lossless morphological tokenizer for Hebrew and Aramaic. On the Tanakh it runs at 1.30 tokens per morphological word, preserves 98.07% of roots as intact tokens, and round trips exactly, with zero round trip failures verified across roughly 1.8 million words and, at corpus scale, across 286 million words. Against the tokenizers that would otherwise process Hebrew it is 1.3 to 1.6 times tighter than o200k and about three times tighter than cl100k, while a compression only subword tokenizer can save a further 12 to 35% on lexically diverse text, but without the root or the lossless guarantee. For pipelines working in Hebrew and Aramaic, Tokemon keeps the token count low, the round trip exact, and the root visible in the token stream, which is the property most likely to help downstream modelling. The same architecture extends along the Semitic line, with Arabic and Judeo-Arabic as natural next targets.

References

[1] Philip Gage. 1994. A New Algorithm for Data Compression. *The C Users Journal*, 12(2).

[2] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of ACL*.

[3] Mike Schuster and Kaisuke Nakajima. 2012. Japanese and Korean Voice Search. In *Proceedings of ICASSP*.

[4] Taku Kudo. 2018. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. In *Proceedings of ACL*.

[5] Taku Kudo and John Richardson. 2018. SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing. In *Proceedings of EMNLP: System Demonstrations*.

[6] Kaj Bostrom and Greg Durrett. 2020. Byte Pair Encoding is Suboptimal for Language Model Pretraining. In *Findings of EMNLP*.

[7] Stav Klein and Reut Tsarfaty. 2020. Getting the ##life out of living: How Adequate Are Word-Pieces for Modelling Complex Morphology? In *Proceedings of the 17th SIGMORPHON Workshop*.

[8] Valentin Hofmann, Janet Pierrehumbert, and Hinrich Schütze. 2021. Superbizarre Is Not Superb: Derivational Morphology Improves BPE Subword Segmentation. In *Proceedings of ACL*.

[9] Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. How Good is Your Tokenizer? On the Monolingual Performance of Multilingual Language Models. In *Proceedings of ACL*.

[10] Aleksandar Petrov, Emanuele La Malfa, Philip H. S. Torr, and Adel Bibi. 2023. Language Model Tokenizers Introduce Unfairness Between Languages. In *Advances in Neural Information Processing Systems 36*.

[11] Aaditya K. Singh and DJ Strouse. 2024. Tokenization Counts: The Impact of Tokenization on Arithmetic in Frontier LLMs. *arXiv:2402.14903*.

[12] Reut Tsarfaty, Amit Seker, Shoval Sadde, and Stav Klein. 2019. What's Wrong with Hebrew NLP? And How to Make it Right. In *Proceedings of EMNLP-IJCNLP: System Demonstrations*.

[13] Amir More, Amit Seker, Victoria Basmova, and Reut Tsarfaty. 2019. Joint Transition-Based Models for Morpho-Syntactic Parsing: Parsing Strategies for MRLs and a Case Study from Modern Hebrew. *Transactions of the ACL*, 7.

[14] Amit Seker, Elron Bandel, Dan Bareket, Idan Brusilovsky, Refael Shaked Greenfeld, and Reut Tsarfaty. 2022. AlephBERT: Language Model Pre-training and Evaluation from Sub-Word to Sentence Level. In *Proceedings of ACL*.

[15] Avi Shmidman, Joshua Guedalia, Shaltiel Shmidman, Cheyn Shmuel Shmidman, Eli Handel, and Moshe Koppel. 2022. Introducing BEREL: BERT Embeddings for Rabbinic-Encoded Language. *arXiv:2208.01875*.

[16] Nizar Y. Habash. 2010. *Introduction to Arabic Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool.