Automatic Text Segmentation of Ancient and Historic Hebrew

Elisha Rosensweig[1,‡], **Benjamin Resnick**[1,‡], **Hillel Gershuni**[1,2,‡],
Joshua Guedalia[1,‡], **Nachum Dershowitz**[3,§], **Avi Shmidman**[1,2,†]
[1]DICTA / Jerusalem, Israel [2]Bar Ilan University / Ramat Gan, Israel
[3]Tel Aviv University / Tel Aviv, Israel
[†]avi.shmidman@biu.ac.il
[‡]{benshafat,benjaminmresnick,gershuni,joshuaguedalia}@gmail.com
[§]nachum@tau.ac.il

Abstract

Ancient texts often lack punctuation marks, making it challenging to determine sentence boundaries and clause boundaries. Texts may contain sequences of hundreds of words without any period or indication of a full stop. Determining such boundaries is a crucial step in various NLP pipelines, especially regarding language models such as BERT that have context window constraints and regarding machine translation models which may become far less accurate when fed too much text at a time. In this paper, we consider several novel approaches to automatic segmentation of unpunctuated ancient texts into grammatically complete or semi-complete units. Our work here focuses on ancient and historical Hebrew and Aramaic texts, but the tools developed can be applied equally to similar languages. We explore several approaches to addressing this task: masked language models (MLM) to predict the next token; few-shot completions via an open-source foundational LLM; and the "Segment-Any-Text" (SaT) tool by Frohmann et al. (Frohmann et al., 2024). These are then compared to instruct-based flows using commercial (closed, managed) LLMs, to be used as a benchmark. To evaluate these approaches, we also introduce a new ground truth (GT) dataset of manually segmented texts. We explore the performance of our different approaches on this dataset. We release both our segmentation tools and the dataset to support further research into computational processing and analysis of ancient texts, which can be found here https://github.com/ERC-Midrash/rabbinic_chunker.

1 Introduction

Ancient languages lack many of the classic features that modern languages use to clarify and disambiguate how to read them. These include spaces between words, diacritics, punctuation and more. This makes it challenging to determine sentence and clause boundaries. Determining these boundaries is a crucial step in any attempt to decipher, analyze and process ancient texts. In the past this would be needed simply as a way to enable humans to read these texts, while in the modern era this need has expanded as more and more NLP tools are coming out, some of which require such sentence segmentation as a prerequisite.

The issue of segmenting text into smaller chunks is a well-known challenge, with a wide range of use-cases and applications. Humans are many times the direct beneficiaries of such a segmentation, in the form of subtitles (Alvarez et al., 2017; Ponce et al., 2023), Easy Read text (Calleja et al., 2024), or text summaries created on a per-segment basis (Cho et al., 2022; Aumiller et al., 2021; Hazem et al., 2020). Furthermore, in recent years we see more and more NLP tools that, while powerful, are limited in the size of text they can accept as input. Accordingly, for each of these tools there arises a need to segment a large text into smaller chunks. These include (but are not limited to) BERT models (Gong et al., 2020) and LLM context windows (Shi et al., 2024).

Although segmenting a text does not change the text itself, the segmentation strategy one selects will impact the quality of downstream tasks that take the segmented text as input. Possible negative impacts include: cutting sentences in half; reduced readability (e.g., in the case of subtitles); loss of information and critical context (e.g. in the case of LLM context and translation), etc. For this reason, different domains and segmentation challenges also require different policies. A sentence segmentation tool, such as discussed in Frohmann et al. (2024), will not be sufficient for handling the needs of chunking large texts for feeding BERTs, where punctuation is assumed to be stable but topic consistency and coherence is a concern. Different tools might be needed for the same task but with changes in the source text properties, such as lan-

Proceedings of the Second Ancient Language Processing Workshop associated with NAACL 2025, pages 1–11
May 4, 2025.

guage and genre (Homburg and Chiarcos, 2016; Aumiller et al., 2021; Hazem et al., 2020).

In this paper, we explore several approaches to the segmentation of ancient and historic Hebrew, as a necessary prerequisite for effective training and running of fine-tuned BERT models for punctuation, morphological tagging, syntactic parsing, and more. The segmented chunks are critical both for creating training samples with which to fine-tune BERT models for the aforementioned tasks, and also at inference time, to ensure accurate results from the models.

As such, our goal is to minimize damage at the sentence level while bounding the length of these segments, which we will refer to as "chunks". A chunk can sometimes be composed of several sentences or, conversely, it may be a syntactically independent part of a single sentence. We are not aware of any work done trying to address this specific challenge w.r.t. ancient and historic Hebrew, and thus draw upon related work from other (similar) domains instead.

Another deficiency in this domain is a lack of ground truth (GT) datasets by which to evaluate these approaches. To this end, our work here also provides the first GT dataset of manually segmented texts for ancient and historical Hebrew, as far as we are aware. We explore the performance of our different approaches on this dataset. We release both our segmentation tools and the dataset to support further research into computational processing and analysis of ancient texts.

The structure of this paper is as follows. In Section 2 we discuss related work, followed by Section 3 where we discuss our GT test set and our approach to curating it, and Section 4 that outlines the metrics we used in this paper. With the background out of the way, we progress to Section 5 to review the various segmentation tools we put to use in this paper. Section 6 presents our results, demonstrating the effectiveness of each of the tools we explore here. We conclude with Section 7 where we discuss the takeaways from this work.

2 Related Work

Several approaches have been used in the past for sentence-level segmentation. One approach is to segment the text using masked language models (e.g., BERT models) to predict punctuation marks. This has been done in various contexts. The basic idea is to use the punctuation marks as natural locations in which to segment the text. These ideas worked well for segmenting Easy Read texts (Calleja et al., 2024) and subtitle generation (Ponce et al., 2023), yet it is not clear a-priori that they would work well for other use-cases. Specifically, it is unclear how the various BERT models would behave when facing ancient and historic Hebrew. Some Hebrew models, which we tested here, were trained only on more modern texts, while others were trained on texts where the punctuation is inconsistent and unreliable. Thus, the degree to which this approach can be useful is an open question, which we explore here.

Then there are the approaches that use Generative AI, specifically instruct and few-shot flows. In both of these, the flow prompts the GenAI tool to reproduce the original text with added markers that indicate where to segment it. These are state-of-the-art approaches and very commonly used in the literature for many tasks, well beyond segmentation. One challenge which these approaches have had in the past is that, when asked to reproduce the original text, the GenAI tool does not return a perfect reproduction of the text. Instead, it adds, subtracts or rephrases some of the original text. To manage this issue at scale researchers have proposed auxiliary scoring methods that provide an approximate evaluation of the segmentation quality (Calleja et al., 2024). This issue was also experienced by us in this work. We make note of how it impacted the overall viability of such approaches in Section 6.

In addition to all the above, it is important to note that ancient and historic Hebrew do not always conform to current grammar rules and conventions. Specifically, sentences can formally come out to be hundreds of words long, making a perfect segmentation an impossible goal at times. See more on this in Section 3, where we discuss how this impacts the construction of a GT dataset.

Finally, it is worth noting here how our work relates to existing notation works in ancient Hebrew, specifically the ETCBC project (Eep Talstra Centre for Bible and Computer, 2023). The ETCBC project provides linguistically annotated texts of the Hebrew Bible, offering researchers a comprehensive database with morphological, syntactic, and semantic features encoded in a hierarchical text model that facilitates computational analysis of biblical Hebrew texts. Our work here tackles Rabbinic texts which have a different set of challenges than that of the Bible. Whereas the bible is already pre-

versified into relatively short verses, the texts we tackle are much longer, and can reach hundreds of words. We hope that our work here will enable us in the future to use auto-segmented ancient texts and move to the next level of developing features along the same line as done in ETCBC.

3 Methodology - Ground Truth Dataset

3.1 Approach

The idea of a ground truth for text segmentation is a fuzzy concept, for several major and distinct reasons.

First, there can be a clash between the natural dynamics of the language and our segmentation goals. On the one hand, for segmentation to be useful, each segment must be capped in length. On the other hand, language in general, and ancient languages in particular, do not have a theoretical bound on sentence length. In Rabbinic texts, for example, it is very common to embed lengthy quotes within a sentence, such that any break between what comes before the quote and what comes after is detrimental to the grammatical structure of the sentence. As such, it is not always possible to segment the text in a manner that both meets the hard length constraints required and simultaneously does no "harm" to the sentence. This, in turn, complicates the process of generating a GT.

Second, to add to the previous point, ancient languages often do not abide by a clear set of grammar rules. As such, it is not always clear where the correct place is to segment the texts, even in cases where length constraints are not an issue.

In light of these two challenges, what is needed is a gradient on which to scale the quality of a segmentation technique. Specifically, we define three levels of segmentation markers:

- Break (B) - Positions that are clear segmentation points. These correspond roughly to ends of sentences, marked in modern Hebrew by a period, a question mark or exclamation point.
- Partial (P) - Positions which reflect a natural pause in the sentence or a completion of an idea. These correspond roughly to semicolons, colons before a list or a lengthy quote, etc.
- Maybe (M) - Positions which should not be used for segmentation in general, but are clearly superior segmentation positions than one word prior or subsequent to them. Many times these can correspond to where a comma would be placed, but not limited to such cases.

For example, taking a statement from the 3nd century Hebrew treatise Mishnah, Tractate Avot, Chapter 1, Unit 2:

שמעון הצדיק היה משירי כנסת הגדולה הוא היה אומר על שלשה דברים העולם עומד על התורה ועל העבודה ועל גמילות חסדים ("Simeon the Just was one of the last men of the Great Assembly. He used to say: the world stands upon three things: the Torah, the Temple service, and the practice of acts of piety.")

One (possible, legitimate) segmentation would be as follows:

שמעון הצדיק [M] (Simeon the Just)

היה משירי כנסת הגדולה [B] (was one of the last men of the Great Assembly.)

הוא היה אומר [P] (He used to say:)

על שלשה דברים ם העולם עומד [P] (the world stands upon three things:)

על התורה [M] (the Torah)

ועל העבודה [M] (the Temple service)

ועל גמילות חסדים [B] (and the practice of acts of piety.)

Note, especially, the usage of "M" markups. An ideal text segmentation would not segment the text at these positions, as they break up the sentence and "destroy meaning" in the process. However, it is clearly worse to break up the sentences in a position one word earlier or later. Therefore, we consider a segmentation tool which would segment at these positions as less than ideal, but which is picking up on language semantics and thus better than a random segmentation tool.

3.2 Text Selection

For the work we did in this paper, we completed the annotation of 25 texts, each 200–400 words in length. Table 1 presents the breakdown into different periods, as well as some high-level statistics on the number of words and different markers in our dataset. The core dataset consists of over 4000 words of Hebrew/Aramaic sources from the period of the "Geonim", dated 8th-10th century. To this we add an additional 3177 words of "Rishonim" texts from the High Middle Ages, in order to test whether the extent to which our results are valid for later medieval Hebrew texts as well. This dataset was annotated by a single annotater, with ten years of study in a rabbinic seminary and highly skilled

in parsing rabbinic texts. While this is a small test-set and with only a single annotator, it is the first of its kind to the best of our knowledge. In the future, we plan on expanding the coverage of the dataset, as well as gathering annotations from additional annotators to allow for evaluation of interannotator agreement. The latest version of the ground truth dataset can be found here https://github.com/ERC-Midrash/rabbinic_chunker.

	Geonim	Rishonim	Total
# Texts	11	14	25
# Words	4088	3177	7265
#B	109	132	241
#P	198	178	376
#M	1086	775	1861

Table 1: Ground Truth Test Set - Breakdown

4 Analysis Metrics

4.1 Approach

In order to analyze the performance of a given text segmentation tool, when comparing it with a GT segmentation, we would naturally want both high precision and high recall. Segmentation with high precision would imply we segment only where appropriate, and high recall would imply we capture most meaningful segmentation positions. An important thing to note here is that high recall, after a certain stage, provides diminishing returns, since the downstream NLP tasks that the segmentation will support are met once we do not exceed some upper length limit of segment length. Thus, in this paper we focus on precision, constrained by the requirement that the produced chunks that are reasonably sized. In our experience, having run fine-tuned Hebrew BERT models across many ancient Hebrew texts, we have found that input chunks of up to 50 words work far better than longer chunks. After the 50-word point, accuracy starts to drop precipitously. Thus, our goal is an upper bound of ≈ 50 words per chunk.

When measuring performance, the question arises: which segmentation markings in the GT should we consider for purposes of evaluation? The reasonable options, using the markup scheme described in Section 3.1, are $\{B\}$, $\{B, P\}$, or all three $\{B, P, M\}$. These represent progressively more permissive/flexible segmentation of the same text. Naturally, the performance scores of any tool will be directly impacted by this decision, with precision monotonically growing and recall decreasing as we move from strict to permissive segmentation. We discuss this impact in the analysis section below (Section 6).

4.2 Notation

Let $\mathcal{L}$ be the set of all layer combos we are interested in evaluating. As just discussed, $\mathcal{L} = \{B, \{B, P\}, \{B, P, M\}\}$. For any $l \in \mathcal{L}$ we define $Precision_l^\alpha$ to be the precision of segmentation algorithm α over the GT when considering only segmentation markers in l. The same goes for $Recall_l^\alpha$.

5 Automatic Text Segmentation Tools

Our research explores several distinct approaches to automatic segmentation of unpunctuated (ancient) texts, each leveraging capabilities of different language models and neural architectures. Each of these approaches offers different advantages in terms of accuracy, computational requirements, and generalizability across different types of texts[1]. Our implementation of the tools described here are publicly available in our repohttps://github.com/ERC-Midrash/rabbinic_chunker.

5.1 Few-Shot Learning with DictaLM2.0

The first approach uses DictaLM2.0, an open-source LLM specifically trained on Hebrew texts (Shmidman et al., 2024a). We use the base model, rather than the instruct-tuned model, in order to leverage the full raw strength of the model. Thus, in order to elicit desired output from the model, we implement a few-shot learning protocol (Brown et al., 2020). The model is presented with several examples of input texts and correctly segmented versions of these texts. This is then followed by the target text requiring segmentation, after which the model proceeds with a completion. This method takes advantage of DictaLM2.0's specialized knowledge of Hebrew language patterns, while requiring minimal fine-tuning or additional training.

In this paper we provided the LLM four examples (shots). Each "shot" was comprised of a JSON object with two entries - *"raw"* to reflect

[1]Our approach in this paper was to explore the type of segmentation information already embedded among various existing models. An alternative approach would be to train models from scratch aimed at chunking, which would enable us to take a variety of approaches, e.g., hierarchical chunking. As explained, this is outside the scope of this work.

the input text, and *"chunked"* to reflect the expected segmented text, which was the same text but with double-slash ("//") markings as indication for where we would expect a segmentation point. See Appendix A.1 for the full specification of the shots used.

5.2 Next-Token Prediction using Masked Language Models (MLMs)

Our second approach follows a similar structure to other works (e.g., Calleja et al. (2024)), leveraging the masked language modeling (MLM) head of pre-trained BERT models. Given a long text, we use a sliding window fitted to the model context window. Within that window, taking each word in turn, we mask the word, and predict the subsequent token. When delimiter tokens (periods and/or colons, depending on the configuration we test) appear among the top predictions, we mark these positions as potential segment boundaries. Once a delimiter appears in the top K (K = 5, 15 for different runs) options, we select the earliest point to cut and move the window. This continues until the text is fully processed.

In this paper, we use the following BERT models:

- HeBERT (Chriqui and Yahav, 2021) - The first dedicated Hebrew BERT model, trained with a 30K-token vocabulary
- AlephBERT (Seker et al., 2021) - An expanded dedicated Hebrew BERT model, trained with a 52K-token vocabulary, and a much larger corpus.
- DictaBERT (Shmidman et al., 2023) - Currently the highest-performing BERT for modern Hebrew (Shmidman et al., 2024b), trained with a 128K-token vocabulary.
- BEREL (Shmidman et al., 2022) - A BERT model specifically trained for Historic Hebrew/Aramaic texts (128K-token vocabulary)[2].

5.3 Segment Any Text (SaT)

Frohmann et al. (2024) trained models with the express goal of providing a "universal approach for robust, efficient and adaptable sentence segmentation", referred to as *SaT*. They specifically aim to improve over the previous tool, *WtP ("Where's the Point")*, which was presented in (Minixhofer et al., 2023). Improvements include handling of short sentences and code-switching, as well as speeding up the model by moving from character-based to token-based processing.

The authors provided a working github project containing their models and code for running their tool (segment-any text, 2025). For our experiments here, we selected their *sat-12l-sm* model, a 12-layer multilingual model which the authors report had the best multilingual performance (96.0 macro-average F1 score, as reported at the time of writing this paper).

5.4 Benchmark: Instruction-Based Segmentation using Closed LLMs

All the methods mentioned thus far are open and free for use. Our final approach utilizes state-of-the-art closed-source language models (GPT-4o and Claude Sonnet) in an instruct flow. One of the main challenges in instruct-based flows is Prompt Engineering (PE), which is notoriously brittle (Errica et al., 2024). However, in our case we utilize the likely fact that these models have seen plenty of punctuated texts during training and are familiar with punctuation tasks. Our approach is therefore a 2-step flow: (a) prompt the LLM to punctuate a given unpunctuated text (see Appendix A.2 for prompt details) and then (b) segment the punctuated text at major punctuation marks (periods, colons, semicolons, and question marks).

Using these commercial models is helpful as an evaluation tool. However, we do not investigate these models in depth, for two reasons. First, these require access to paid services, and also lack transparency (e.g., model weights, open to fine-tuning etc.), thus making them less appropriate for broader use in the research community. Second, as we shall discuss in the analysis section, they have properties that make them unstable and less suitable to building reliable segmentation flows.

6 Performance Analysis

6.1 The Trouble with Generative LLMs

Generative models are known to be difficult to control via prompts when very precise output is required. In the case of segmentation, we find (as others have commented as well) that using gen-

[2]Note that while BEREL might have seen the GT texts in its training data, the texts BEREL was trained on were almost exclusively without punctuation marks. Thus, it will not have had prior hints to the correct segmentation of the GT dataset.

erative models is fraught with instability, as the LLMs at times inject or delete parts of the text they are asked to segment. While these issues can be handled with further work (modified instructions, modified examples in few-shot, guided decoding, and additional techniques), they are for the most part heuristic improvements that cannot be ensured with total certainty. Table 2 lists the number of texts, out of 25, for which the flow did not add or remove any text. All performance analysis in the rest of Section 6 relates only to this subset of texts.

Genre	Count
Total	25
AlephBERT	25
heBERT	25
Dictabert	25
BEREL	25
DictaLM2.0	17
SaT-12L-sm	25
gpt4o	22
claude-sonnet-3.5 v1 (20240620)	20
claude-sonnet-3.5 v2 (20241022)	12

Table 2: List of segmentation tools and the number of GT texts that they segment correctly, i.e., w/o adding/removing text to the input text. Generative models demonstrate instability in this flow, making them less suitable to be used as part of a streamlined flow.

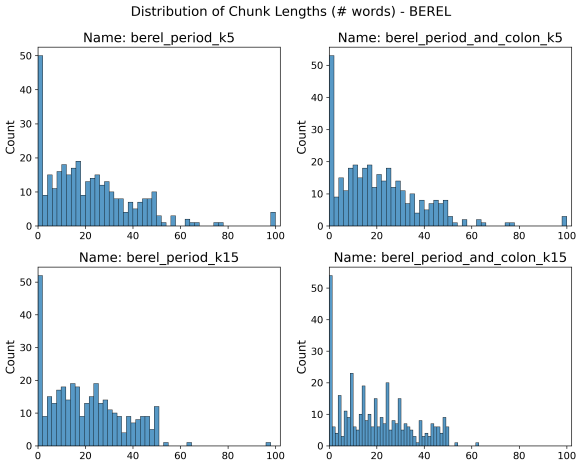


Figure 1

6.2 Segmentation Evaluation: Chunk Size

In our problem formulation, we want to have segmentation occur at reasonable places while ensuring that segments are not too long. Let us begin by reviewing the distribution of segment lengths as output by the different tools.

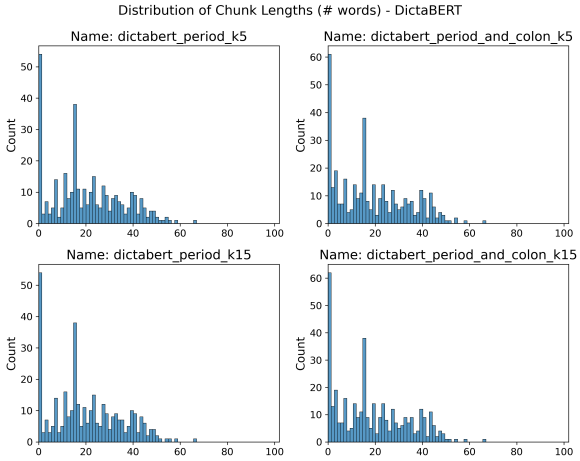


Figure 2

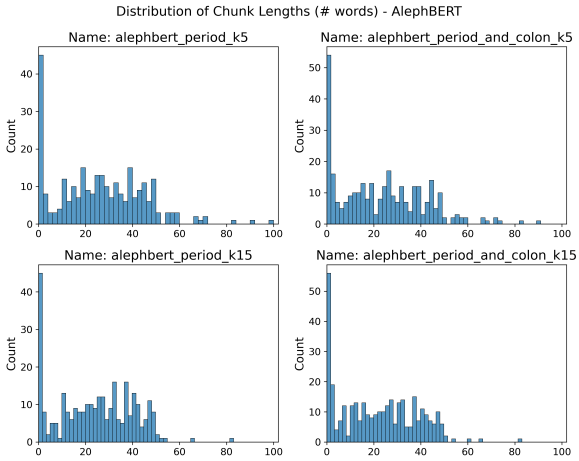


Figure 3

Fig. 1-4 show the performance of BEREL, DictaBERT, AlephBERT and HeBERT respectively, w.r.t. meeting chunk length constraints. In these, we collect all the chunks from across all 25 test samples and plot chunk length histograms. As we can see, all BERT-based models maintain our upper-limit of 50 words, with a small number of outliers. Going deeper in the search for candidate delimiters (K=15) seems to reduce those outliers as well. Note that this is not a straightforward result, as earlier segmentation choices could, at times, cause later segmentation opportunities to be fewer, thus lengthening future chunks.

Finally, we compare this performance to that of the Generative LLMs and SaT (Fig. 6). We can see that they too abide by the limits we were aiming for, though it is clear that the LLMs are concentrating the segment lengths on the shorter side. This might be due to the fact that they start by fully punctuating the text using modern standards, which could result

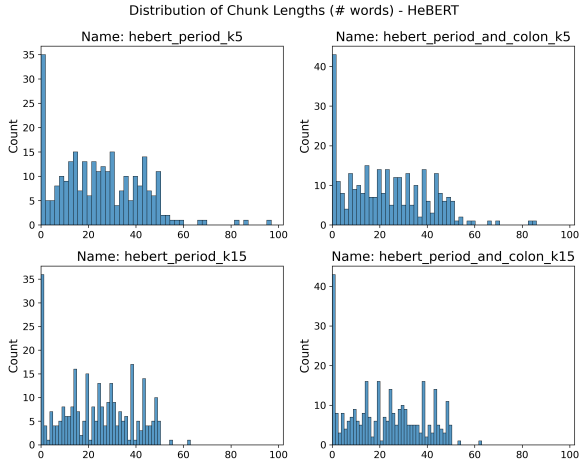


Figure 4

in shorter sentences and therefore shorter chunks.

6.3 Segmentation Evaluation: Precision

Satisfied that our segmentation tools are bound by length as required, let us now turn and check how good they are at finding good positions to segment the text. We focus first on the performance regarding Geonic texts of the first millennium and then consider whether to what degree this performance is maintained when the same methods are run slightly later medieval Hebrew texts of the Rishonim period.

Figure 5 shows how precision of segmentation behaves for Geonic texts, as a function of tool and segmentation strictness in GT. Moving from left to right, we see the results for the closed-model LLMs, then the few-shot flow over DictaLM2.0. After that we have *SaT-12L-sm*, and then finally we review our three Hebrew BERT models. For each of the three BERT models we present here, there are four results, corresponding to the four variations of using them: whether we use periods or also colons to predict a segmentation point, and how deep in the options-stack we look in search of these delimiters ($K = 5$ or 15). Figure 7 shows the same plot, but this time for the entire dataset, combining Genoim and Rishonim.

General Trends. As expected, as we move from top to bottom and restrict segmentation to more obvious markers, we see a decrease in precision. From a birds-eye view, we can see that most models have similar median performance, and experience the same trends as we move from top to bottom. Specifically, it seems from this that the open models compete well with the closed-model LLMs, at least for the current instruct prompts we used here. Note also that, as mentioned in Section 6.1 and shown in Table 2, the generative models results are not for the full dataset, but rather limited to those where the model did not modify or corrupt the text.

Best MLM. The best performance among the MLM solutions was seen with BEREL, the BERT model trained on Historical Hebrew. Especially for the most relaxed mode (BPM) it has near-perfect precision, and outperforms all other solutions. (Note once more that *claude-sonnet-v2*, which also seems to do well for BPM, is scored on only half of the texts, which makes it difficult to draw conclusions from that performance).

Performance of SaT. The model provided by Frohmann et al. (2024) seems to do as well and even better than all MLMs, with the exception of BEREL. This holds for all three marker-selection options. As this is a universal model, compared to the other tools which were all trained specifically on some variation of Hebrew language, this is quite impressive and satisfying, considering that the competition has an "unfair advantage".

The above points hold both when limiting our view to the earlier Geonic texts, as well as when expanding our view to include the Rishonim. This is encouraging, as it means our method will serve us well not only for ancient Hebrew, but for broader sections of historic Hebrew as well.

7 Conclusion

In conclusion, we have identified a practical and high-performing method of segmenting historic Hebrew texts, using the MLM-based method with the BEREL BERT model for historic Hebrew. We have shown that it far outperforms SaT, and that its performance rivals that of the LLMs, without the instability of the LLMs, and without having to rely on the commercial/closed nature of the big LLMs. Furthermore, we release our code so that this method can be easily run by the NLP community any other Hebrew text, plugging in any desired Hebrew BERT model as desired. Finally, we release the test dataset, the first of its kind, so that future segmentation models developed by the NLP community can be evaluated and compared to the benchmarks that we report here.

Acknowledgments

This work has been funded by the European Union (ERC, MiDRASH, Project No. 101071829; Principal investigators: Nachum Dershowitz, Tel-Aviv

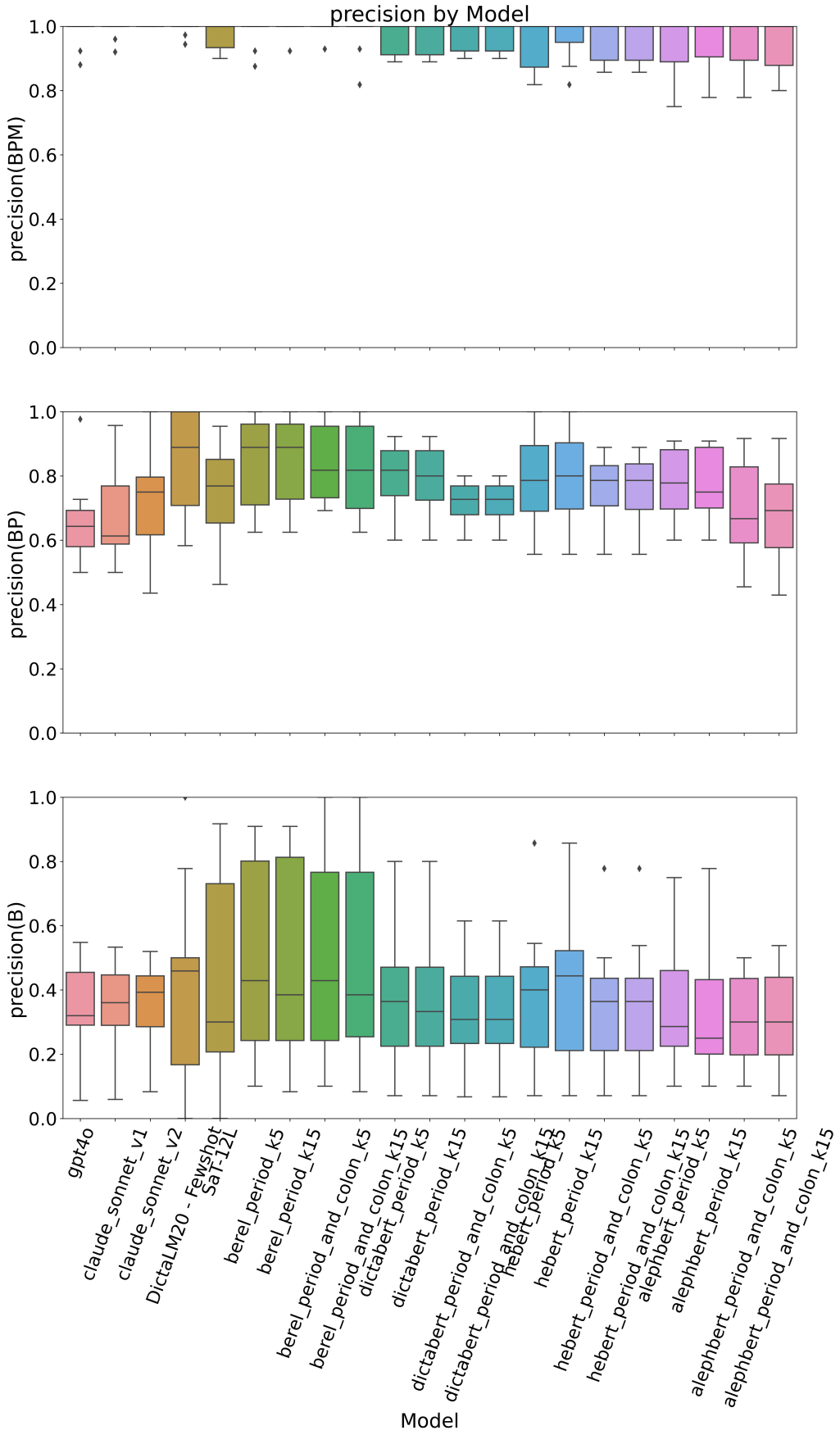


Figure 5: Precision of each method, as dependent on the GT labels we use, for the 8th-10th century Geonic texts. The plots show how precision scores change as we move from more relaxed segmentation demands $l = \{B, P, M\}$ (top), to stricter ones where $l = \{B\}$ (bottom).

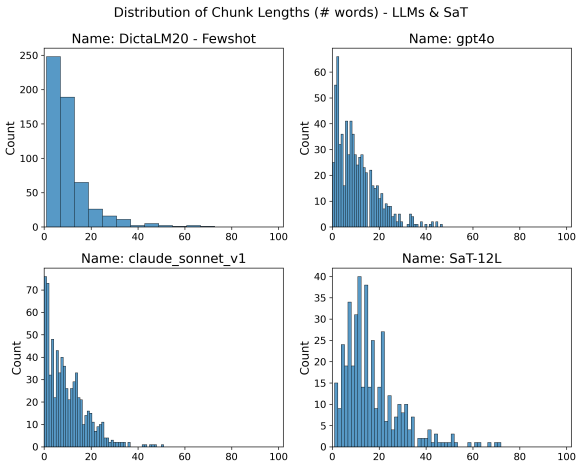


Figure 6

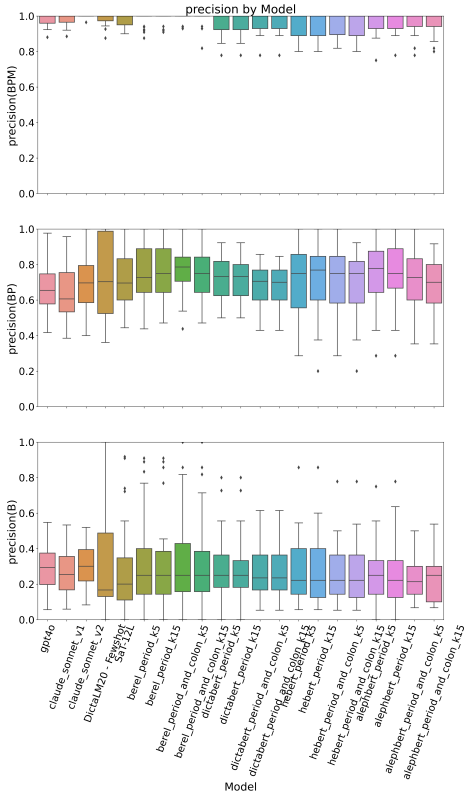


Figure 7: Precision across the combined corpus.

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

Aitor Alvarez, Carlos-D Martínez-Hinarejos, Haritz Arzelus, Marina Balenciaga, and Arantza del Pozo. 2017. Improving the automatic segmentation of subtitles through conditional random field. *Speech Communication*, 88:83–95.

Dennis Aumiller, Satya Almasian, Sebastian Lackner, and Michael Gertz. 2021. Structural text segmentation of legal documents. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*, pages 2–11.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Jesús Calleja, Thierry Etchegoyhen, and David Ponce. 2024. Automating Easy Read Text Segmentation. *Preprint*, arXiv:2406.11464.

Sangwoo Cho, Kaiqiang Song, Xiaoyang Wang, Fei Liu, and Dong Yu. 2022. Toward Unifying Text Segmentation and Long Document Summarization. *Preprint*, arXiv:2210.16422.

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.

Eep Talstra Centre for Bible and Computer. 2023. Bhsa: Biblia hebraica stuttgartensia amstelodamensis. https://github.com/ETCBC/bhsa. Accessed: 2025-03-18.

Federico Errica, Giuseppe Siracusano, Davide Sanvito, and Roberto Bifulco. 2024. What Did I Do Wrong? Quantifying LLMs' Sensitivity and Consistency to Prompt Engineering. *Preprint*, arXiv:2406.12334.

Markus Frohmann, Igor Sterner, Ivan Vulić, Benjamin Minixhofer, and Markus Schedl. 2024. "Segment Any Text: A Universal Approach for Robust, Efficient and Adaptable Sentence Segmentation". In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11908–11941, Miami, Florida, USA. Association for Computational Linguistics.

Hongyu Gong, Yelong Shen, Dian Yu, Jianshu Chen, and Dong Yu. 2020. Recurrent chunking mechanisms for long-text machine reading comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6751–6761, Online. Association for Computational Linguistics.

Amir Hazem, Béatrice Daille, Dominique Stutzmann, Christopher Kermorvant, and Louis Chevalier. 2020. Hierarchical text segmentation for medieval manuscripts. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6240–6251.

Timo Homburg and Christian Chiarcos. 2016. Akkadian word segmentation. In *Proceedings Tenth International Conference on Language Resource Evaluation.(LREC 2016)*, pages 4067–4074.

Benjamin Minixhofer, Jonas Pfeiffer, and Ivan Vulić. 2023. "Where's the Point? Self-Supervised Multilingual Punctuation-Agnostic Sentence Segmentation". In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7215–7235, Toronto, Canada. Association for Computational Linguistics.

David Ponce, Thierry Etchegoyhen, and Victor Ruiz. 2023. Unsupervised Subtitle Segmentation with Masked Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 771–781.

segment-any text. 2025. wtpsplit: Toolkit to segment text into sentences or other semantic units in a robust, efficient and adaptable way. `https://github.com/segment-any-text/wtpsplit`. Accessed: 2025-01-22.

Amit Seker, Elron Bandel, Dan Bareket, Idan Brusilovsky, Refael Shaked Greenfeld, and Reut Tsarfaty. 2021. Alephbert:a hebrew large pre-trained language model to start-off your hebrew nlp application with. *Preprint*, arXiv:2104.04052.

Wei Shi, Shuang Li, Kerun Yu, Jinglei Chen, Zujie Liang, Xinhui Wu, Yuxi Qian, Feng Wei, Bo Zheng, Jiaqing Liang, et al. 2024. SEGMENT+: Long Text Processing with Short-Context Language Models. *arXiv preprint arXiv:2410.06519*.

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

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, and Moshe Koppel. 2023. Dictabert: A state-of-the-art bert suite for modern hebrew. *Preprint*, arXiv:2308.16687.

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.

A Appendix: LLM Prompts

In this appendix, we share the prompts used when using Generative AI LLMs for segmentation, to allow full reproducibility.

A.1 DictaLM2.0 Few-shot prompt

Figure 8 provides the shots we used for the few-shot flow with DictaLM2.0 (Shmidman et al., 2024a). For each shot we provided the raw text (using the *"raw"* key) and the segmented text (using the "chunked" key"). Segmentation points were marked using double-slash ("//"). Few-shot flows work by providing the LLM the shots as context, and then the input text as a new input, and then allowing the LLM to continue by completing the output. LLMs have been found to pick up on patterns in the "shots" and then apply them directly to the new input text.

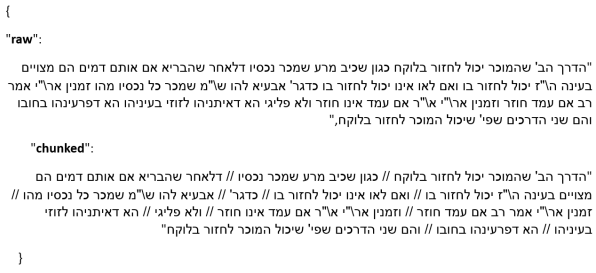
It is important to note that few-shot flows are very brittle. Specifically, they will react differently even when the order of the shots is changed, or the key names (in our case - "raw" and "chunked") are changed. Thus, for reproducing the results shown in our paper, please make sure to use the exact setup we used, as can be found in our github repo.

A.2 Closed LLMs Instruct prompt

For the closed LLMs, we used the following prompt template:

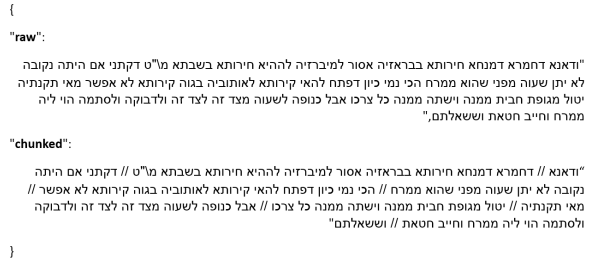
```
"please take the following unpunctuated text, and punctuate it. Punctuation includes periods, commas, question marks, semicolons and colons. Other than punctuating, keep the text exactly as it is. If a word is clipped at the end, like 'אמ, leave it like that. Return the punctuated text between <punctuation> tags: $text Punctuated text:"
```

During runtime, the *$text* variable is assigned the value of the inputted text. The *"Punctuated text"* text is the prefix of the reply, which for models such as Claude is a "hint" to nudge the model into following the instructions.



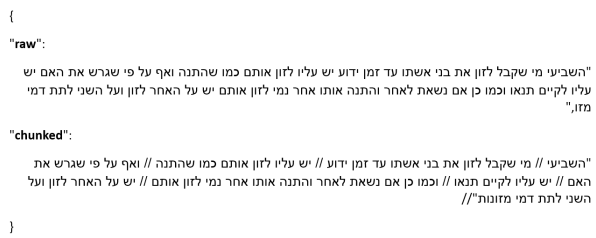
```
{
"raw":
"הדרך הב' שהמוכר יכול לחזור בלוקח כגון שכיב מרע שמכר נכסיו דלאחר שהבריא אם אותם דמים הם מצויים בעינה ה\"ז יכול לחזור בו ואם לאו אינו יכול לחזור בו כדגר' אבעיא להו ש\"מ שמכר כל נכסיו מהו זמנין ארו\"י אמר רב אם עמד חוזר וזמנין ארו\"י א\"ר אם עמד אינו חוזר ולא פליגי הא דאיתניהו לזוזי בעיניהו הא דפרעינהו בחובו והם שני הדרכים שפי' שיכול המוכר לחזור בלוקח,"
"chunked":
"הדרך הב' שהמוכר יכול לחזור בלוקח // כגון שכיב מרע שמכר נכסיו // דלאחר שהבריא אם אותם דמים הם מצויים בעינה ה\"ז יכול לחזור בו // ואם לאו אינו יכול לחזור בו // כדגר' // אבעיא להו ש\"מ שמכר כל נכסיו מהו // זמנין ארו\"י אמר רב אם עמד חוזר // וזמנין ארו\"י א\"ר אם עמד אינו חוזר // ולא פליגי // הא דאיתניהו לזוזי בעיניהו // הא דפרעינהו בחובו // והם שני הדרכים שפי' שיכול המוכר לחזור בלוקח"
}
```

(a) shot #1



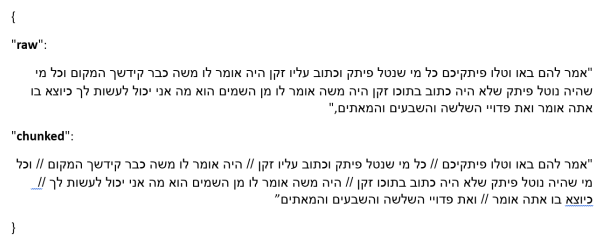
```
{
"raw":
"ודאנא דחמרא דמנחא בבראזיה אסור למיברזיה להיא חירותא בשבתא מ\"ט דקתני אם היתה נקובה לא יתן שעוה מפני שהוא ממרח הכי נמי כיון דפתח להאי קירותא לאותוביה בגוה קירותא לא אפשר מאי תקנתיה יטול מגופת חבית ממנה וישתה ממנה כל צרכו אבל כנופה לשעוה מצד זה לצד זה ולדבוקה ולסתמה הוי ליה ממרח וחייב חטאת וששאלתם,"
"chunked":
"ודאנא // דחמרא דמנחא חירותא בבראזיה אסור למיברזיה להיא חירותא בשבתא מ\"ט // דקתני אם היתה נקובה לא יתן שעוה מפני שהוא ממרח // הכי נמי כיון דפתח להאי קירותא לאותוביה בגוה קירותא לא אפשר // מאי תקנתיה // יטול מגופת חבית ממנה וישתה ממנה כל צרכו // אבל כנופה לשעוה מצד זה לצד זה ולדבוקה ולסתמה הוי ליה ממרח וחייב חטאת // וששאלתם"
}
```

(b) shot #2



```
{
"raw":
"השביעי מי שקבל לזון את בני אשתו עד זמן ידוע יש עליו לזון אותם כמו שהתנה ואף על פי שגרש את האם יש עליו לקיים תנאו וכמו כן אם נשאת לאחר והתנה אותו אחר נמי לזון אותם יש על האחר לזון ועל השני לתת דמי מזו,"
"chunked":
"השביעי // מי שקבל לזון את בני אשתו עד זמן ידוע // יש עליו לזון אותם כמו שהתנה // ואף על פי שגרש את האם // יש עליו לקיים תנאו // וכמו כן אם נשאת לאחר והתנה אותו אחר נמי לזון אותם // יש על האחר לזון ועל השני לתת דמי מזונות"//
}
```

(c) shot #3



```
{
"raw":
"אמר להם באו וטלו פיתקיכם כל מי שנטל פיתק וכתוב עליו זקן היה אומר לו משה כבר קידשך המקום וכל מי שהיה נוטל פיתק שלא היה כתוב בתוכו זקן היה משה אומר לו מן השמים הוא אני יכול לעשות לך כיוצא בו אתה אומר ואת פדויי השלשה והשבעים והמאתים,"
"chunked":
"אמר להם באו וטלו פיתקיכם // כל מי שנטל פיתק וכתוב עליו זקן // היה אומר לו משה כבר קידשך המקום // וכל מי שהיה נוטל פיתק שלא היה כתוב בתוכו זקן // היה משה אומר לו מן השמים הוא מה אני יכול לעשות לך // כיוצא בו אתה אומר // ואת פדויי השלשה והשבעים והמאתים"
}
```

(d) shot #4

Figure 8: Shots used in few-shot segmentation flow using DictaLM2.0. Note how we varied the type of input, and specifically ensured that some cases had segmentation points at the end of the text and some not, so as to encourage the model away from simplistic segmentation rules such as "end of line".