Final Checkpoint

By Viktoria Fahnemann, Nicole Li, Nyambura Njenga-Benton, Nandha Thiru, and Emily Zhao

Introduction/Background

- Literature Review: Cuneiform, an ancient writing system used by the Sumerians, Akkadians, and Babylonians, is made of wedge-shaped impressions on clay tablets [1]. Analyzing cuneiform has been a challenge in both archaeology and linguistics. Thanks to machine learning, automatic recognition and classification of these symbols is becoming more viable. Prior work to classify transcriptions has used algorithms such as K-means, SVMs, decision trees, random forests, and neural networks [2], [3]. Recent works have even used CNNs to directly analyze images of tablets [4].

- Dataset Description: This dataset consists of 134,000 cuneiform text snippets represented in unicode symbols and labeled by one of seven ancient languages/dialects.
- Dataset Link

Language Timeline:

- SUX: Sumerian, 3100–1600 BC
- Old Akkadian, 2500–1950 BC
- OLB: Old Babylonian and Old Assyrian, 1950–1530 BC
- STB, MPB: Middle Babylonian and Middle Assyrian, 1530–1000 BC
- NEB, NEA: Neo-Babylonian and Neo-Assyrian, 1000–600 BC
- LTB: Late Babylonian, 600 BC–100 AD

Problem Definition

- We aim to distinguish between different languages/dialects of transcribed cuneiform. We will classify the snippets with ML models such as decision trees, random forests, and neural networks. If time allows, we would also like to investigate more experimental strategies such

as GMMs and LSTMs. Finally, we aim to create a novel hybrid model that combines two of the most successful algorithms and differs from approaches in previous literature.

Methods

Data Preprocessing:

1. Since the category sizes are uneven, we balanced the numbers by oversampling from smaller categories. We did this methematically for only the training data to avoid contamination. This method was shown to increase accuracy in previous work [4] and it helped us as well.
2. We implemented unigram extraction of the symbols to vectorize the cuneiform texts so the algorithms can interpret them.
3. To represent the categories for our supervised methods, we implemented one-hot encoding. This prevents algorithms from learning unwanted dependencies between our nominal categories.

ML Algorithms/Models Implemented:

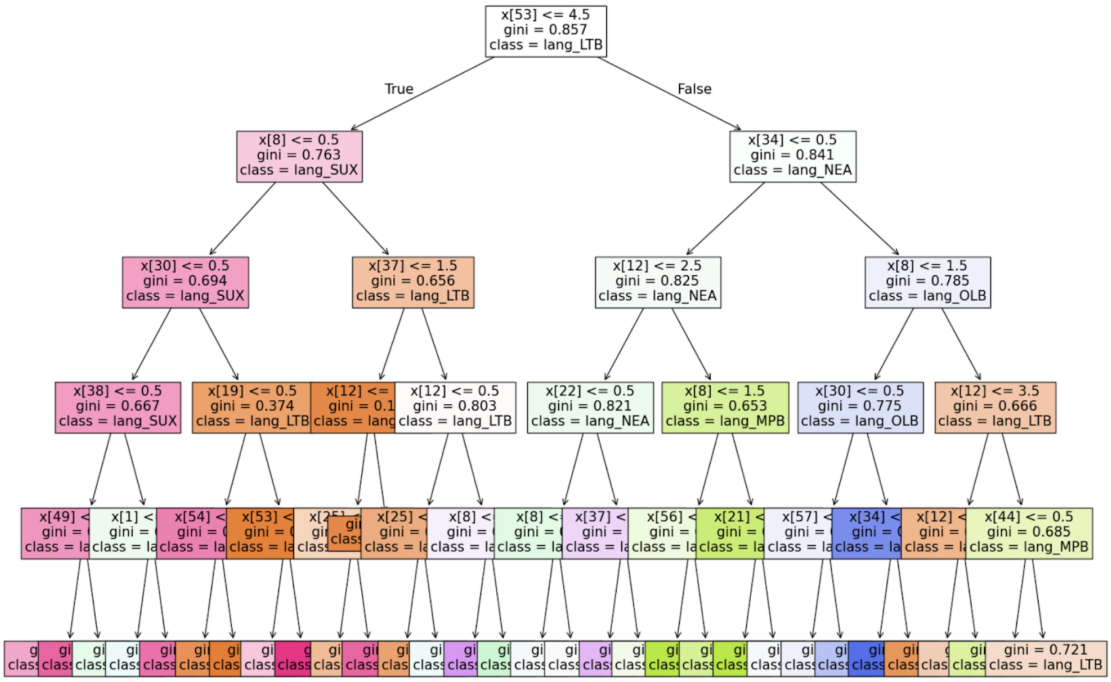
1. Decision tree: Binary decisions, like certain words only belonging to one of the languages, will be helpful for classifying languages. The interpretable nature of decision trees will allow us to easily generate hypotheses.
2. Neural network: Deeper patterns, such as language grammar, can be discovered using a neural network. The structure of the neural network will also allow us to easily create a hybrid model.
3. Random forest: Edge cases, such as incorrect or uncommon grammatical structures, may throw off a single decision tree. A random forest would make the results less swayed by the edge cases.
4. (Not Implemented) GMM: We were planning to use GMM to see if it can categorize the phrases into languages unsupervised. Due to cultural diffusion, some segments of text may appear plausible for multiple languages. Soft assignment may allow for combined models to leverage likelihood to make decisions. We decided not to implement it due to time restriction.

CS: 4641: All methods are supervised

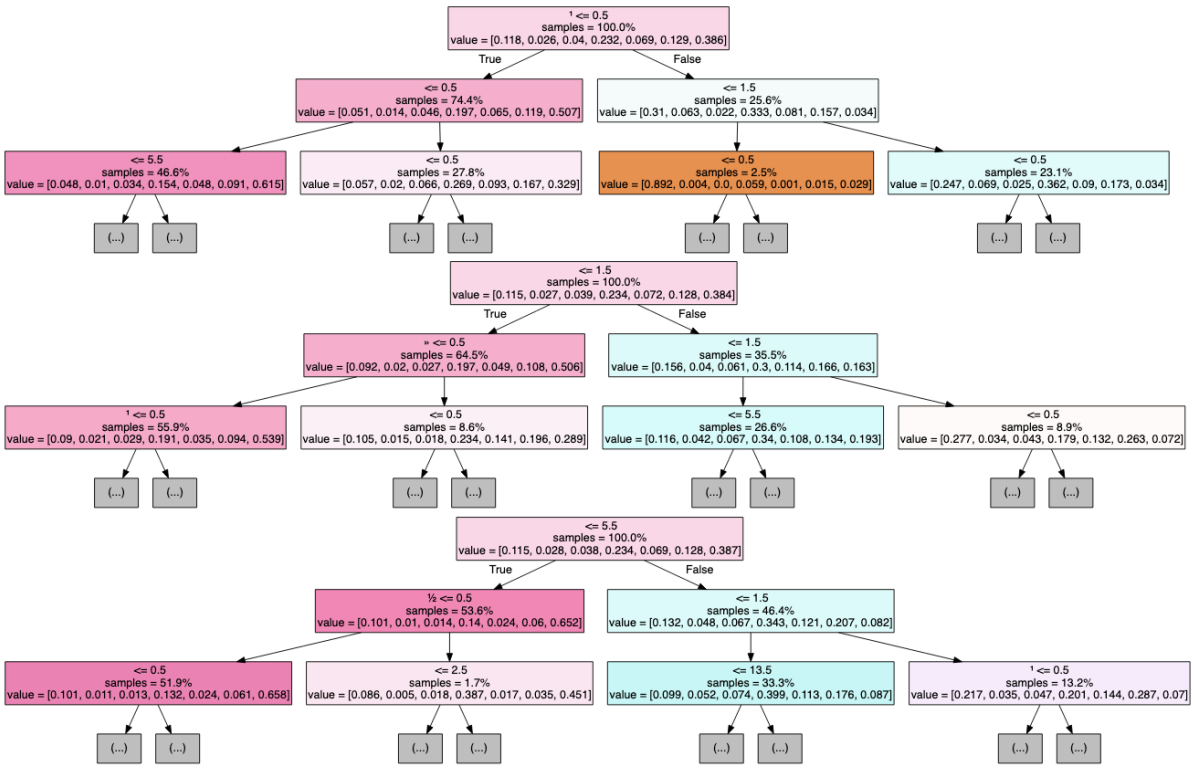
Results and Discussion

Visualizations

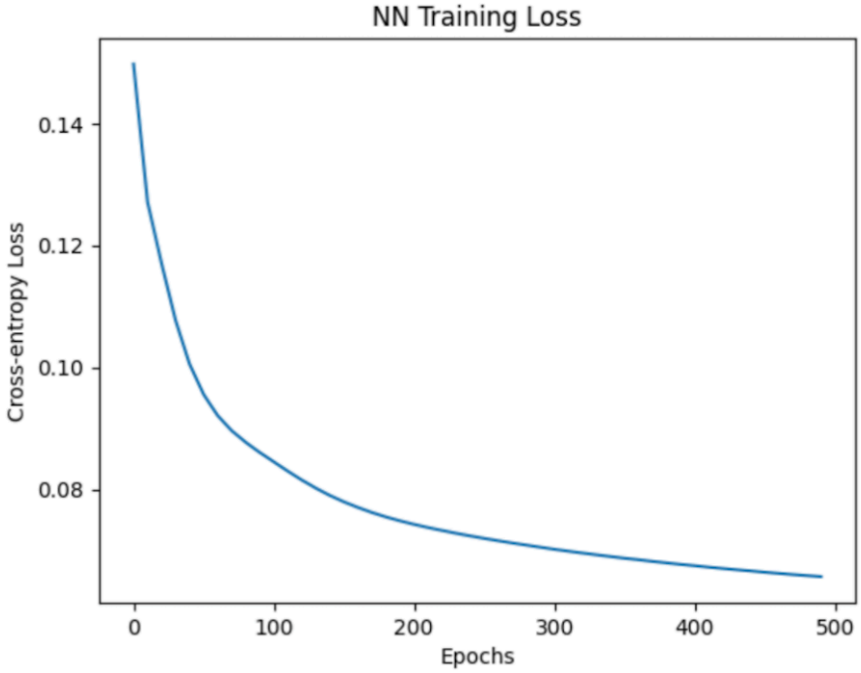
Decision Tree:



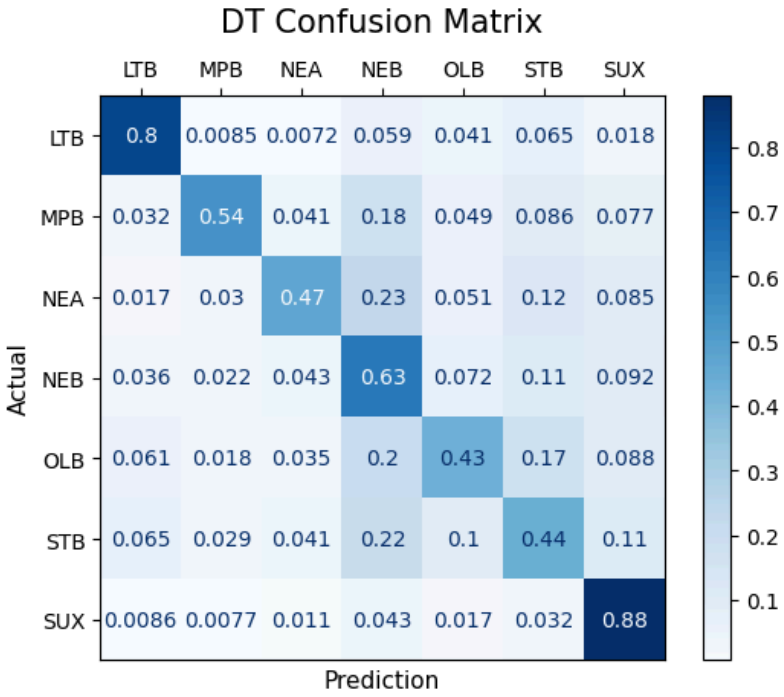
Sample of Random Forest Trees:



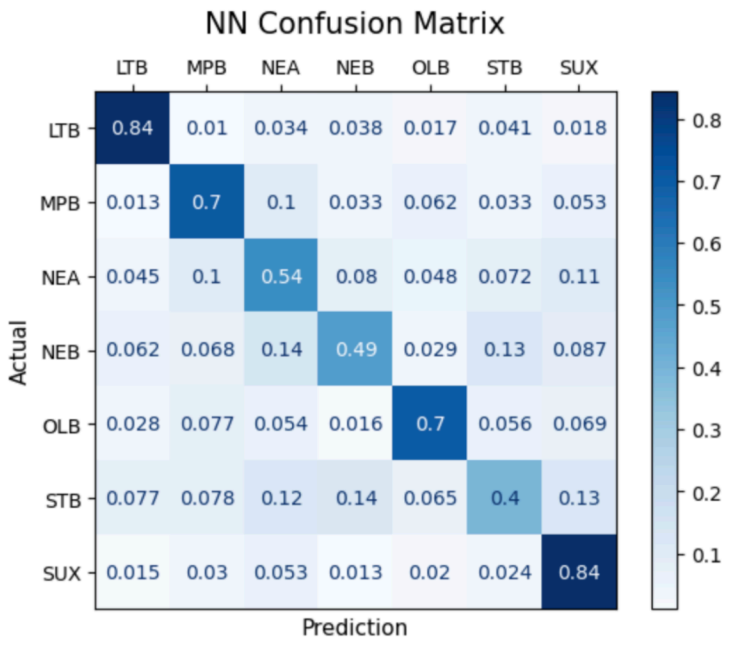
Neural Network Loss:



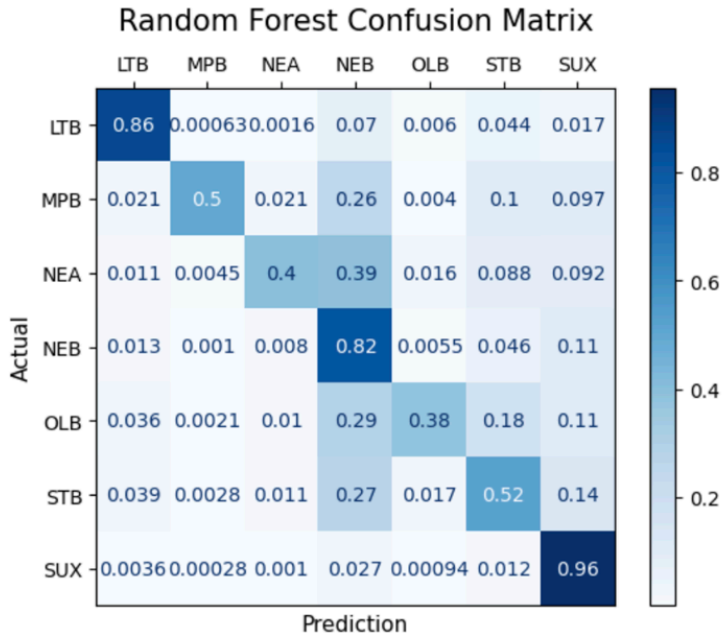
Decision Tree Confusion Matrix:



Neural Network Confusion Matrix:



Random Forest Confusion Matrix:



Quantitative Metrics

1. Confusion matrix: We generated confusion matrices for all classes to identify where our algorithm excels and where it tends to confuse one language for another.
2. Macro-averaged F1 score: A single score combining both precision and recall gives us an overview of whether a model is working in the right direction.
3. Weighted F1 score: By evaluating the model on its F1 score weighted by class, we obtain a general performance score. Weighted average gives more importance to classes that occur

more often.

Decision Tree Classification Report:

	precision	recall	f1-score	support
LTB	0.78	0.80	0.79	3176
MPB	0.49	0.54	0.51	756
NEA	0.44	0.47	0.45	1113
NEB	0.66	0.62	0.64	6725
OLB	0.39	0.43	0.41	1895
STB	0.46	0.44	0.45	3573
SUX	0.87	0.88	0.88	10647
accuracy			0.70	27885
macro avg	0.58	0.60	0.59	27885
weighted avg	0.70	0.70	0.70	27885

Decision Tree Folds:

```
Fold 1 Mean Accuracy on Test Data: 0.5968760362190478
Fold 2 Mean Accuracy on Test Data: 0.9725343587230276
Fold 3 Mean Accuracy on Test Data: 0.9733054136350118
Fold 4 Mean Accuracy on Test Data: 0.972950598186418
Fold 5 Mean Accuracy on Test Data: 0.9720778502202342
```

Neural Network Classification Report:

	precision	recall	f1-score	support
LTB	0.74	0.84	0.79	3176
MPB	0.35	0.73	0.48	1113
NEA	0.72	0.54	0.62	6725
NEB	0.42	0.47	0.45	1895
OLB	0.36	0.71	0.48	756
STB	0.55	0.41	0.47	3573
SUX	0.85	0.85	0.85	10647
accuracy			0.69	27885
macro avg	0.57	0.65	0.59	27885
weighted avg	0.71	0.69	0.69	27885

Random Forest Classification Report:

	precision	recall	f1-score	support
LTB	0.88	0.85	0.87	3176
MPB	0.93	0.51	0.66	756
NEA	0.76	0.41	0.53	1113
NEB	0.67	0.81	0.73	6725
OLB	0.82	0.38	0.52	1895
STB	0.63	0.52	0.57	3573
SUX	0.86	0.96	0.90	10647
accuracy			0.78	27885
macro avg	0.79	0.63	0.68	27885
weighted avg	0.78	0.78	0.77	27885

Analysis of 3+ Algorithm/Model

Decision Tree:

First off, what is a Decision Tree model and why did we choose it? Decision Tree is a type of supervised ML model that uses a tree-like structure to make decisions based on input features. Based on the feature “value” we can split them into branches as observed in the visual; in this case the value is gini (an entropy measurement for information gain). Predictions are made at the final level when we reach a leaf in the tree. We selected a decision tree for one of our models for multiple reasons. First, it is very easy to interpret how the model is making decisions, and what features led to the classification of what language. It is able to be visualized clearly. Another reason is because decision trees work well with discrete data, and Cuneiform symbols are discrete distinct characters. Another reason why we tried Decision Tree is because it is computationally cheap and was not resource intensive to train. Now let’s look at the results. The macro-average of our F1 scores for each class was 0.589, and the weighted average was 0.7. This is relatively low, with some classes having much higher accuracies than others. Looking at the confusion matrix we can see that SUX and LTB have very high scores of 0.88 and 0.8 and NEB and MPB are relatively high with 0.63 and 0.54. Most of the other classes scored around 0.45. This score can be caused by class imbalance in sample size, visual similarity between some classes(“NEA” is misclassified as “NEB” in 23% of cases), or our decision tree is not complex enough. We can improve our accuracy and F1 score by using Dimension Reduction techniques such as PCA, or by using more Decision Trees in a Forest similar to a Random Forest, and by potentially improving our feature engineering.

Neural Network:

What is a Neural Network and why did we choose it? A Neural Network is a ML model that is similar to that of the human brain in the sense that it is made up of layers of many connected nodes (comparable to neurons). Each node processes input data, applies weights and biases, and passes it through an activation function to produce an output. These outputs feed into other neurons in

subsequent layers, allowing the network to learn complex patterns and make predictions. Neural Networks are very good at classification tasks, which is one of the reasons we selected it for Cuneiform language identification. Neural Nets with multiple layers are able to accurately capture nonlinear patterns that may appear in languages with similar symbols such as Cuneiform languages. Our prediction was that the Neural Net would perform better than our Decision Tree. Let's look at our results and see. After training over 500 epochs and tracking the Cross Entropy Loss, we can see that the Training Loss steadily decreased from 0.1458 to 0.0659. The testing loss was 0.07. This is a good sign that shows that the fitting is going well. When looking at the confusion matrix, we have great results with very accurate classifications of the LTB, SUX, MPB, and OLB classes and an overall macro-averaged F1 score of 0.59 and weighted average at 0.68. The Neural Network was able to perform comparably than the Decision Tree. Interestingly, as can be seen from the confusion matrix, the neural network and decision tree seemed to be good/bad at classifying different classes than each other.

Random Forest:

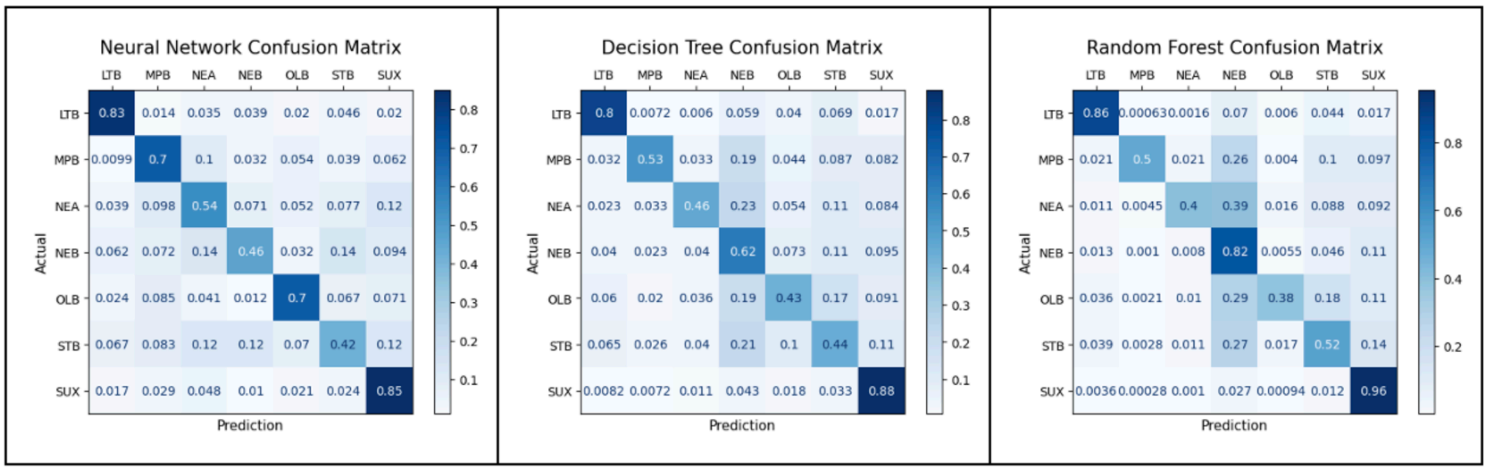
Random forest is a type of ensemble learning algorithm that uses Decision Tree as the base model. Random forest can be thought of as random bootstrap-aggregating (bagging) decision trees. Each decision tree in the random forest is trained on a subset of the training set and uses a randomly selected portion of the total features to make splits. Once all the decision trees have been constructed, we are ready to make predictions on a test sample. As the test sample is passed through the model, each tree predicts a label. The final prediction of the random forest is the majority decision of all the decision trees. The random forest model offers several advantages over using an individual Decision Tree model. For instance,

1. Overfitting is reduced since each tree is only using a portion of the total features
2. Combining multiple decision trees and using majority voting reduces the variance of the predictions, leading to more stable outcomes
3. Unusual edge cases/outliers cannot impact the outcome of the model since each decision tree is only trained on a subset of the training data

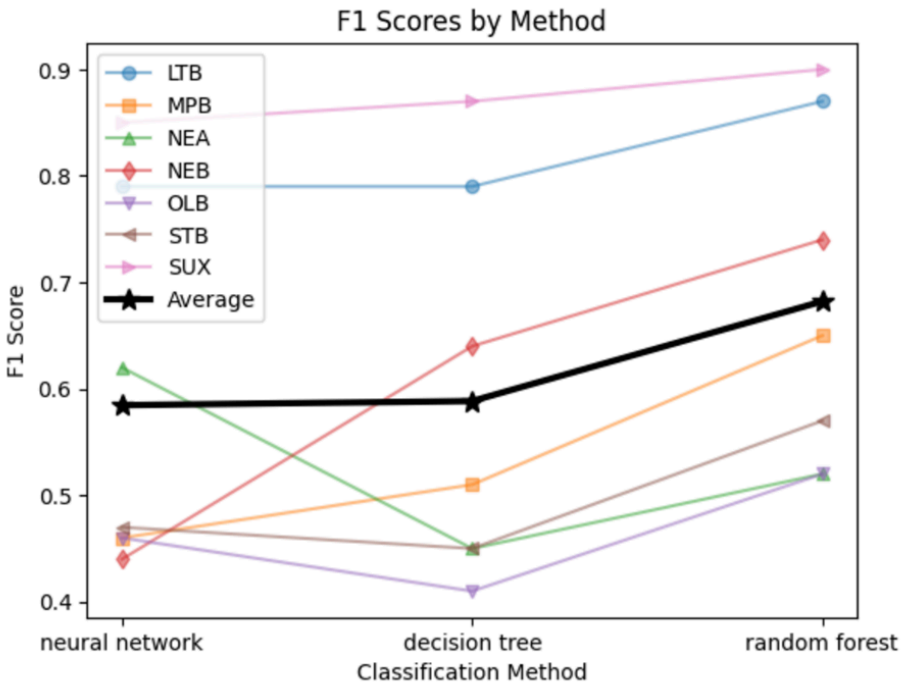
Our model consists of 50 independent decision trees. Each tree is trained on 50% of the dataset's samples and eight of the dataset's 66 features are randomly selected for the trees to use for splits. Our model's results demonstrate the strengths of Random Forests. The macro-averaged F1 score was 0.69, and the weighted average was 0.77. This is quite high and a significant improvement over the decision tree model. As with the decision tree model, some classes had higher accuracy than others likely due to imbalances within the dataset. Overall, the model demonstrated exceptional performance despite the complexity of the dataset.

Comparison of 3+ Algorithms/Models

The charts below show the confusion matrices. We can see that overall, all three classification methods definitely perform better than random chance, and the true positive rates range from 0.38 (OLB classified with random forest) to 0.96 (SUX classified with random forest).



We used F1 score as one of our primary metrics. The F1 scores per method for each class and as a weighted average are displayed in the below graph.



The neural network was one of the weaker classifiers, but it still performed acceptably in general. The strength of the neural network is its ability to identify deeper patterns by using combinations of neurons, which corresponds well to finding grammars in languages. This may explain its improved performance in identifying Neo-Assyrian (NEA) in comparison to our other two classifiers. Some tradeoffs of the neural network are the time the neural network takes to train (it trained the slowest out of all of our models) and that it is a black box—compared to the other two methods, it

is more difficult to understand why a phrase should belong to a certain language, which may make it less desirable for certain use cases.

The decision tree's performance was around the same as the neural network's performance but worse than the random forest's performance. The strength of the decision tree is that it will choose attributes that provide the greatest information gain, so the attributes in the tree are guaranteed to provide the most information. This made it easily able to identify words that are in one language but not others. However, a tradeoff/limitation is that the size of the tree can grow too big, which causes overfitting, but limiting the depth of the tree also limits the ability for the tree to make finer differentiations between similar languages. In the context of language classification from unigrams, the decision tree classifies languages in a simple manner based on whether a phrase has certain words. This is problematic because a phrase from a given language may not use any of the common words of that language. Due to these limitations, the decision tree was not the best classifier.

The random forest was the best classifier. It achieved this performance by covering for the weaknesses of the decision tree. A single decision tree is limited by its depth, but by training multiple trees using subsets of the data, the random forest was able to overcome this limitation without overfitting. Bagging also allows it to handle unusual phrases in the test dataset better, since the results of multiple trees will be considered, and each tree will have its own way to attempt to classify the unusual phrase. A tradeoff is that the random forest classifier takes much longer to train than a single tree, since multiple trees need to be trained, but it still trains faster than the neural network, and a limitation is that each tree's decision-making process is still relatively simple compared to the neural network, which may not be able to capture intricate grammatical concepts present in some languages but not others.

Next Steps

If we were to continue this project in the future, we would try implementing Gaussian Mixture Models to explore their effectiveness in classifying cuneiform text. GMMs could be better at modeling the underlying distribution of the cuneiform data, specifically where the languages exhibit overlapping characteristics. This method may help us incease our accuracy on similar languages. In the future we can also try benchmark our current models against Support Vector Machines (SVMs). This comparison will assess whether our implemented models can provide higher accuracy or better handle ambiguous cases in the cuneiform transcription data.

References

- [1] Jaan Puhvel, "cuneiform | Definition, History, & Facts," Encyclopædia Britannica. 2019. Available: https://www.britannica.com/topic/cuneiform

- [2] N. Edan, “Cuneiform Symbols Recognition Based on K-Means and Neural Network,” AL-Rafidain Journal of Computer Sciences and Mathematics, vol. 10, no. 1, pp. 195–202, Mar. 2013, doi: https://doi.org/10.33899/csmj.2013.163436.
- [3] M. Mahmood, F. M. Jasem, A. A. Mukhlif, and B. AL-Khateeb, “Classifying cuneiform symbols using machine learning algorithms with unigram features on a balanced dataset,” Journal of Intelligent Systems, vol. 32, no. 1, Jan. 2023, doi: https://doi.org/10.1515/jisys-2023-0087.
- [4] V. Yugay et al., “Stylistic classification of cuneiform signs using convolutional neural networks,” it - Information Technology, Apr. 2024, Accessed: Oct. 01, 2024. [Online]. Available: https://www.degruyter.com/document/doi/10.1515/itit-2023-0114/html#MLA

Gantt Chart

- Link

Proposal Contributions

Name	Proposal Contributions
Emily	Slides, random forest qualitative metrics
Nandha	Slides, Video Recording, Website, next steps
Nicole	Slides, model comparison
Nyambura	Slides, random forest bootstrapping, random forest analysis
Viktoria	Random forest implementation, random forest visualization

---