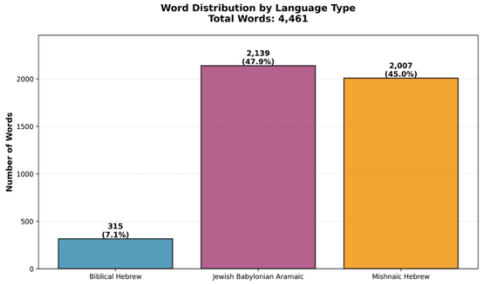
TalmudTokenizer: Framework for Analysis of Babylonian Jewish Legal Texts

Helfgot, Shlomi[1]; Advisor: Dr. Ruzica Piskac[1]; Consulting Advisor: Prof Scott Shapiro[2]
[1]Dept. of Computer Science, Yale University | [2]Yale Law School

Motivation

- **Tokenization** is essentially a mapping of words into numbers, coupled with compression, to enable efficient computational representation of text for NLP and ML tasks.
- Tokenizing the **Talmud**, a text comprising Jewish law from the 3rd-7th centuries, composed in three languages (Biblical Hebrew, Aramaic, Mishnaic Hebrew), would allow for the undertaking of various important digital humanities projects, e.g.
  - **Automated vocalization** for Aramaic.
  - Pipeline for detection of **intertextual allusions**.
  - **Stylometric dating** of various levels of redaction.
  - **Semantic RAG systems** which might allow the user to query the Talmud.



Problem Description

Which tokenization regimen is best for Talmud, as evaluated by:

1. **Intrinsic statistical analysis**
2. **Downstream performance on ML tasks (e.g. classification or generation)**

Unique Challenges

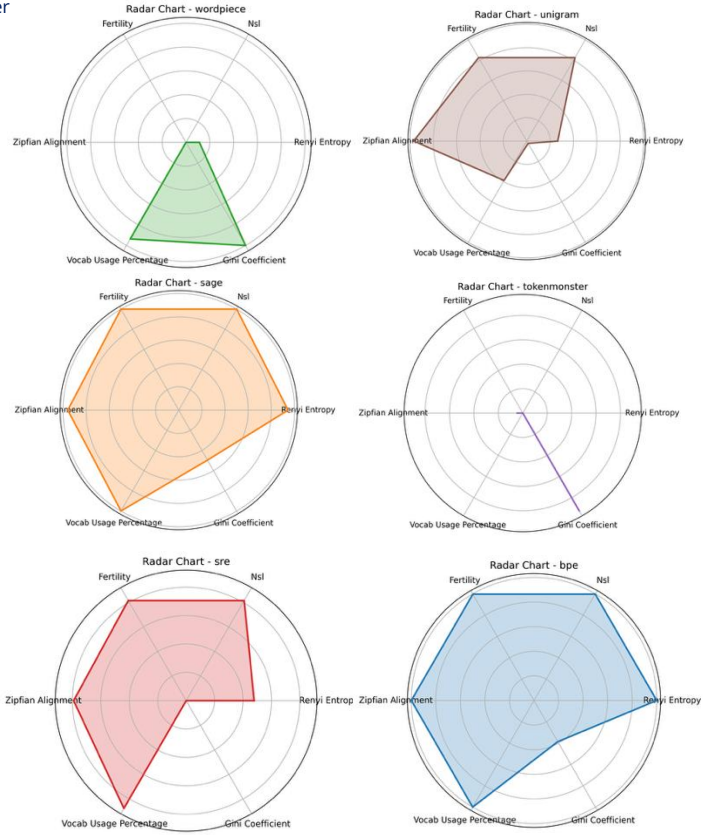
1. Talmud is written in Semitic languages. These are non-agglutinizing (i.e. not fully comprised of subwords), but rather based on root patterns and interpolated vowels. Hence while "walking" can be parsed into "walk" and "ing" in English, in Hebrew, we have *"h-l-k"* and separate meanings for '*h**o**lek*', '*h**a**lak*', '*h**a**lak**a**h*'.
2. Talmud comprises three languages (see sample distribution above); the tokenizer must be sensitive to cognates and shared roots.

TOKENIZER RESULTS

> **SRE** (Semitic Root Encoding) excels at morphology, and performs best on downstream tasks, as predicted.

> **SaGe** has best morphological coherence over the corpus.

> **WordPiece** fails catastrophically, as expected: the algorithm uses greedy matching to find subwords! Semitic triconsonantal root morphology implies that this will not work.



Experimental Setup

- We implemented six algorithms: three baseline (**BPE**, **Unigram**, and **WordPiece**) and three SOTA candidates (**TokenMonster**, **SRE**, and **SaGe**). WordPiece attempts to find subwords using a greedy algorithm; SRE, Semitic Root Encoding, preprocesses words using root templates (we used heuristics for this, given that no morphological analyzer exists for Aramaic), and SaGe (SkipGram Entropy) incorporates contextual information into vocabulary construction (it trains word embeddings and uses contextual predictability to guide merges.)
- Given the **vocabulary coverage curve** for unique words vis a vis tokens in the Talmud as well as the large-tail phenomenon of the text (i.e., many hapax legomenon) as seen using Zipf's law, we used a vocabulary of **32k** words **(88%** coverage.)
- **Criteria for intrinsic evaluation**: fertility and NSL (calculate compression statistics), Renyi entroy (measure of how well language is captured, especially homophony), vocabulary usage and Gini coefficient (distribution of token size)
- **Hypothesis:** standard metrics measure *Western*, agglutinizing language. SRE may suffer on compression but will exhibit better comprehension of the text owing to the root analysis built-in.

(pictured: sample BPE tokenization; sample SaGe tokenization. SaGe is more efficient, token-wise; 78 vs. 112 in this pericope from **Menachot 44a**)

Downstream Performance

SRE outshone all others in comprehension of the text, over various downstream benchmarks enumerated below!
(With the exception of the analogy test, which may have proven too nuanced for the embedding to capture.)