Linguistic Dating of the Hebrew Bible

Problem

Although several texts in the Hebrew Bible refer to historical events and can therefore be dated with relative certainty (e.g., Jeremiah 37:5), the vast majority do not. As a result, biblical scholars like Avi Hurvitz have used linguistic dating in order to establish a chronological horizon for undated texts. So far these scholars have largely worked "by hand," laboriously sifting and weighing individual linguistic features. Their research suggests that the Hebrew language passed through four phases:

Phase	Absolute Dates	Representative Text
Archaic	1200-1000 BCE	Exodus 15
Classical	1000-586 BCE	1 Samuel
Transitional	586-539 BCE	Jeremiah
Late	539 BCE onward	Ezra

In many verses, however, the chronological signals are too faint for human researchers to detect and so these texts remain unassigned. The current project attempts to solve this problem using a combination of natural language understanding and deep learning. I trained a language model for Biblical Hebrew and developed a classification model to sort its outputs into chronological phases.

Approach

The data for this project came from Text-Fabric, a python package created by scholars at the Vrije Universiteit Amsterdam for preforming statistical analysis of the Bible and other ancient texts. Text-Fabric's internal structure differs significantly from the tabular data familiar to data scientists. It stores the biblical text as a series of interconnected nodes representing words, clauses, sentences, verse and chapters. As a result, I needed to perform a significant amount of data wrangling at the outset of the project: I extracted each verse into its own row, added a column for chapter and verse numbers, updated the chapter names to fit scholarly convention, set verses as an index to make the DataFrame easier to navigate, fixed a mistake in the representation of 1 Kings 16:26-27; and removed verses in Aramaic, a Semitic language related to Hebrew.

I then created a labeled dataset based on my own expertise as a former biblical scholar. I chose the following verses to represent each chronological phase:

Phase	Verses	Total
ABH	Genesis 49:2-27; Exodus 15:1-18; Numbers 23:7-10, 18-24; 24:3-9, 15-24; Deuteronomy 32:1-43; 33:2-29; Judges 5:2-31; 2 Samuel 22:2-51; Psalms 18:1-50; 68:2-35	307
CBH	1 Samuel; 2 Samuel; 1 Kings; 2 Kings	3042
TBH	Isaiah 40-54; Jeremiah; Ezekiel	2956
LBH	Ecclesiastes; Esther; Daniel; Ezra; Nehemiah; 1 Chronicles 1:1-9:44; 12:1-40; 15:1-24; 16:7-43; 21:1-29:19; 2 Chronicles 7:1-3; 14:9-15:7; 17:1-19; 21:12-17; 24:15-22; 26:6-21; 29:3-31:21; 33:10-20; 34:3-7; 36:22-23	2087

At this point, a potential problem became apparent: the number of ABH verses is much smaller than the other samples. So—in order to avoid working with unbalanced data—I down sampled the other three groups.

The next step in the project was train a language model for Biblical Hebrew. I chose a RoBERTa architecture for this purpose for two reasons. First, it uses an auto-encoding algorithm which is well suited to natural language understanding tasks. Second, it uses a byte-level BPE encoder which can handle symbols outside of the Latin-1 character set. I trained the model for 10 epochs in a Colab Pro notebook using a Tesla P100 GPU. Continuing the tradition of punning names for autoencoders (e.g., RoBERTa, CameBERT), I named the finished model BERiT after the biblical Hebrew word for 'covenant', *bərît*. Like these other models, BERiT transforms words and verses into 768 dimensional vectors.

Once training was complete, I performed a few tests to gauge BERiT's understanding of Biblical Hebrew. When asked to predict the identity of the masked word in the phrase '*ha māqôm ha* <mask>', for example, BERiT identified five common and plausible options (Fig. 1). These results indicate that BERiT has a good grasp of Biblical Hebrew syntax and semantics. But it was unclear whether BERiT captured any chronological information about Biblical Hebrew.

```
[{'sequence': 'ha māqôm  ha zê',
  'score': 0.45529893040657043,
  'token': 285,
  'token_str': ' zê'},
 {'sequence': 'ha māqôm  ha hû',
  'score': 0.05017251893877983,
  'token': 255,
  'token_str': ' hû'},
 {'sequence': 'ha māqôm  ha šəbîʕî',
  'score': 0.017493825405836105,
  'token': 1897,
  'token_str': ' šəbîʕî'},
 {'sequence': 'ha māqôm  ha yôm',
  'score': 0.016817649826407433,
  'token': 246,
  'token_str': ' yôm'},
 {'sequence': 'ha māqôm  ha šəlîšî',
  'score': 0.015264025889337063,
  'token': 2509,
  'token_str': ' šəlîšî'}]
```

This place

That place

The seventh place

The place today

The third place

Fig. 1: An example of BERiT's predictive power

To test whether this was the case, I checked if the verse embeddings formed relatively discrete clusters. I used PCA to reduce the 768 "features" of each verse embedding to the two most salient ones for visualization (Fig. 2).

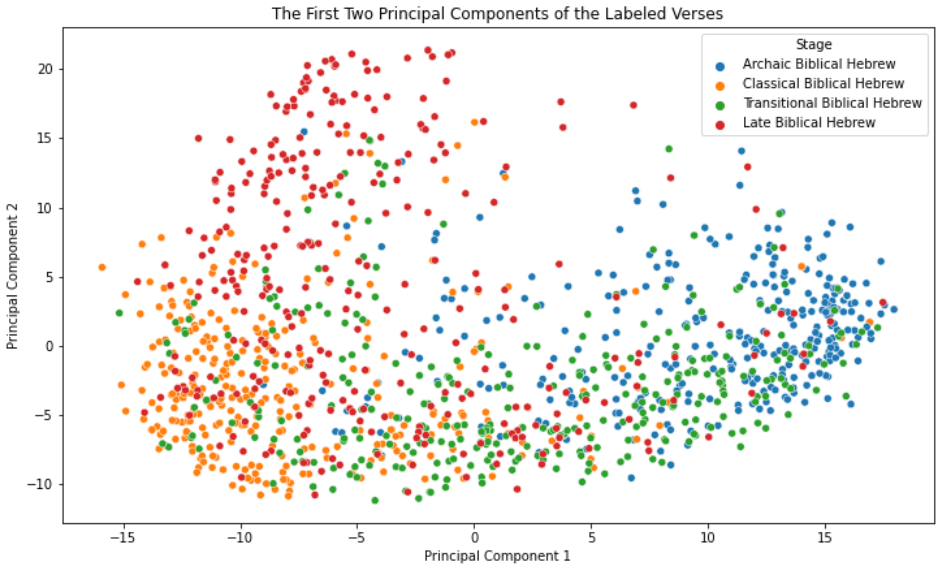

Fig. 2: Distribution of the first two principal components of the labeled verse embeddings

At first glance, the groupings were not as clear as I would have hoped. Although ABH, CBH, and to a lesser extent LBH cluster in distinct regions of the graph, TBH is much more diffuse, overlapping significantly with ABH and CBH. It is important to remember, however, that language change is a slow, continous process. For every linguistic feature that changes, many others remain the same. Areas of overlap between the different stages may represent verses that contain fewer distinctive chronological features. It's also possible that the four groups are more easily distinguishable in higher dimensions.

The position of TBH is particularly strange because TBH represents the transitional phase between CBH and LBH and so we would expect it to overlap with these groups. One possible explanation for this outcome is that TBH texts contains a mixture of poetry and prose, while CBH and ABH texts are predominantly written in either poetry (ABH) or prose (CBH). But such a hypothesis does not explain why TBH does not overlap as much with LBH, which is also written primarily in prose.

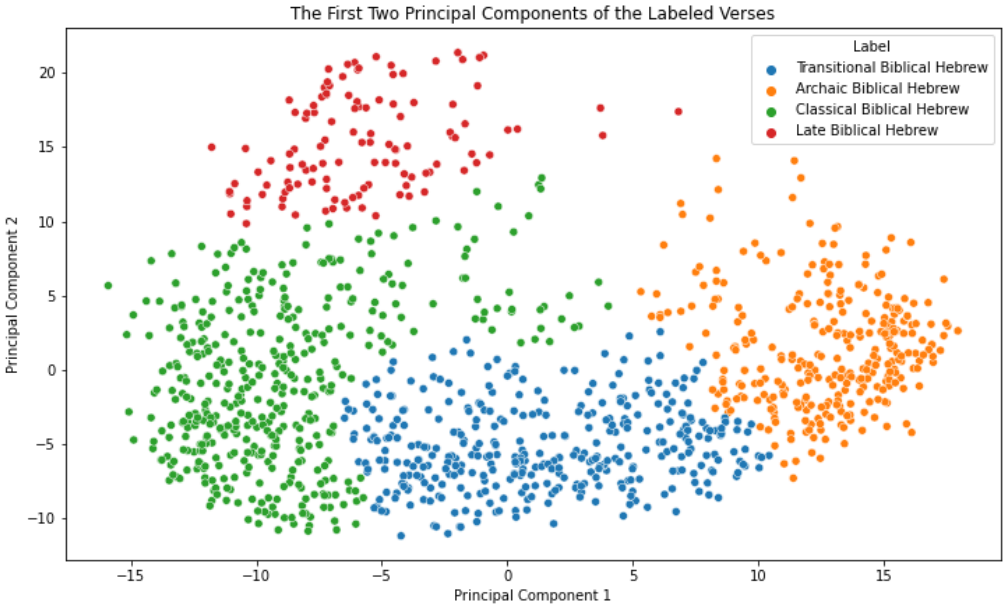

Fig. 3: An ideal grouping of the data generated with agglomerative clustering

I compared the actual distribution of the data to an ideal clustering generated by unsupervised learning (Fig. 3) using a heatmap (Fig 4 below) The actual grouping matches the clusters generated by unsupervised learning remarkably well for ABH, CBH, and TBH. LBH remains problematic, having lost nearly half of its datapoints to CBH. But I felt more confident that the verse embeddings capture some chronological information.

Fig. 4: Heatmap comparing the distribution of the data with an ideal clustering

For the final stage of modeling, I designed a shallow neural network to classify the verses of the Hebrew Bible into one of four chronological phases. My initial model consisted of a single dense layer followed by a softmax layer. I then used Bayesian optimization to tune the following hyperparameters on the basis of accuracy over the course of 50 epochs: 1) the size of each dense layer (32-1152 neurons) and the dropout rate (15-25%); 2) the number of dense layers (1-4); 3) the activation function (ReLU, SeLU, softsign, or tanh); and 4) the weight initialization function for the model (Glorot normal or He normal). I chose accuracy as the evaluation metric because the final model should pay equal attention to each class. The best model consisted of three dense layers with 576 Neurons a piece and ReLU activation function alternating with three 16% dropout layers and used Glorot normal as the weight initialization function.

Findings

The final model achieved 71.2% accuracy on the test set, which is impressive considering the small size of the training data (982 verses). A heatmap (Fig. 5) comparing the true and predicted labels for each verse in the text set shows that the model does an excellent job of identifying ABH and—to a lesser extent—LBH and CBH. But it struggles with TBH. This result is consistent with the results of exploratory data analysis, which showed that TBH overlapped significantly with ABH and CBH.

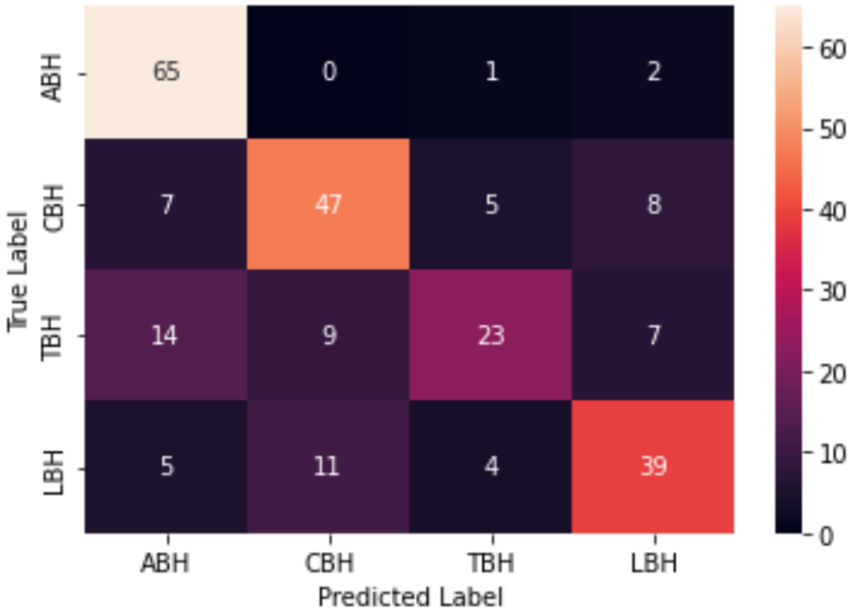

Fig. 5: Heatmap comparing the predictions of the best classification algorithm with the test data

Because the training set was incredibly small, I was curious whether using a larger training set would lead to improved accuracy. So I retrained the best model using the full labeled data set (8492 verses) and assessed its performance. This revised model achieved a maximum accuracy of 74.9% on the test data, representing a gain of 2.4% over training the model on a subsample. But this gain came at a price: the almost complete inability of the model to distinguish ABH from the other classes (Fig. 6). In my opinion, this trade-off is not worth it.

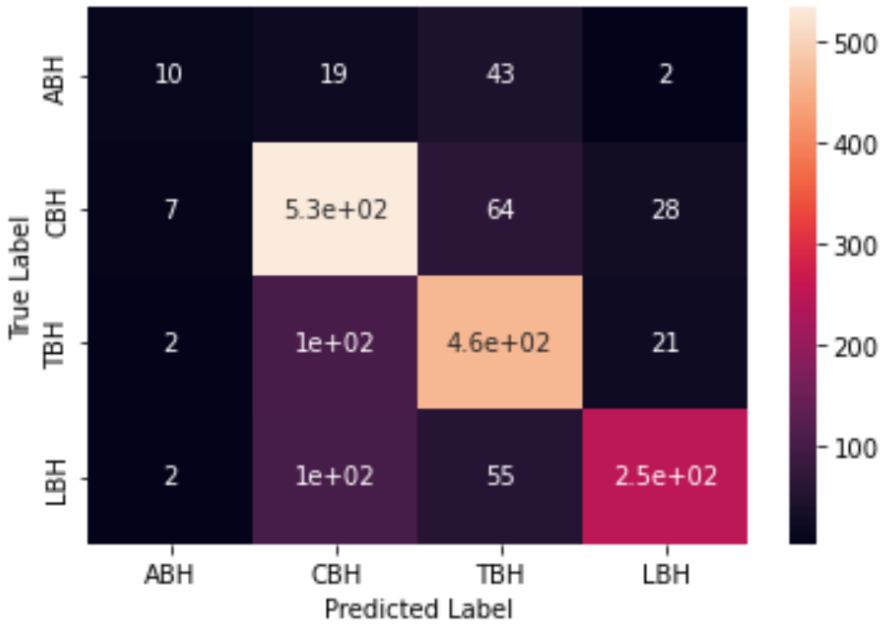

Fig. 6: Heatmap comparing the predictions of the second classification algorithm with the test data

Recommendations

Based on my results, I would make the following recommendations regarding the use of Machine Learning for classifying Biblical Hebrew texts:

1. Use a RoBERTa-like architecture for modeling Biblical Hebrew
2. Use a balanced data set for training
3. Use accuracy as an evaluation metric for classification

Ideas for Future Research

I foresee three possibilities for future work: increasing training time, modifying the training data, and augmenting interpretability. The first possibility is fairly straightforward: further training of both BERiT and the classification model—while tedious and computationally expensive—would undoubtedly improve the results.

The second avenue for further work involves two tweaks to the training data itself. First, I suspect that training BERiT on individual sentences rather than entire verses would lead to enhanced performance. Verse divisions, after all, represent a later, mediesval addition to the Hebrew Bible, which do not always respect editorial boundaries. Some verses contain material written by different authors, writing at different times. Numbers 24:3, for instance, pairs a prose sentence in CBH (“And he took up his discourse and said”) with two lines of poetry in ABH (“The oracle of

Balaam, son of Beor. The oracle of the man whose eye is opened.”). Such composite verses send mixed signals to the classification algorithm and would be better processed separately. Second, up sampling the ABH data would allow me to train the classification algorithm on a larger data set without sacrificing its ability to recognize ABH.

Finally, interpretability remains an issue. Because verse embeddings represent abstract generalizations from individual words and grammatical features, it is unclear how the classification algorithm arrives at particular results. An interpretability package for transformers like ecco or transformers interpret could help overcome this problem by identifying the input tokens that are most salient for each outcome. In particular, I would be interested in comparing the features that the classification algorithm identifies as significant with the diagnostic features identified by human scholars. Has the algorithm learned something new or simply recreated human efforts?