EN.520.666 Information Extraction from Speech
Project 1

March 3, 2025

Prabhav Singh
psingh54@jhu.edu
MSE CS

---

1 CODE

For the code, please visit the GitHub Repository[1] I have set up for this project. It contains the plots as well (even though I have added most plots here also). Few pointers:

- Use the provided `environment.yml` file to make the environment. Code is in `./src`.
- The plots are in the folder `./src/outputs` once you `cd` inside the repository.
- There are two folder: `./src/outputs/ORIGINAL_HMM_ARCHITECTURE` (contains the 2 and 4 state standard initialization plots), `./src/outputs/ALTERNATE_HMM_ARCHITECTURE` (contains 2 state alternate initialization plot).
- The comparison plot is in the base path `./src/outputs`.

2 DISCUSSION OF RESULTS

1. **Question 1**

   Part (a)

   Ans: In this experiment, as asked in the question, the code implements a 2 state HMM with defined initialization to run the Baum-Welch Algorithm.

   Why not uniform initialization: We cannot initialize the HMM to uniform initialization for all of values becuase of the following reason:

   1. **Symmetry won't let the HMM learn:** If transition and emission probabilities were completely uniform, the model would have no preference for any state transitions or emissions. This symmetry can make the learning process unstable, as the model would struggle to differentiate between states.
   2. **Could cause super slow convergence:** Slightly non-uniform initialization introduces a small bias that helps the model break symmetry early.
   3. **Numerical Stability**: Exact uniform probabilities can lead to issues in numerical precision. Small perturbations help avoid zero gradients and allow the model to update its parameters effectively.

[1] `https://github.com/Prabhav55221/520.666_Project1`

Part (b)

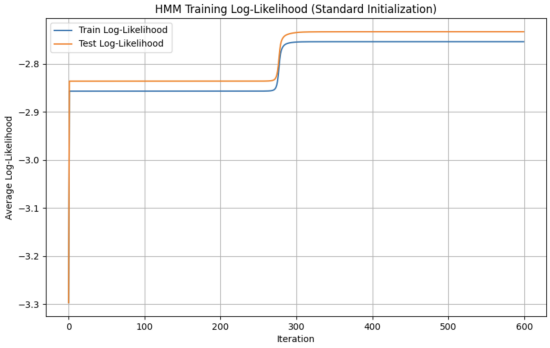
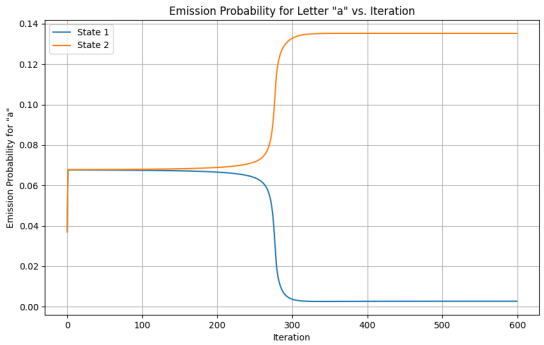


Figure 1: Standard Initialization 2 State HMM

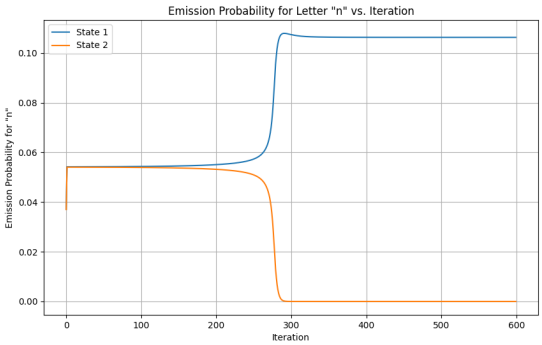
Ans: Figure 1 shows the training and test log-likelihood of **the 2-state HMM** with standard initialization over 600 iterations.

Initially, the log-likelihood starts at a lower value and quickly increases as the model learns. Around iteration 300, there is a noticeable jump, indicating a sudden improvement in model estimation, likely due to parameter adjustments or escaping a poor local optimum. After this point, both the training and test log-likelihoods stabilize, suggesting convergence. The close alignment of the two curves indicates that the model generalizes well without severe overfitting.

Part (c)



(a) Emission of 'a'



(b) Emission of 'n'

Figure 2: Emission of Letters

Ans: As shown in Figure 2, initially, both states have similar probabilities, but around iteration 300, a sharp divergence occurs, with one state dominating the emission of "a" while the other state dominates "n." Super cool to see that ***'a'*** is more probable in state 2 while ***'n'*** is more probable in state 1!

Part (d)

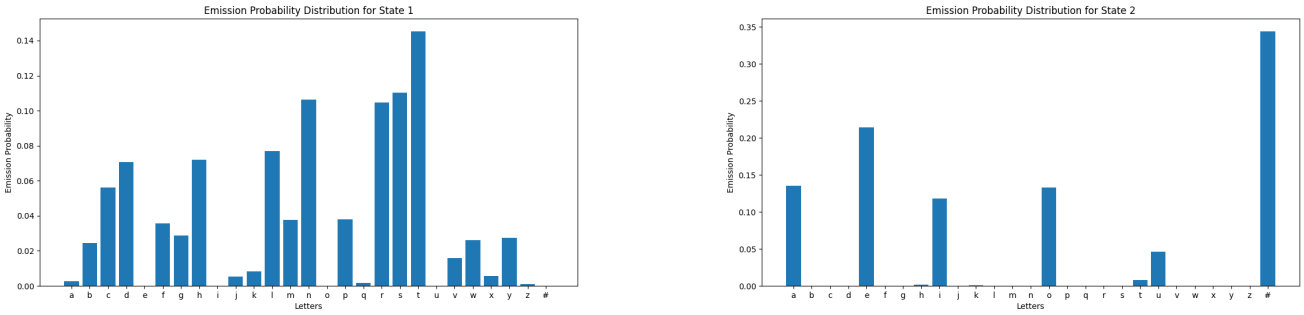


Figure 3: Emission Probability Distribution: State 1 vs State 2

Ans: The emission probability distributions shown in Figure 3 highlight the key distinctions between the two states learned by HMM. State 1 assigns relatively high probabilities to consonants such as 't', 's', 'n', and 'r', as well as a few vowels, particularly 'o' and 'i'. This suggests that State 1 is responsible for modeling the core structure of English words, where frequent consonants and vowels appear in natural sequences. The presence of high emission probabilities for these letters aligns with their expected frequency in standard English text.

In contrast, the emission probabilities for State 2, as seen in Figure are notably different. This state assigns significantly higher probabilities to vowels such as 'e' and 'a', as well as to the special character '#', which likely represents spaces or word boundaries. The strong presence of 'e' suggests that this state may be capturing common suffixes, word endings, or high-frequency transitions involving vowels. The sharp divergence in probability mass between the two states implies that the model has learned a structural distinction between the consonant-heavy body of words and the vowel-driven transitions, possibly including spaces.

2. **Question 2**

Part (a)

Ans: In this case, I used a 4-state HMM as asked. For the transition probabilities, I used the same idea (uniform with very little variation across states). They still sum up to 1 for each state as shown:

```
[[0.24, 0.26, 0.24, 0.26],
[0.26, 0.24, 0.26, 0.24],
[0.26, 0.26, 0.24, 0.24],
[0.24, 0.24, 0.26, 0.26]]
```

For emission, the even states (0, 2) get what state 0 got in the first case. While for state (1, 3), it is what state 1 got:

```
emission_prob[i, 0:13] = 0.0370
emission_prob[i, 13:26] = 0.0371
emission_prob[i, 0:13] = 0.0371
emission_prob[i, 13:26] = 0.0370
```

Space symbol still gets `emission_prob[i, 26] = 0.0367`.

Part (b)

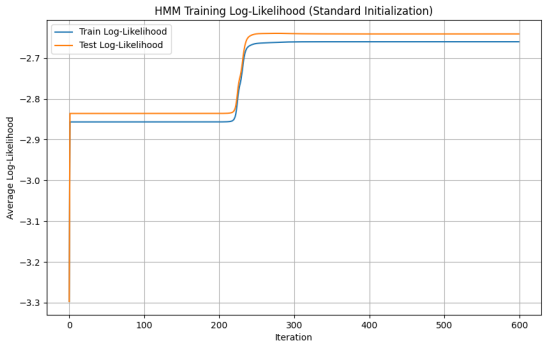


Figure 4: Standard Initialization 4 State HMM

Ans: The 4-state HMM in Figure 4 achieves **a slightly higher final log-likelihood** than 2-state HMM, indicating that it is able to model the data more effectively. Also, we see that it rise in LL faster (around 210 compared to 290th iteration). The gap between train and test log-likelihood remains small in both cases, implying that neither model is significantly overfitting.

Part (c)

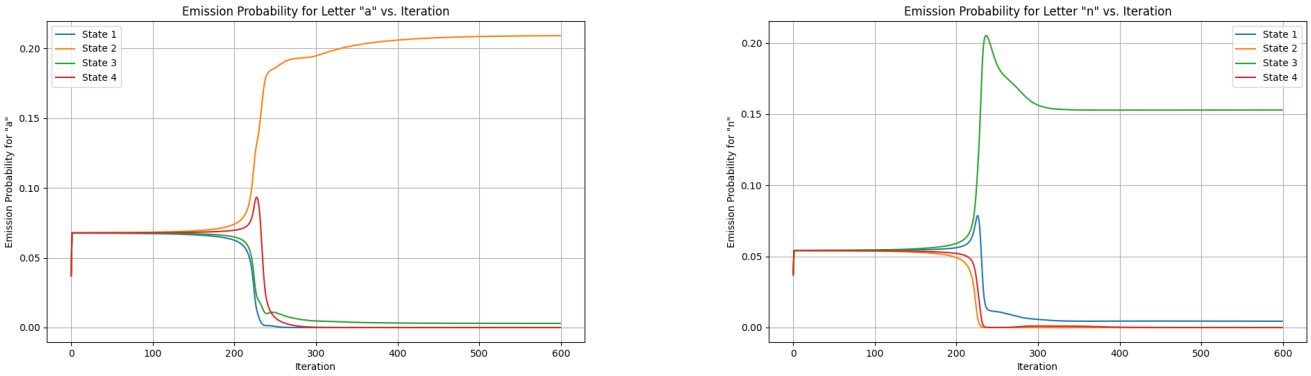


Figure 5: Emission of Letters 'a' and 'n'

Part (d)

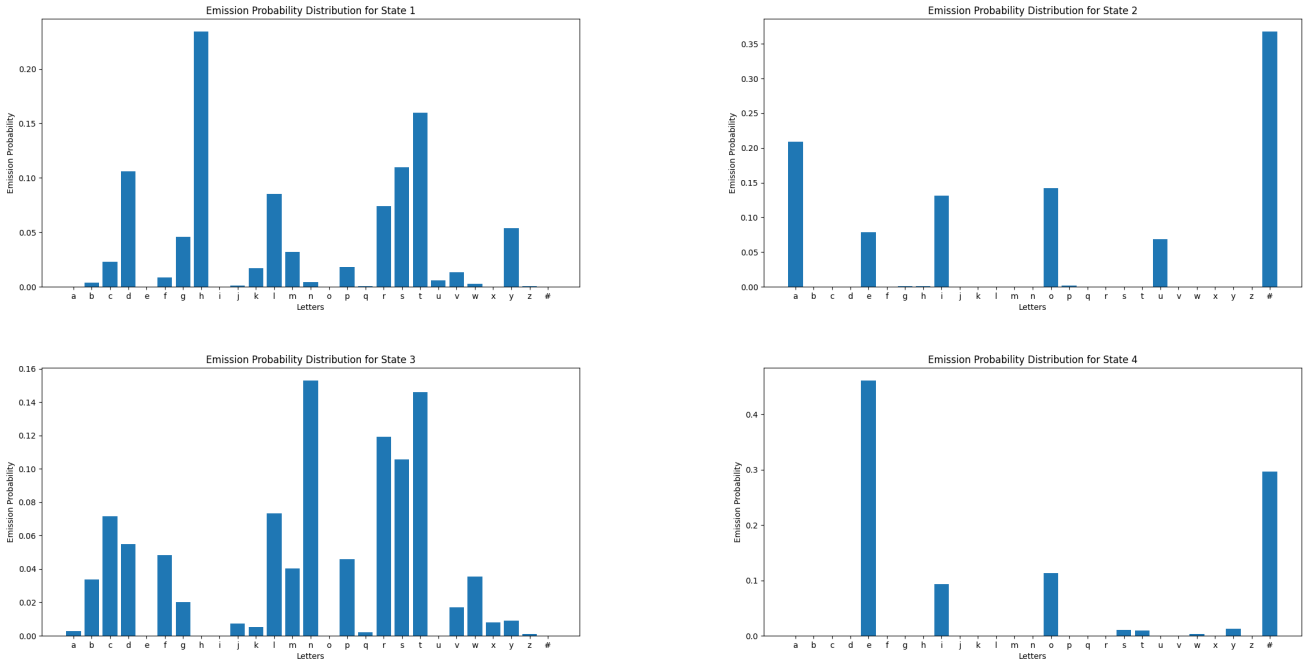


Figure 6: Emission Probability Distribution: States (1 vs 2 vs 3 vs 4)

Ans: The 4-state HMM learns a more fine-grained segmentation of letter distributions compared to the 2-state model. Each state specializes in a distinct subset of characters, with some states favoring common letters while others emphasize less frequent ones or punctuation markers like #. The emission probabilities indicate that certain states are responsible for structured linguistic components, such as

vowels, consonants, or specific phonetic groupings, rather than a simple binary split as in the 2-state model. This finer categorization suggests that the HMM is capturing underlying substructures in English text, potentially distinguishing between different linguistic contexts or stylistic variations.

3. **Question 3**

Part (a,b,c)

Ans: Refer to the code for a, b, c! For the question in the note:

The probability distributions $q(\cdot|①)$ and $q(\cdot|②)$ remain valid probability assignments because they are constructed by perturbing the base distribution $q(y)$ with a zero-mean adjustment $\delta(y)$. Since $\delta(y)$ is centered around its mean, the overall sum of probabilities remains normalized:

$$\sum_{y\in\mathcal{Y}} q(y|①) = \sum_{y\in\mathcal{Y}} (q(y) - \lambda\delta(y)) = 1 - \lambda \sum_{y\in\mathcal{Y}} \delta(y) = 1.$$

Similarly,

$$\sum_{y\in\mathcal{Y}} q(y|②) = \sum_{y\in\mathcal{Y}} (q(y) + \lambda\delta(y)) = 1 + \lambda \sum_{y\in\mathcal{Y}} \delta(y) = 1.$$

Additionally, choosing $\lambda > 0$ ensures that the adjustments do not violate the non-negativity constraint, keeping all probabilities strictly greater than zero:

$$q(y|①) = q(y) - \lambda\delta(y) > 0, \quad q(y|②) = q(y) + \lambda\delta(y) > 0, \quad \forall y \in \mathcal{Y}.$$

This guarantees that both perturbed distributions still satisfy the fundamental requirements of probability distributions: non-negativity and summation to one.

Plot: See Figure 7 for the plot. This method converges much faster! However, the Log Likelihood does not change much.

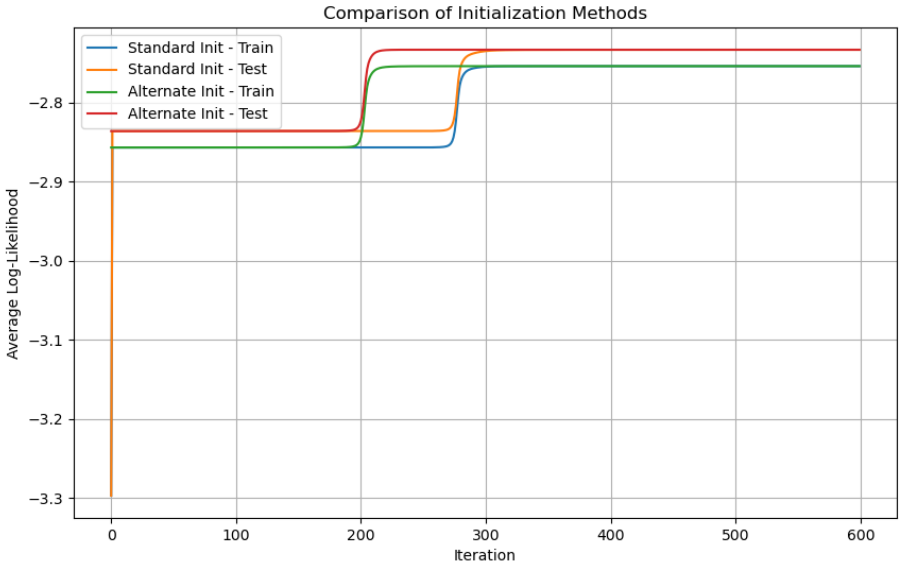


Figure 7: Comparison of Initialization