CSC 246 Project 03 Report

Qingjie Lu, qlu7

Haoqi Zhang, hzhang84

Section 1. Collaboration Statement

1. Both of us agree that we contributed an equal amount of effort in the project.
2. Both of us agree that we should receive the same grade on the project.
3. Both of us promise that we did not give or receive any unauthorized help on this project, and that all work will be our own.
4. Qingjie Lu's Contribution: Initialization, Optimization, Log Likelihood Implementation, File Loading, File Testing, Accuracy Analysis, Theory Understanding, Configuration, **REPORT**.
5. Haoqi Zhang's Contribution: EM Algorithm Implementation, Parameters Debugging, Graph Plotting, Graph Analysis, README, Normalization, NLP Util Modification, **REPORT**,

Section 2. Important User Messages

1. See README.txt for more information about how my program works and how to save it.
2. Some files have been saved during our experiments, which you can check for accuracy.
3. For optional flourishes: we finished (d) and (f) ***(We have two models!)***. We have discussed (a), (b), (c), (e) inside the group, but will not be presented here for the sake of time.
4. We believe perplexity metric is optional, so we only calculated accuracy. We think it is very representative in evaluating the quality of our hidden Markov Model.
5. Our code is properly commented. Unused functions have been commented out including word-based model and complete_sequence. We are using other functions that have similar functionalities that can assist the model from loading to training to testing.
6. Our experiments are summarized below (more experiments are tested by our own):

Experiments	Hidden States	Training Sample Size	Iterations to Converge	Time Taken for 1 EM	Accuracy

#1	3	200	Pos = **3**, Neg = **5**	12 seconds	73.12%
#2	3	500	Pos = **7**, Neg = **3**	32 seconds	79.68%
#3	3	1,000	Pos = **7**, Neg = **3**	58 seconds	81.92%
#4	3	2,000	Pos = **3**, Neg = **3**	110 seconds	80.50%
#5	3	4,000	Pos = **9**, Neg = **4**	230 seconds	82.54%
#6	3	8,000	Pos = **3**, Neg = **3**	493 seconds	84.05%
#7	3	12,500	Pos = **4**, Neg = **5**	798 seconds	83.97%
#8	6	200	Pos = **4**, Neg = **3**	16 seconds	73.13%
#9	6	500	Pos = **3**, Neg = **5**	47 seconds	79.67%
#10	6	1,000	Pos = **3**, Neg = **3**	84 seconds	81.93%
#11	6	2,000	Pos = **3,** Neg = **3**	172 seconds	80.50%
#12	6	4,000	Pos = **4**, Neg = **6**	315 seconds	82.54%
#13	6	8,000	Pos = **4**, Neg = **4**	671 seconds	84.05%
#14	6	12,500	*Memory Infeasible*	*Infeasible*	*Infeasible*
#15	10	200	Pos = **5**, Neg = **3**	21 seconds	73.13%
#16	10	500	Pos = **3**, Neg = **3**	63 seconds	79.67%
#17	10	1,000	Pos = **3**, Neg = **3**	112 seconds	81.93%
#18	10	2,000	Pos = **4**, Neg = **3**	260 seconds	80.50%
#19	10	4,000	Pos = **3,** Neg = **5**	428 seconds	82.54%
#20	10	8,000	Pos = **3**, Neg = **3**	945 seconds	84.04%
#21	10	12,500	*Memory Infeasible*	*Infeasible*	*Infeasible*

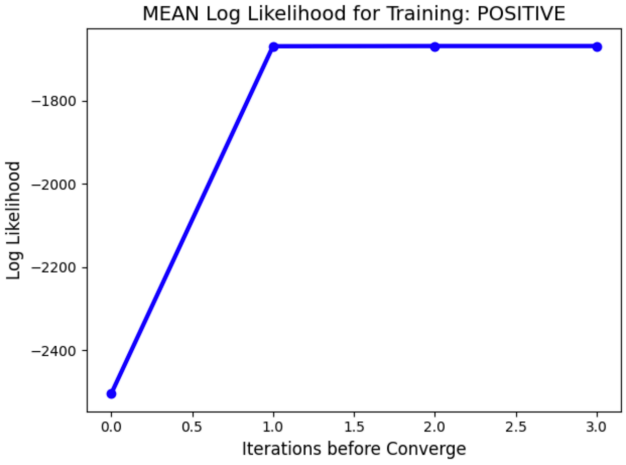
(Note: some files are dropped because of underflow or unfamiliar characters.)

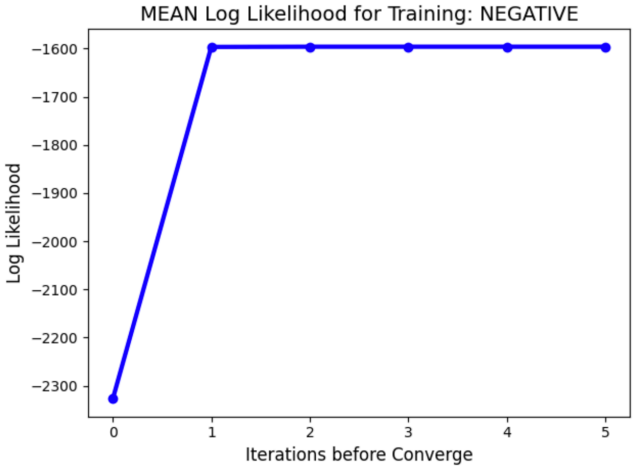
(Note: every time, different number of files will be dropped because models are different.)

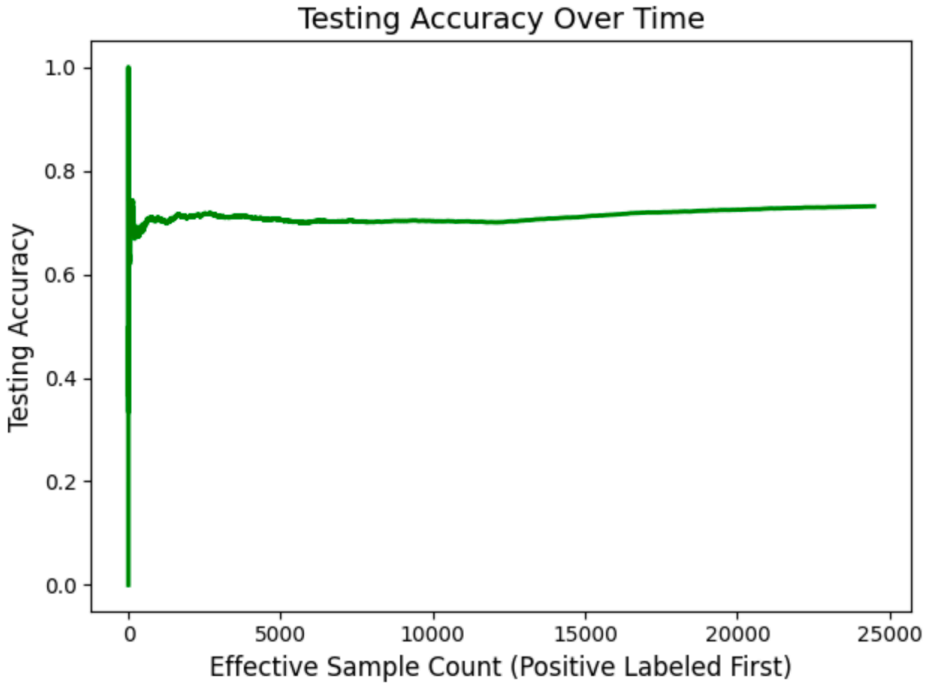
(Note: EM running time is roughly proportional with the training sample size.)

***** Section 3. Figure and Analysis *****

#1: Hidden State = 3 and Train Sample Size = 200



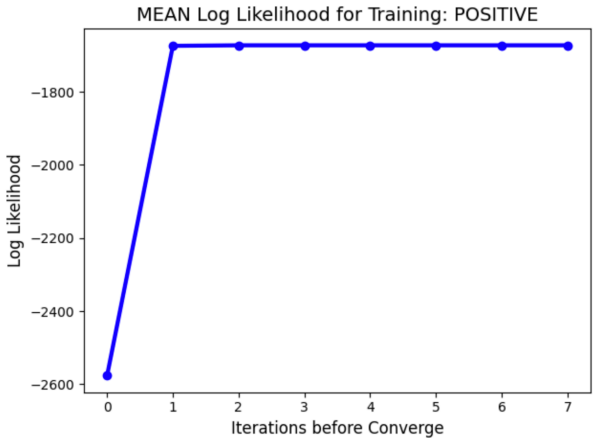


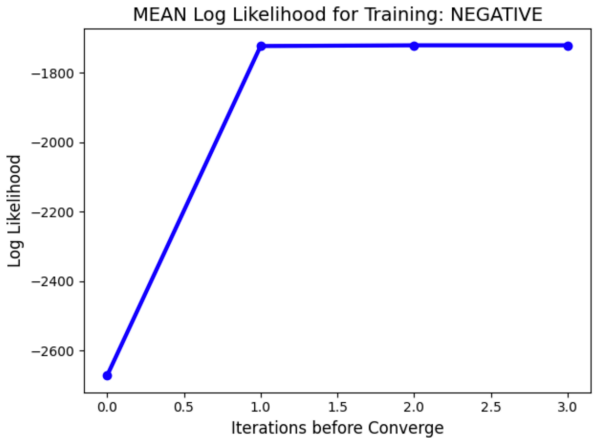


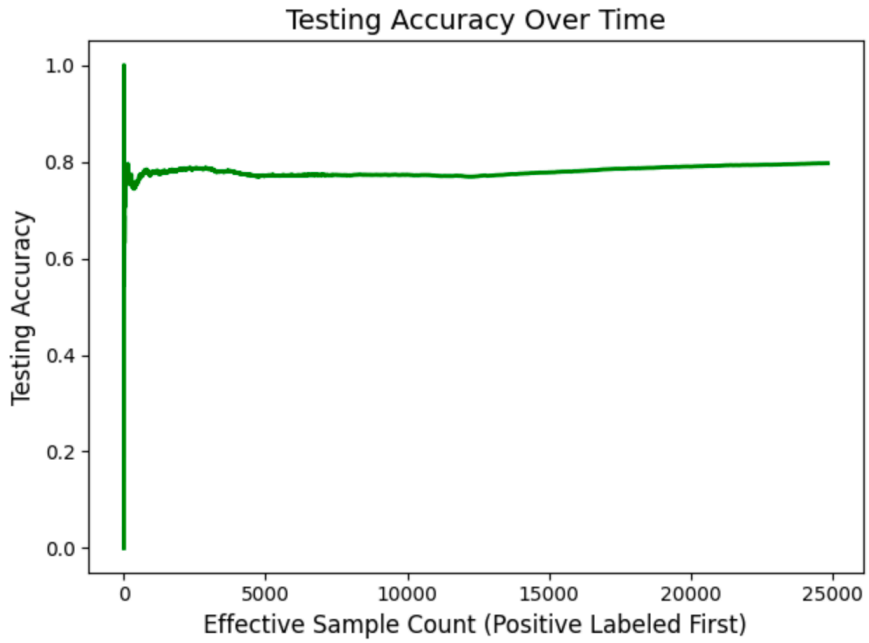

```
Total Accurate Sample: 17904
Total Effective Sample: 24484
Total Accuracy: 0.7312530632249632

Program Finishes.
```

#2: Hidden State = 3 and Train Sample Size = 500



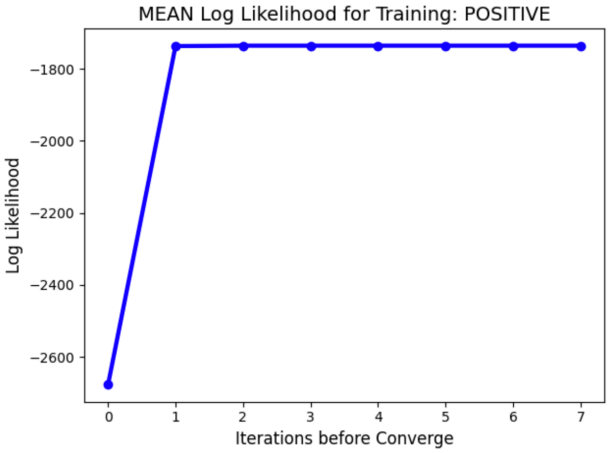


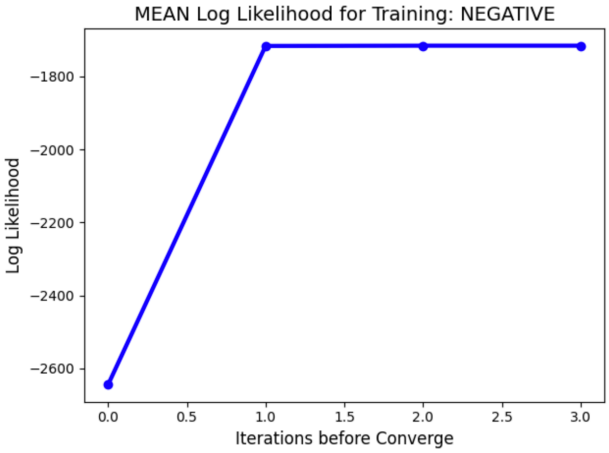


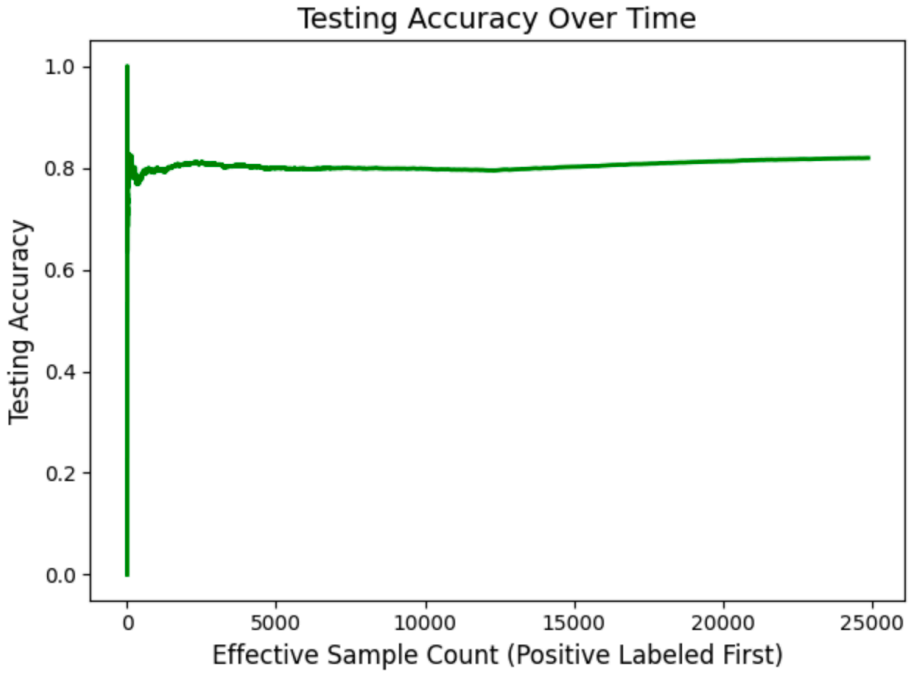

```
Total Accurate Sample: 19787
Total Effective Sample: 24834
Total Accuracy: 0.7967705564951276

Program Finishes.
```

#3: Hidden State = 3 and Train Sample Size = 1,000



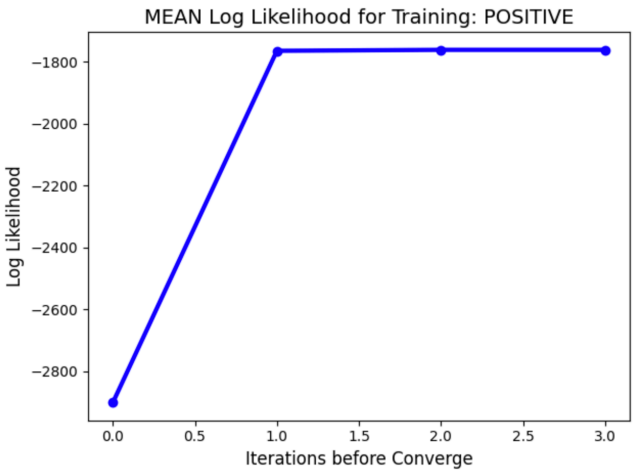


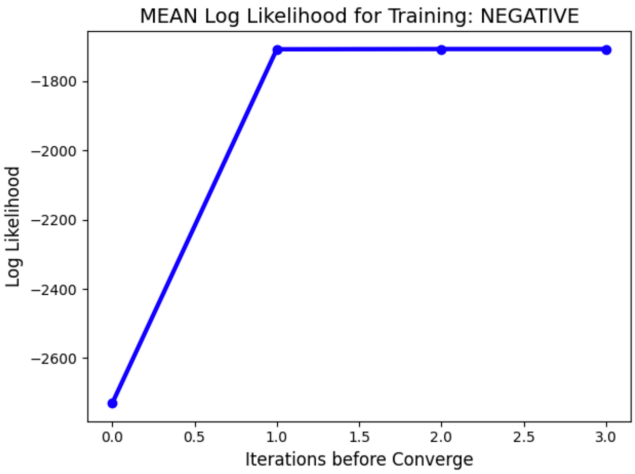


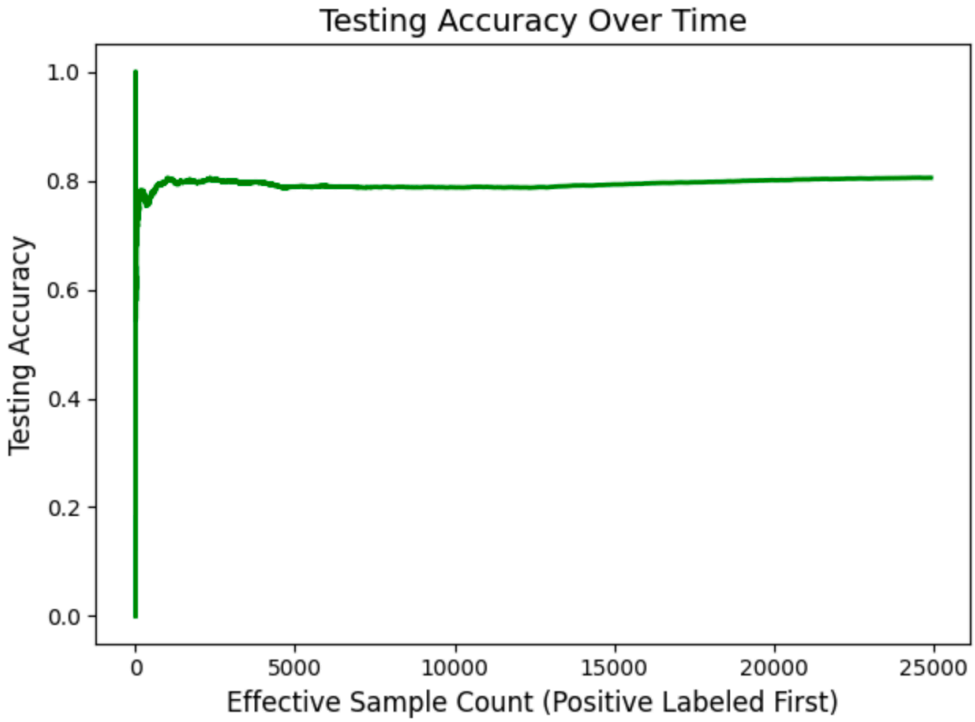

```
Total Accurate Sample: 20383
Total Effective Sample: 24880
Total Accuracy: 0.8192524115755627

Program Finishes.
```

#4: Hidden State = 3 and Train Sample Size = 2,000



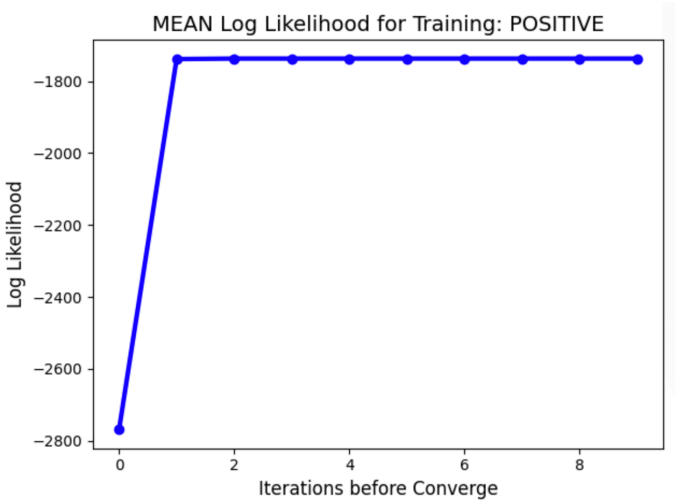


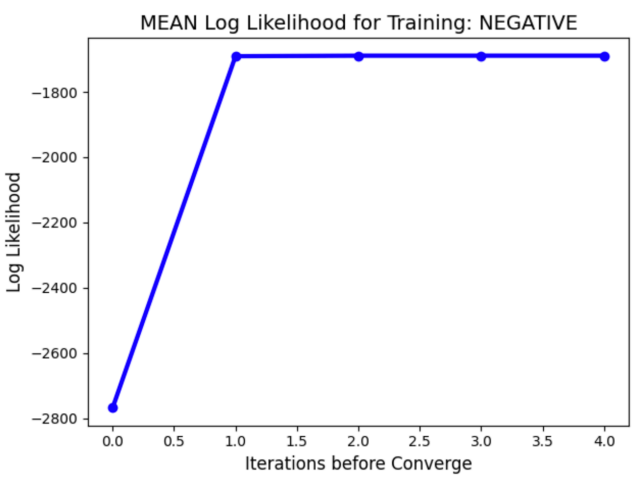


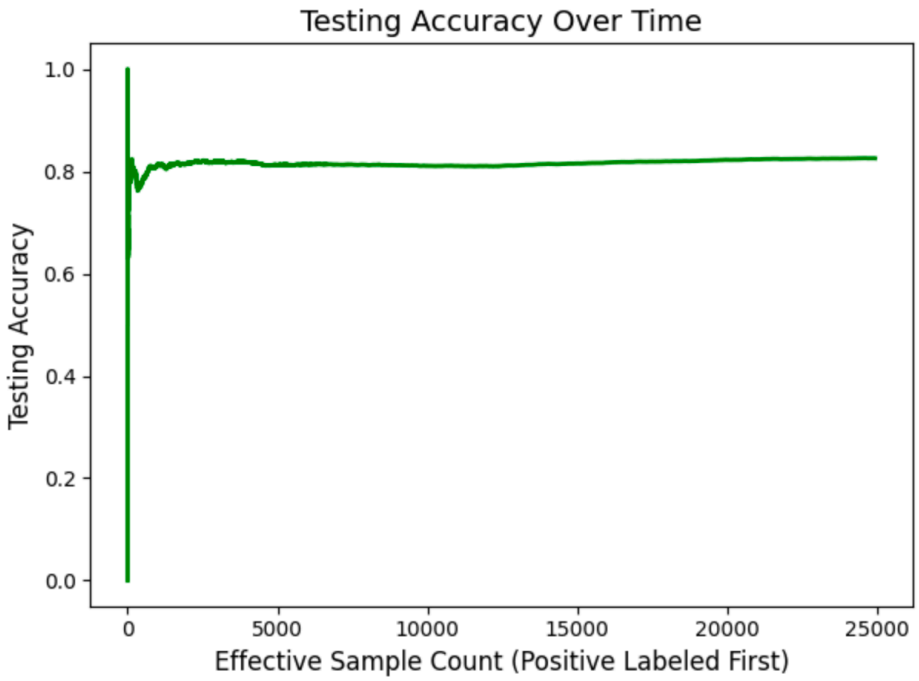

```
Total Accurate Sample: 20070
Total Effective Sample: 24932
Total Accuracy: 0.8049895716348467

Program Finishes.
```

#5: Hidden State = 3 and Train Sample Size = 4,000



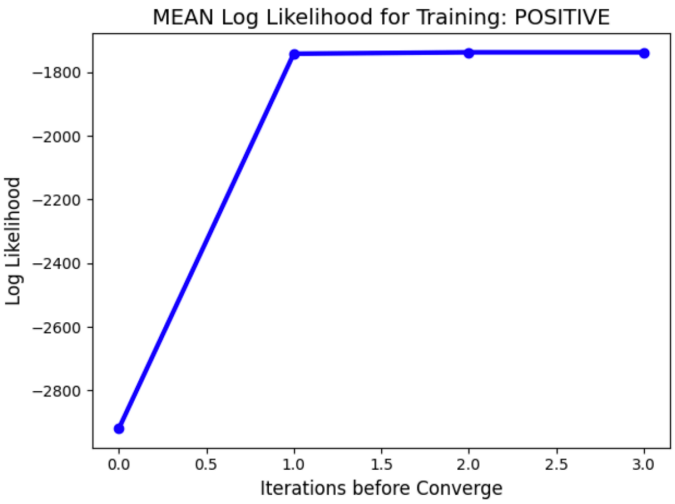


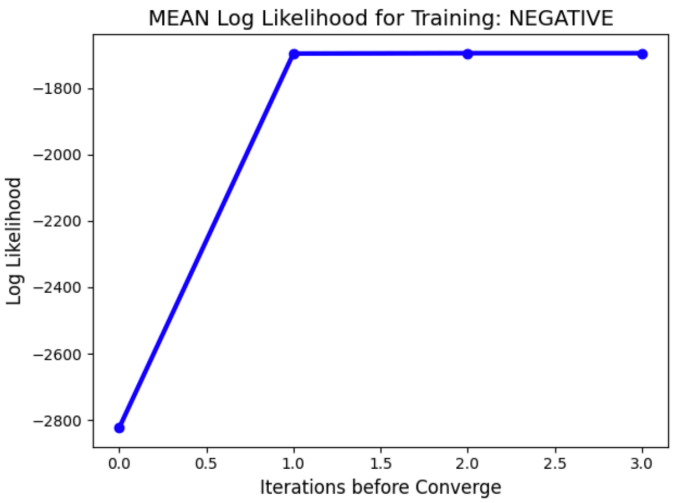


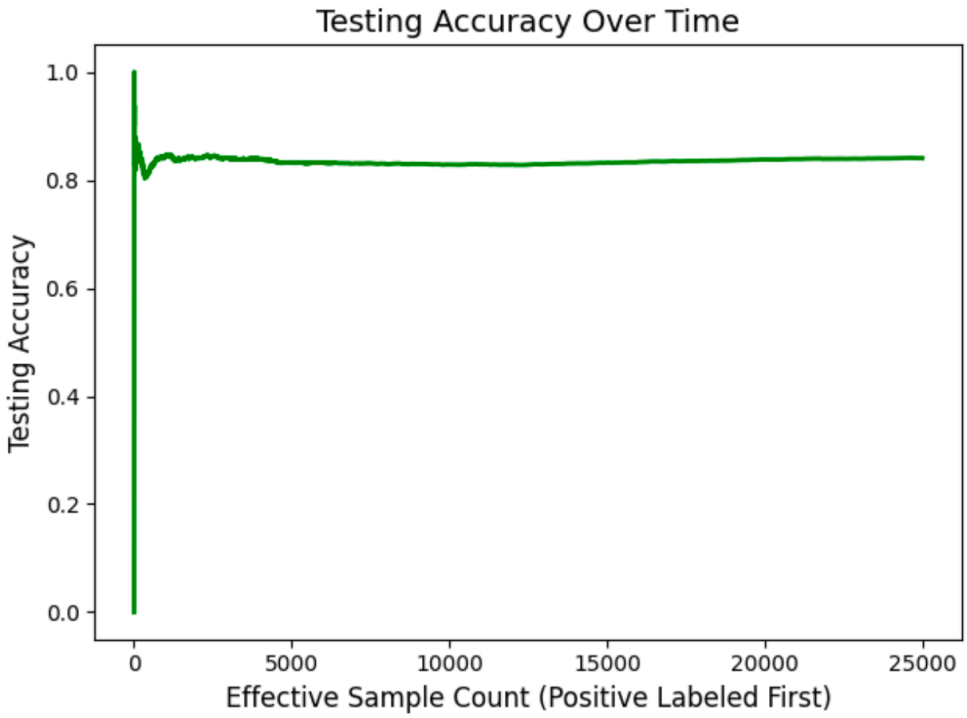

```
Total Accurate Sample: 20594
Total Effective Sample: 24950
Total Accuracy: 0.8254108216432866

Program Finishes.
```

#6: Hidden State = 3 and Train Sample Size = 8,000



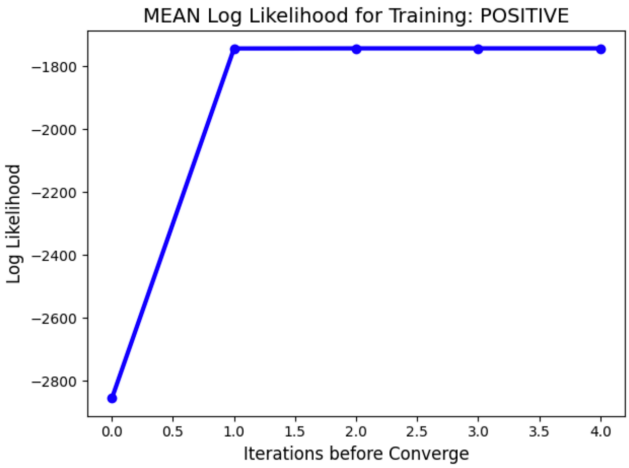


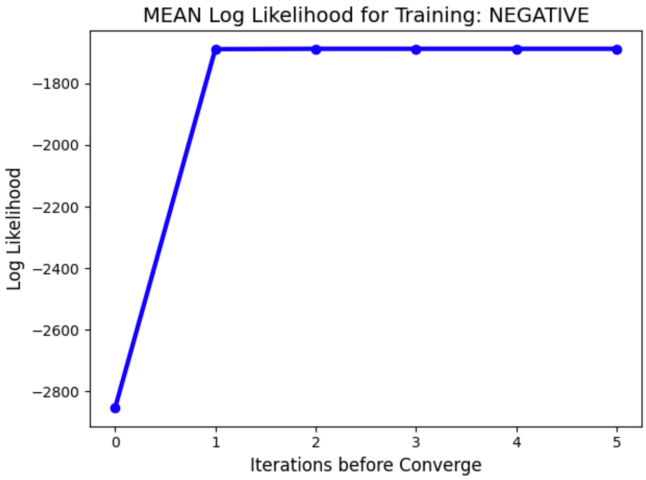


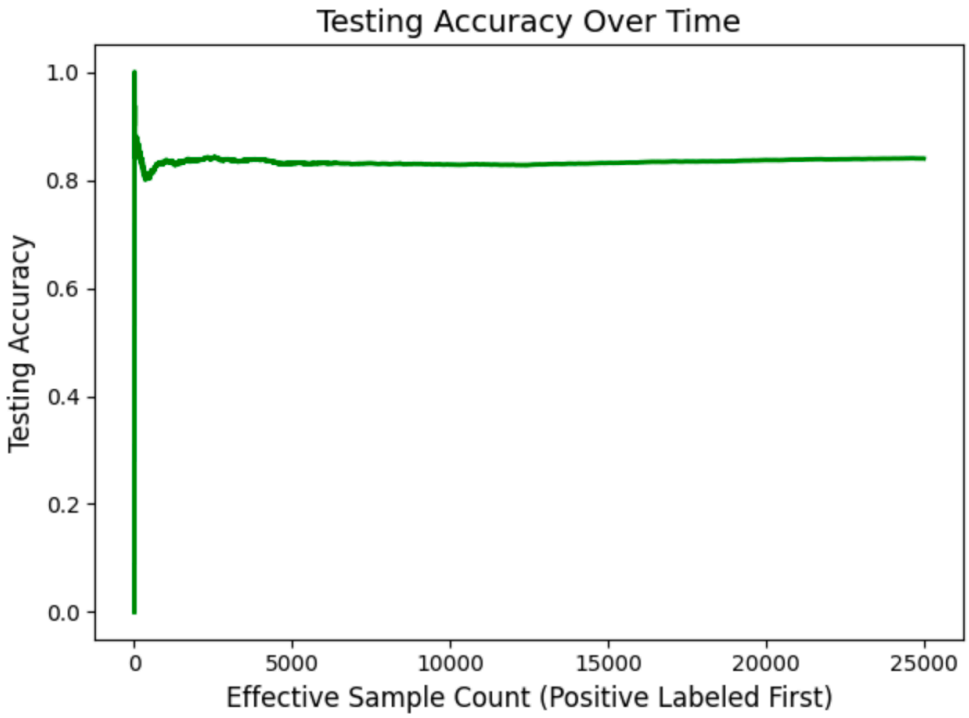

```
Total Accurate Sample: 20998
Total Effective Sample: 24984
Total Accuracy: 0.8404578930515529

Program Finishes.
```

#7: Hidden State = 3 and Train Sample Size = 12,500



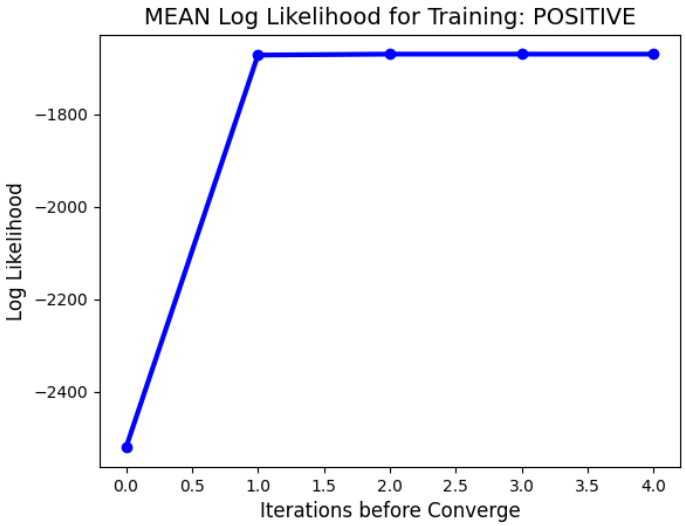


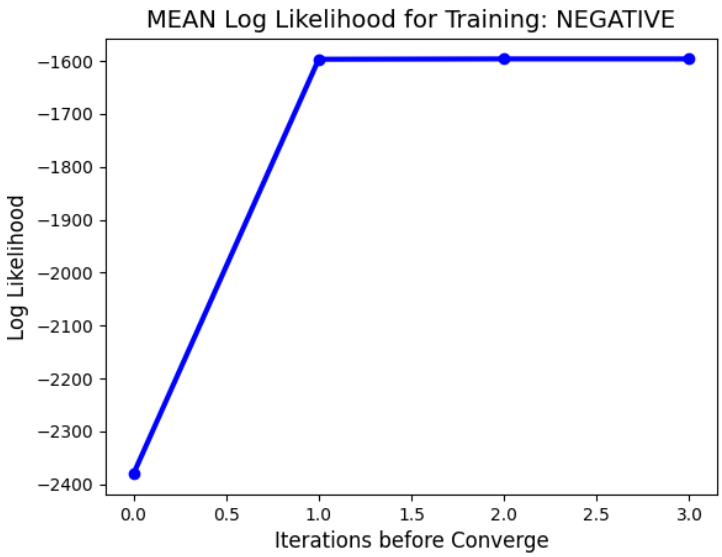


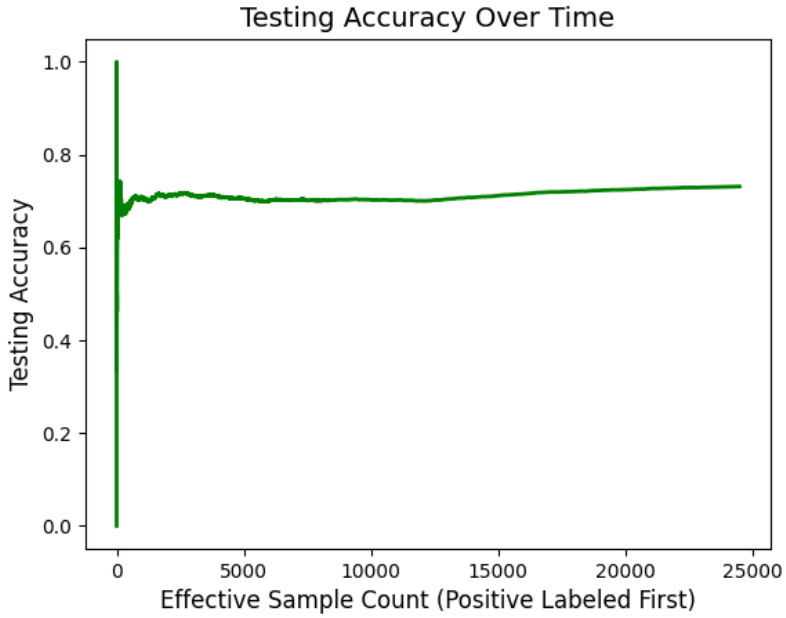

```
Total Accurate Sample: 20984
Total Effective Sample: 24990
Total Accuracy: 0.8396958783513405

Program Finishes.
```

#8: Hidden State = 6 and Train Sample Size = 200



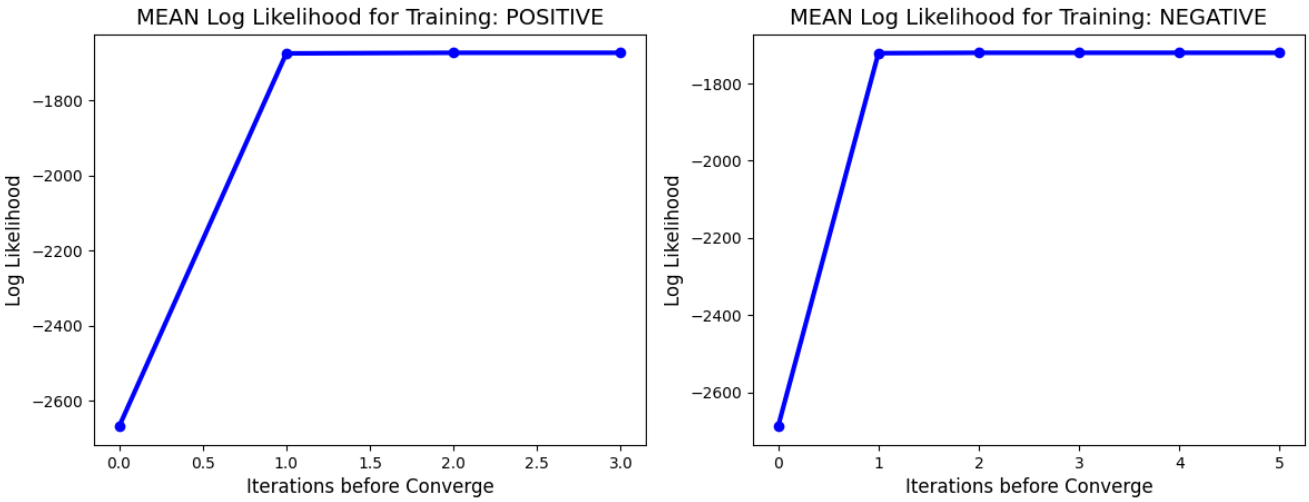


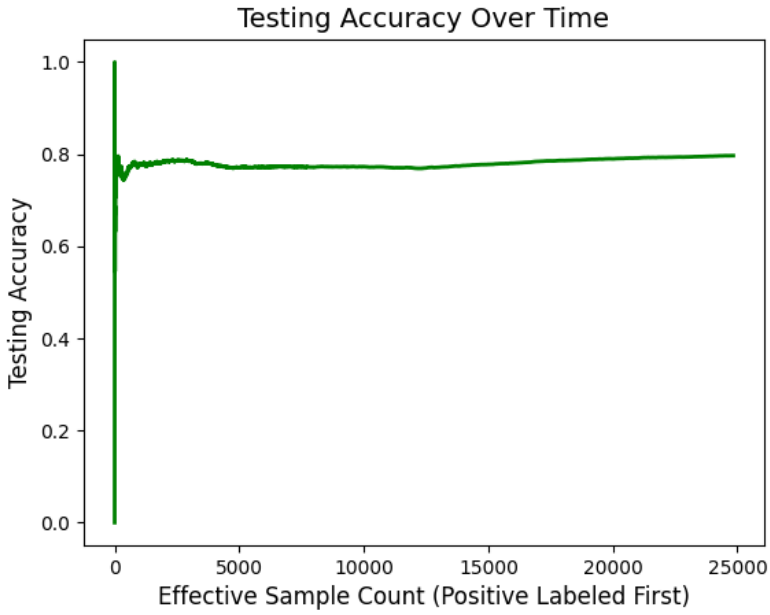



```
Total Accurate Sample: 17905
Total Effective Sample: 24485
Total Accuracy: 0.7312640392076781

Program Finishes.
```

#9: Hidden State = 6 and Train Sample Size = 500

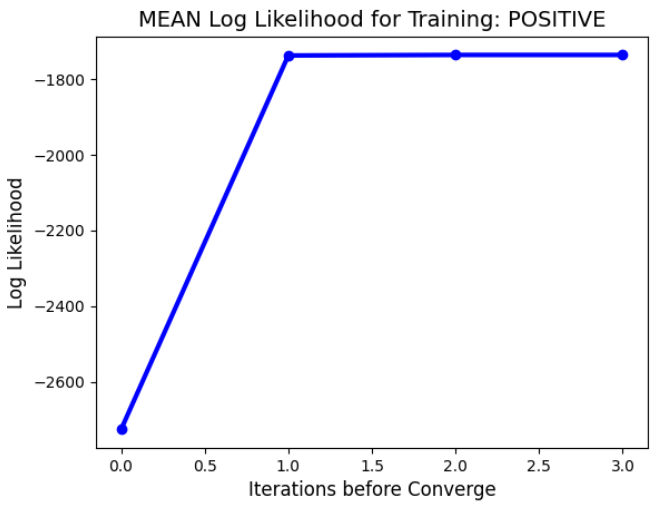


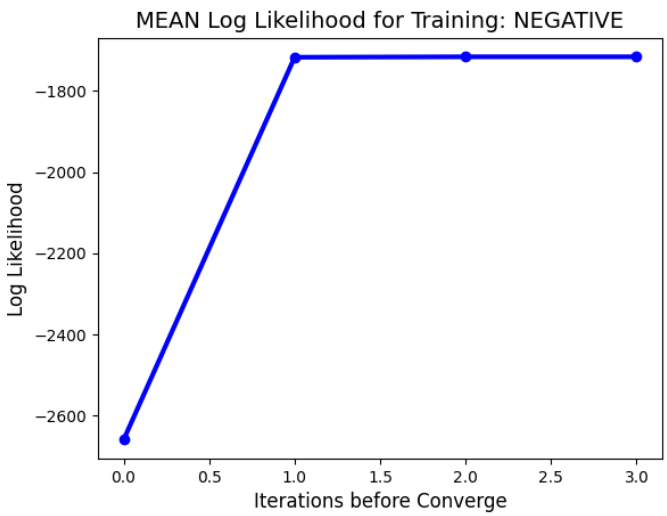


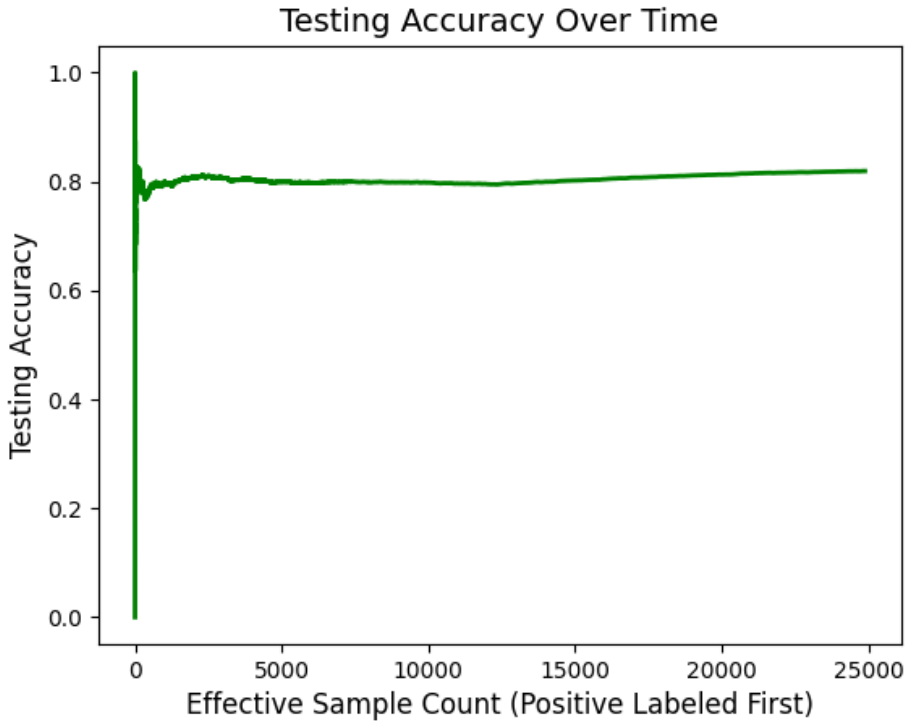

```
Total Accurate Sample: 19785
Total Effective Sample: 24834
Total Accuracy: 0.7966900217443827

Program Finishes.
```

#10: Hidden State = 6 and Train Sample Size = 1,000



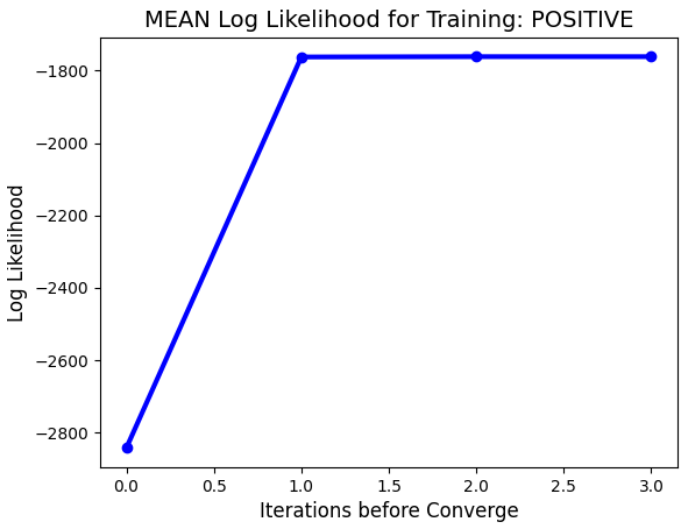


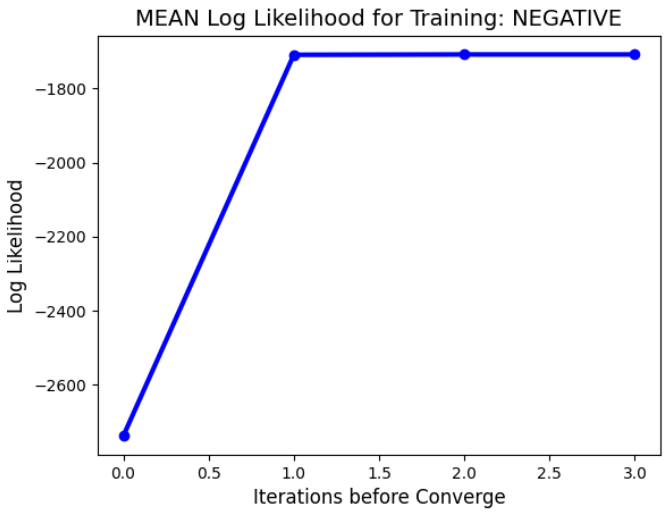


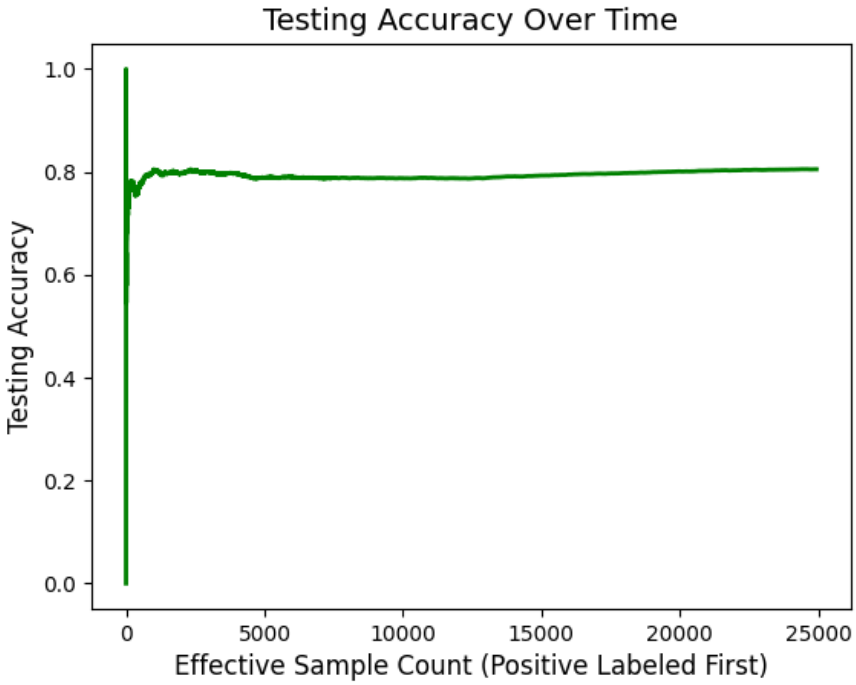

```
Total Accurate Sample: 20383
Total Effective Sample: 24880
Total Accuracy: 0.8192524115755627

Program Finishes.
```

#11: Hidden State = 6 and Train Sample Size = 2,000



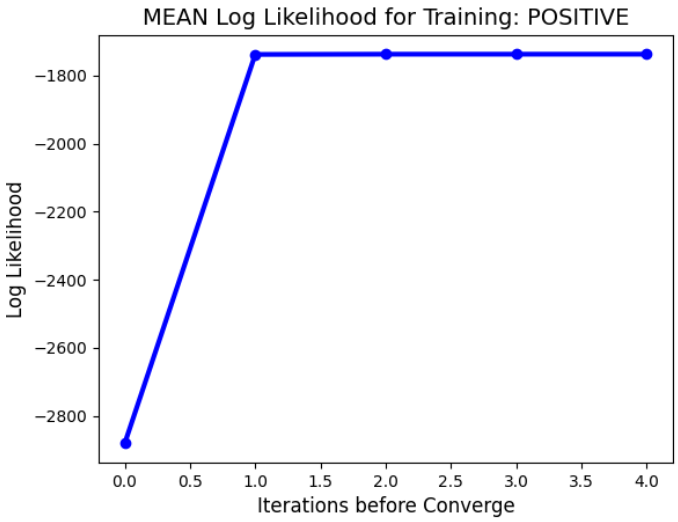


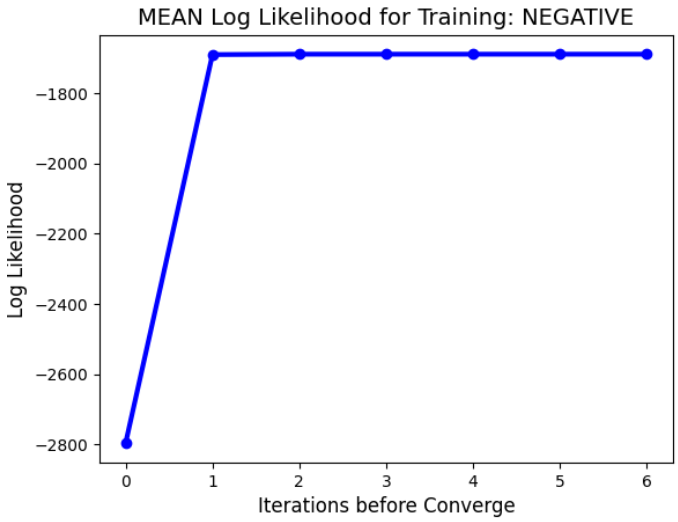


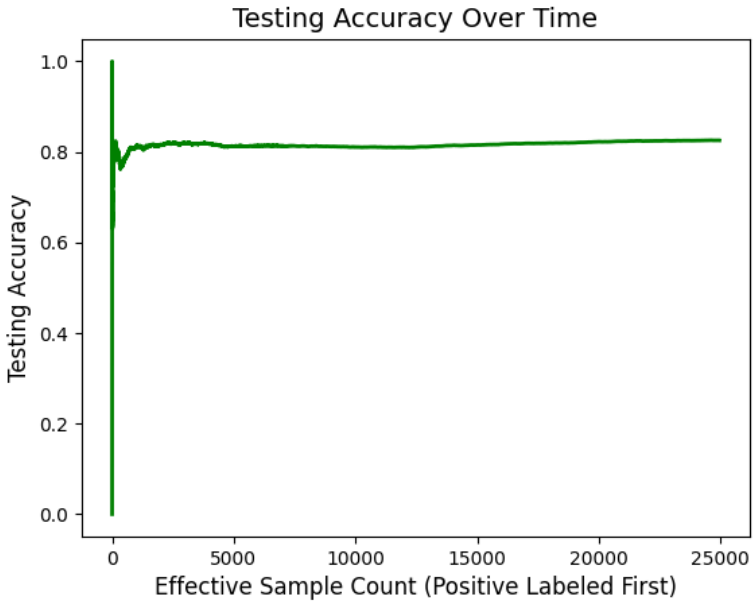

```
Total Accurate Sample: 20071
Total Effective Sample: 24932
Total Accuracy: 0.8050296807315899

Program Finishes.
```

#12: Hidden State = 6 and Train Sample Size = 4,000



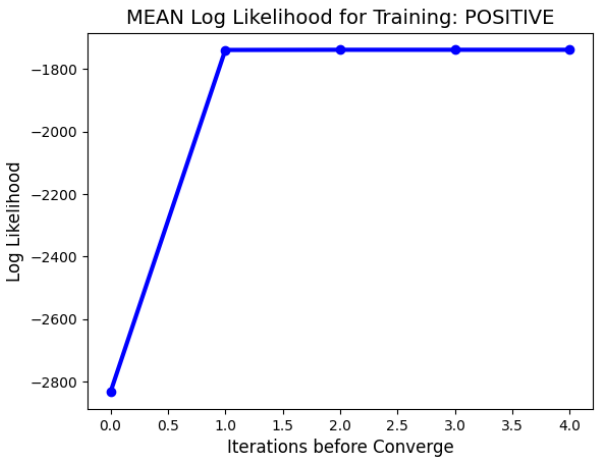


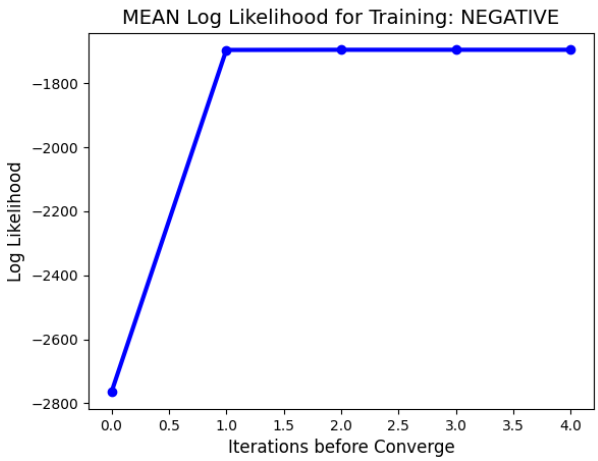


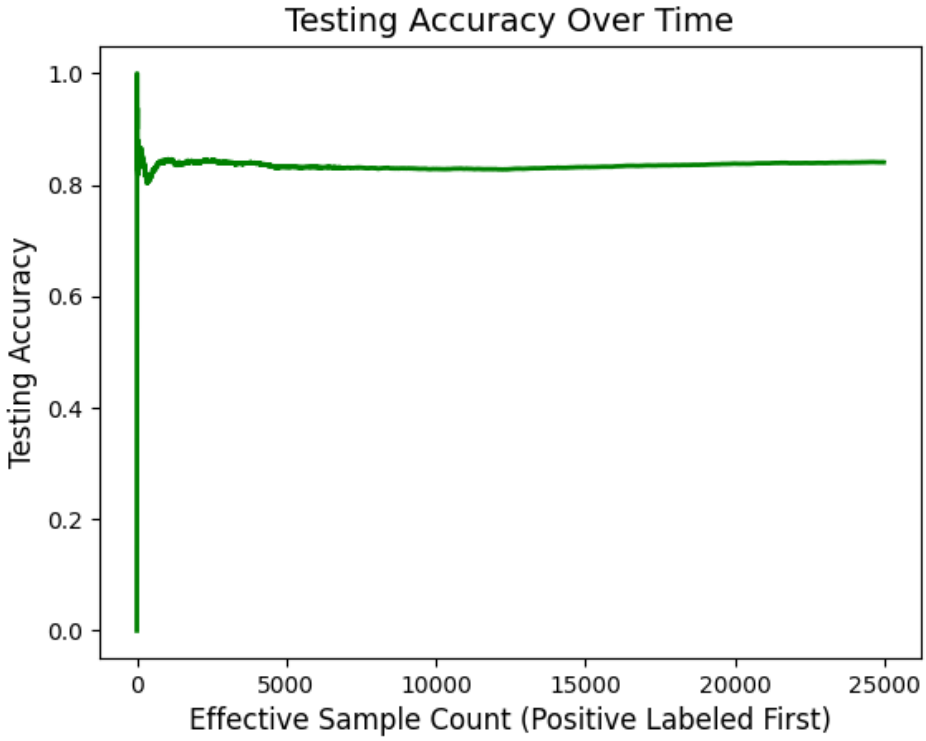

```
Total Accurate Sample: 20594
Total Effective Sample: 24950
Total Accuracy: 0.8254108216432866

Program Finishes.
```

#13: Hidden State = 6 and Train Sample Size = 8,000



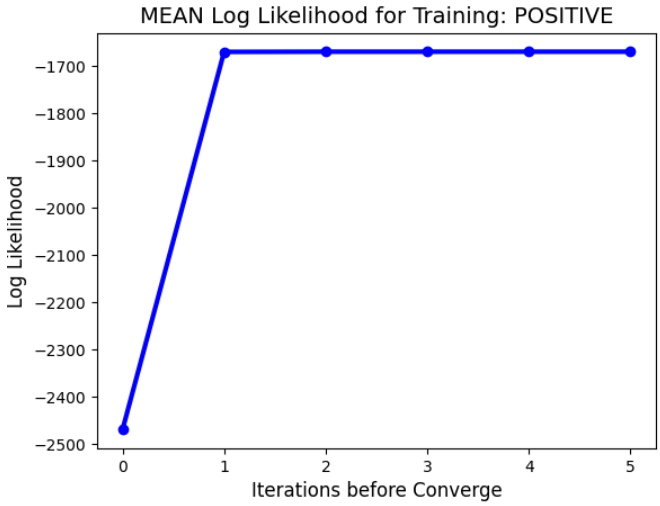


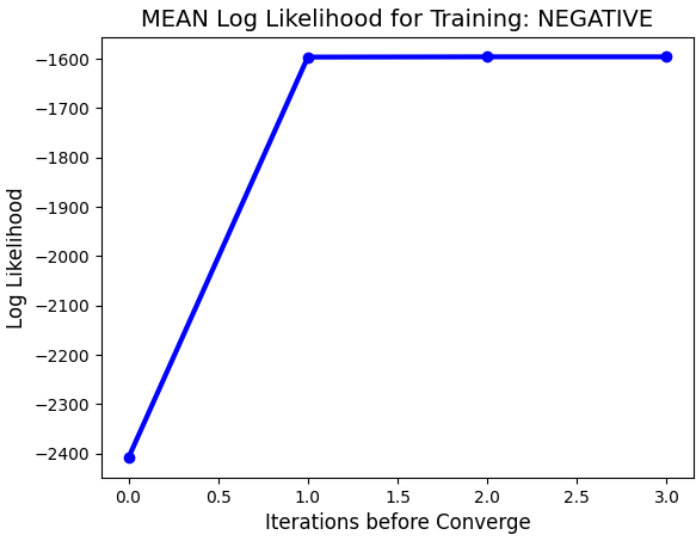


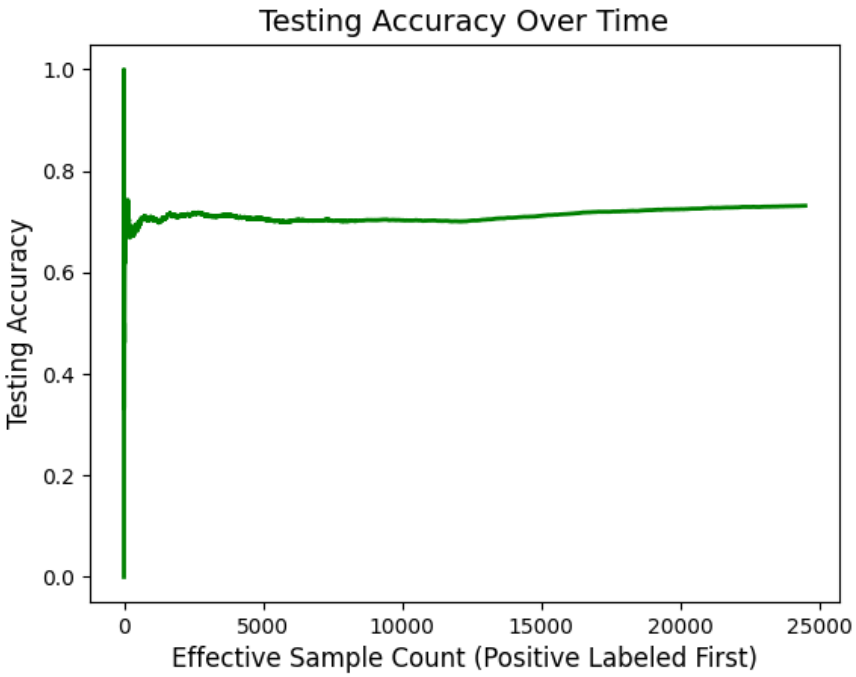

```
Total Accurate Sample: 20998
Total Effective Sample: 24984
Total Accuracy: 0.8404578930515529

Program Finishes.
```

#15: Hidden State = 10 and Train Sample Size = 200



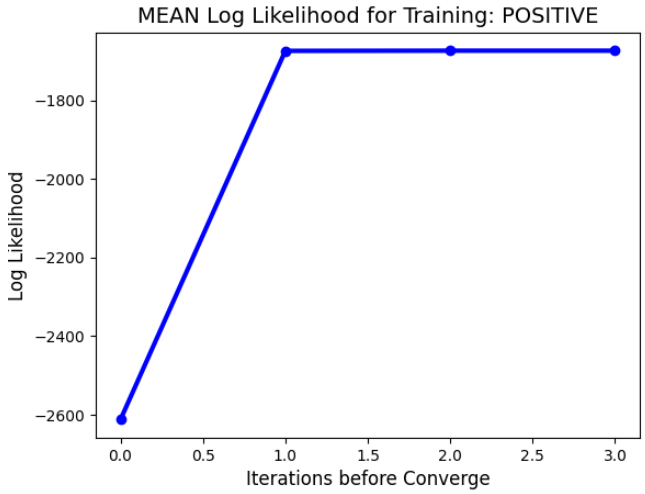


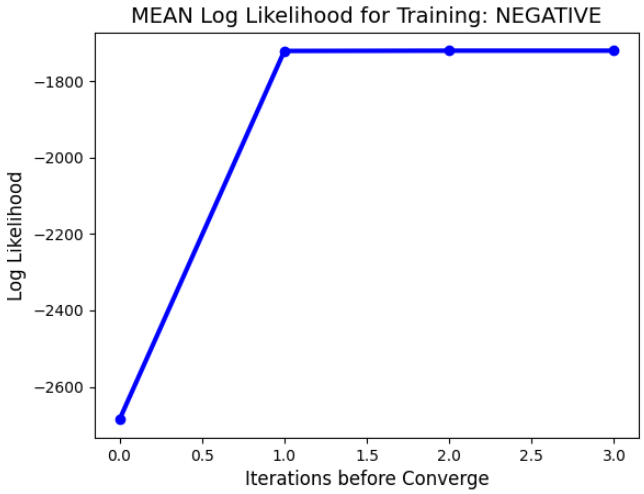


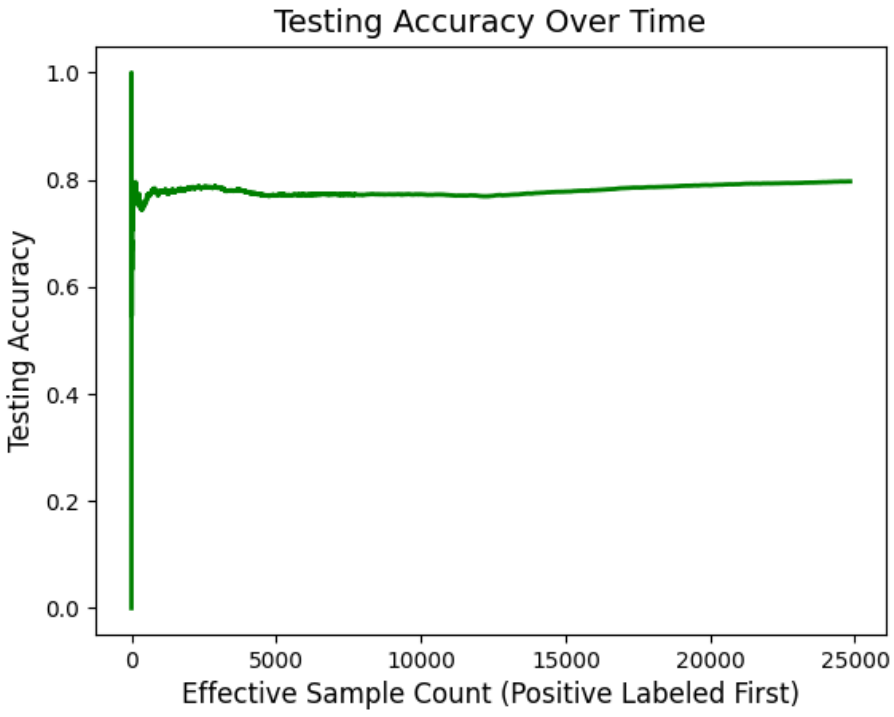

```
Total Accurate Sample: 17905
Total Effective Sample: 24485
Total Accuracy: 0.7312640392076781

Program Finishes.
```

#16: Hidden State = 10 and Train Sample Size = 500



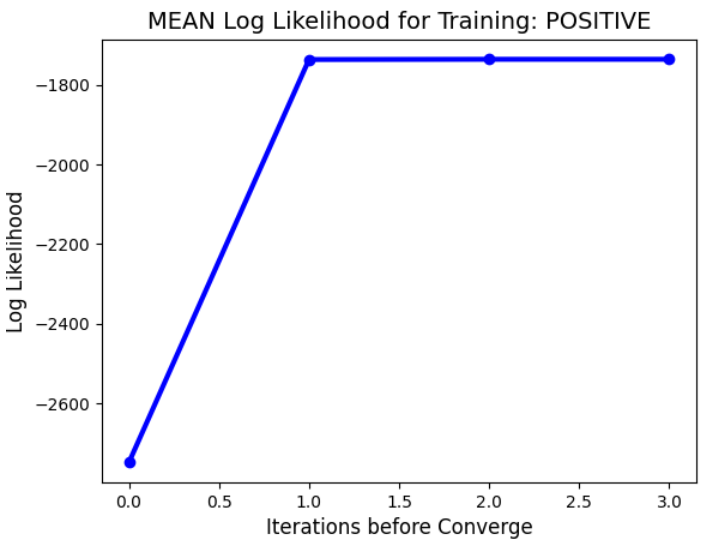


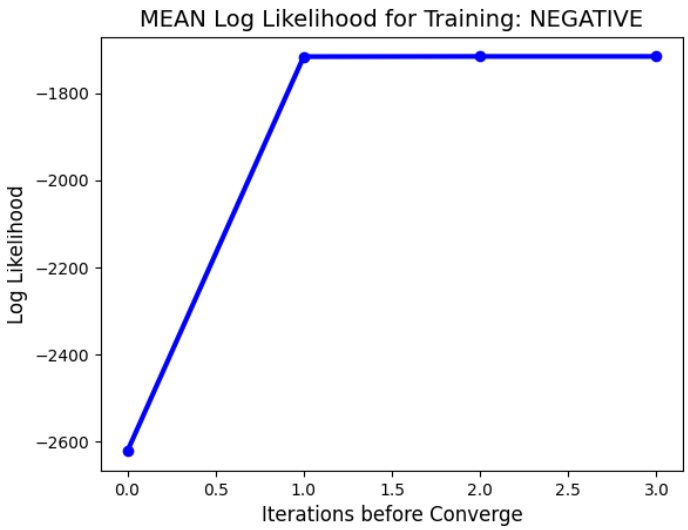


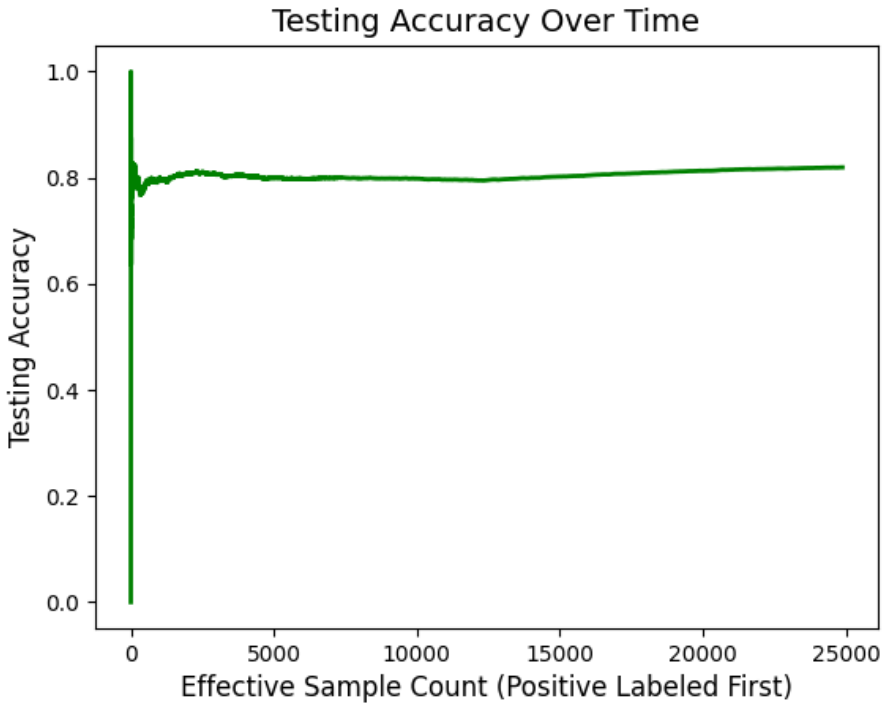

```
Total Accurate Sample: 19785
Total Effective Sample: 24834
Total Accuracy: 0.7966900217443827

Program Finishes.
```

#17: Hidden State = 10 and Train Sample Size = 1,000



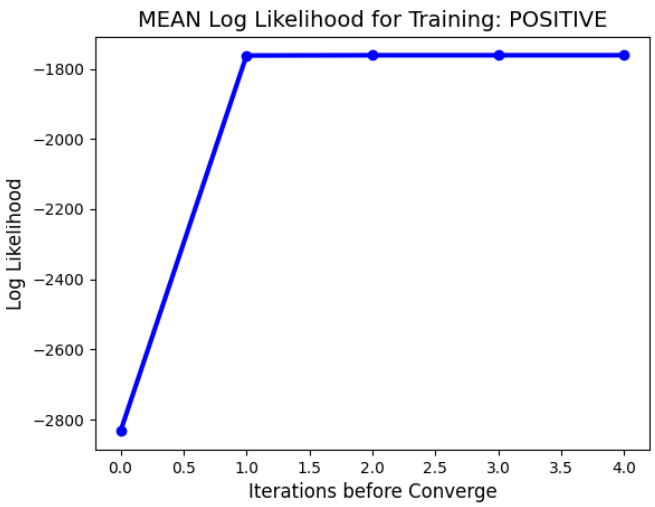


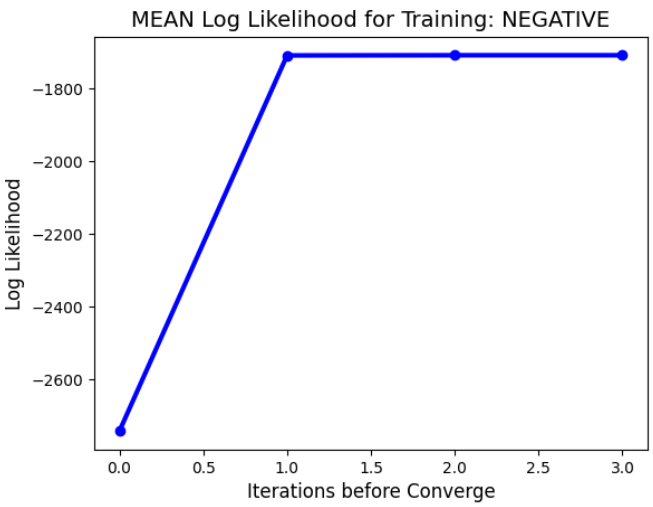


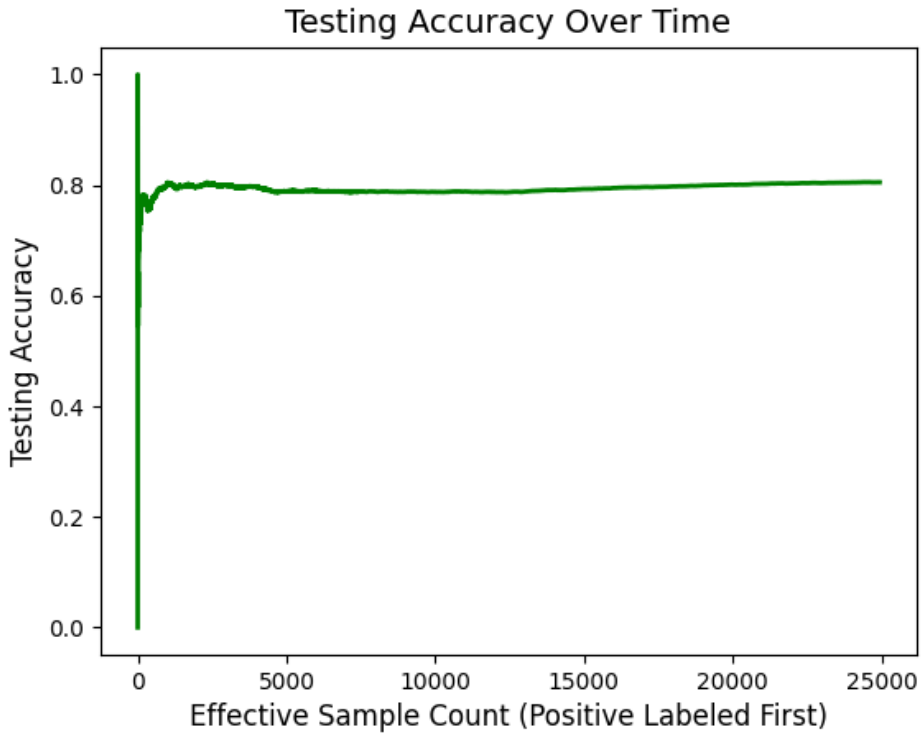

```
Total Accurate Sample: 20383
Total Effective Sample: 24880
Total Accuracy: 0.8192524115755627

Program Finishes.
```

#18: Hidden State = 10 and Train Sample Size = 2,000



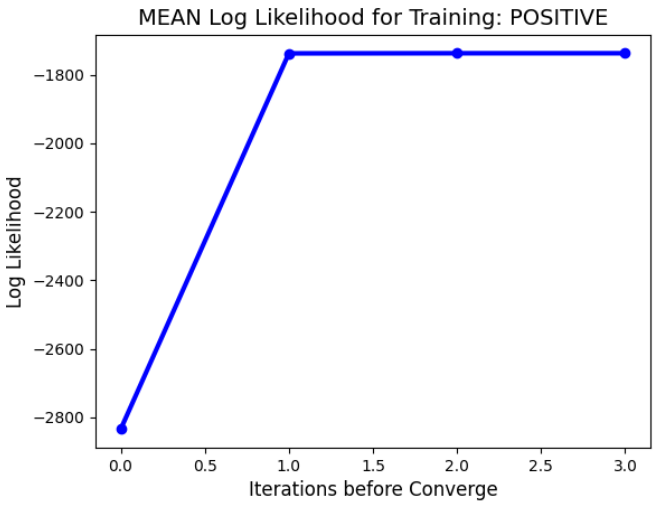


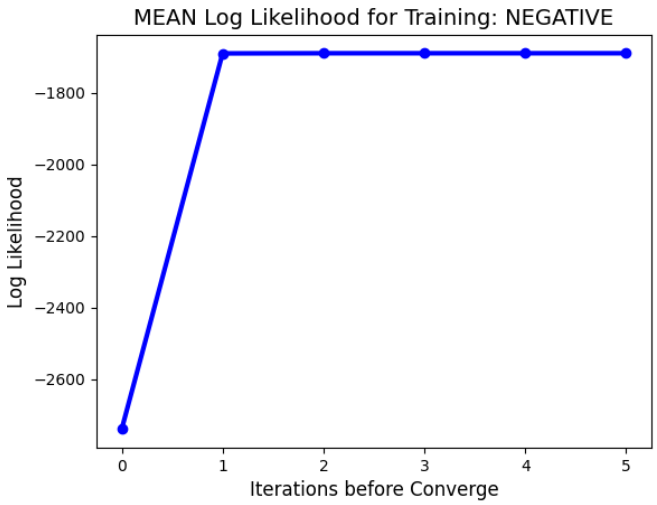


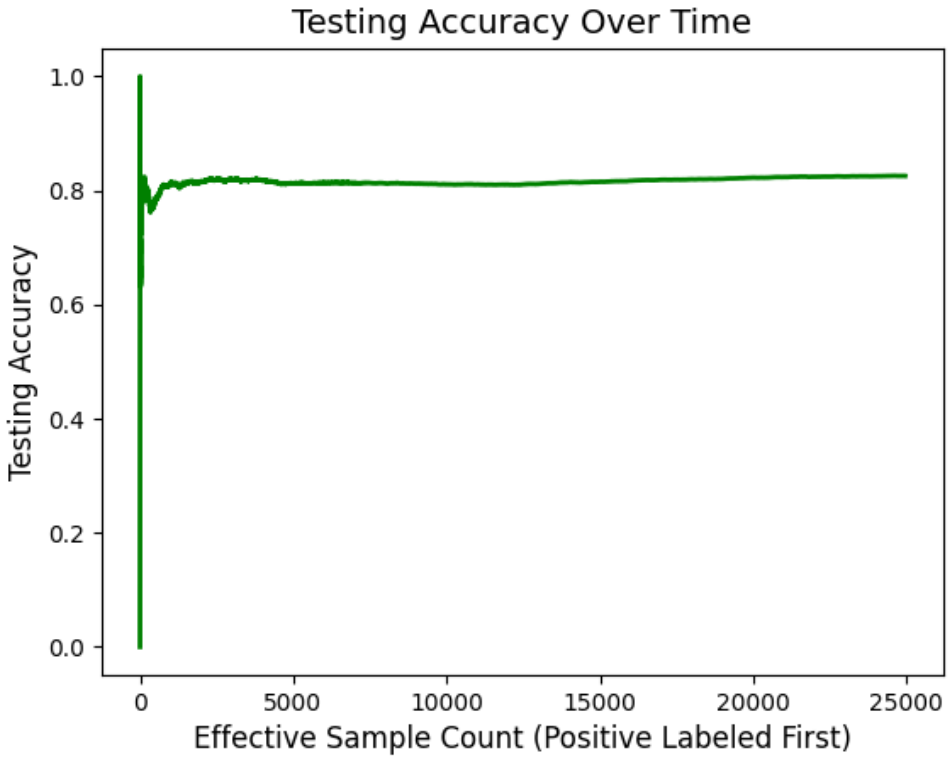

```
Total Accurate Sample: 20071
Total Effective Sample: 24932
Total Accuracy: 0.8050296807315899

Program Finishes.
```

#19: Hidden State = 10 and Train Sample Size = 4,000



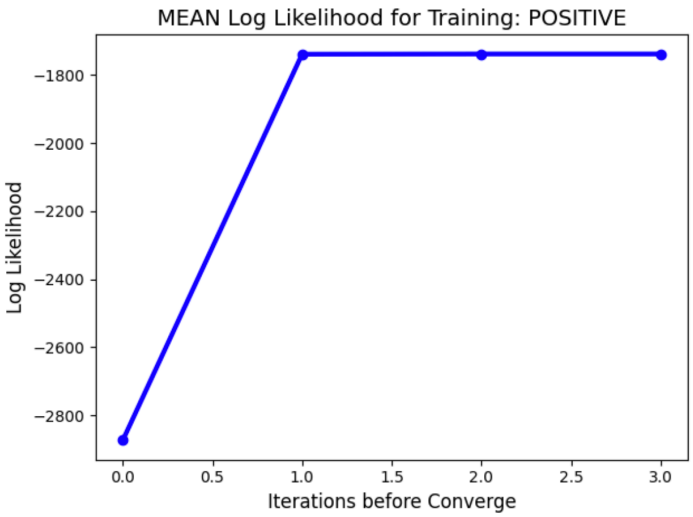


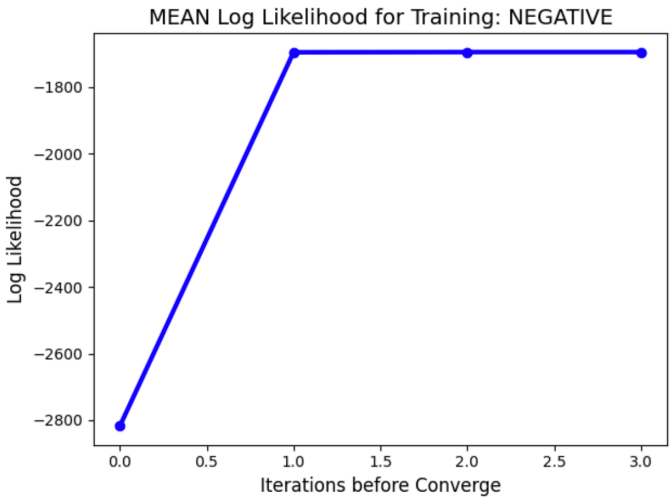


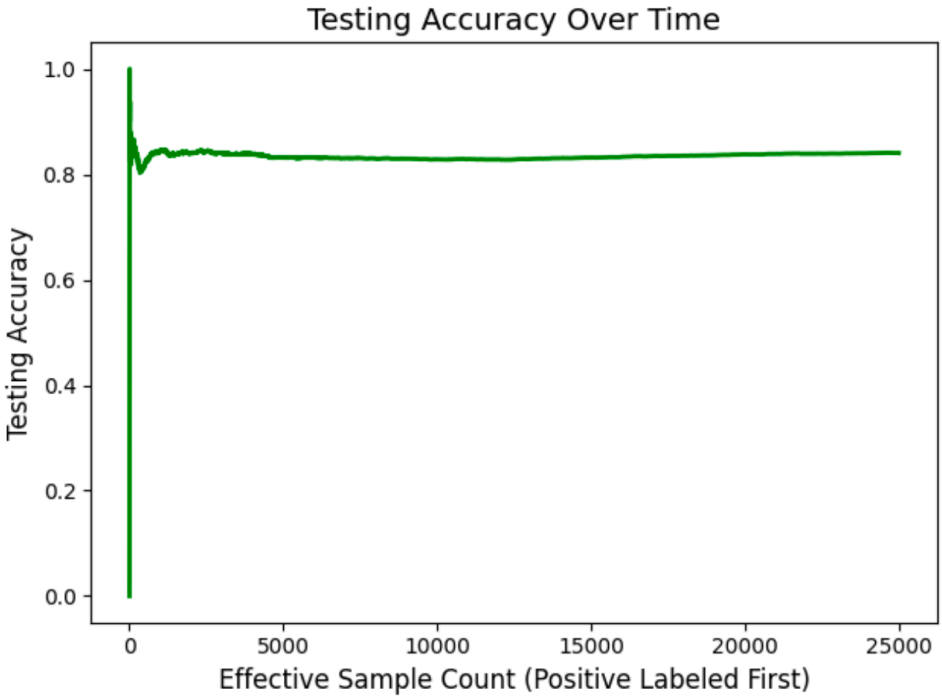

```
Total Accurate Sample: 20594
Total Effective Sample: 24950
Total Accuracy: 0.8254108216432866

Program Finishes.
```

#20: Hidden State = 10 and Train Sample Size = 8,000








```
Total Accurate Sample: 20998
Total Effective Sample: 24984
Total Accuracy: 0.8404578930515529

Program Finishes.
```

Section 4. Number of Iterations to Convergence

Most of our experiments converge in about 3 to 5 iterations. After 3 iterations, the log likelihood changes for less than 1, or even less than 0.00001. I think ε is very close to 0. It should be 10^-5 or even 10^-7. We can even set it to 0 and wait until the log likelihood to decrease. In fact, this is what our project is using, which is quite effective because the EM algorithm very rarely requires more than 7 iterations except some very bad initializations.

Section 5. Number of Hidden States

We feel 3 to 5 hidden states are the best. Confessedly, 10 hidden states will not make a big difference in accuracy, but the running time increase proportionally with the number of hidden states. Our model is having a very big difficulty in running 10,000 samples and 10 hidden states at the same time. It is taking many hours even with optimization. Instead, 1 hidden state is very fast but it has nearly the same accuracy as pure chance (we will not show here because it is meaningless). The biggest model we could obtain in under an hour is 8,000 training samples and 3 hidden states, which we believe is very fast. It is very hard to complete running anything with a few seconds unless we only use 50 samples or so; however, the accuracy is again very close to pure chance. We think the model built using 4,000 training samples and 3 hidden states are the most cost-effective one.

Section 6. Theory, Compute, and Memory Bound

We think it probably should be theory bound. First of all, memory usage for calculating log likelihood is far smaller than the EM algorithm with an initialization of the three-dimensional array. Also, we used np.longdouble during calculation, which is less prune to underflow. We can see that Python is saving many digits after the number when they are very similar. We believe Python is not suffering compute bound because it is capable of reserving many digits. Therefore, the EM algorithm makes it gradually converges in theory. However, if we are using C or other languages with less “digit protection”, then compute bound will be very likely to happen.

Section 7. Underfitting and Overfitting

There is a clear underfitting when there is just 1 hidden state. The accuracy suddenly drops from 75% to around 55%, which is very close to random chance. Also, we can even imagine that the language in human culture is a really complicated stuff. For example, there exists sarcasm, anastrophe, composite sentences, and many other stuff, which cannot be explained by a simple dimension, which is why we need more hidden states. However, it is very hard for us to test for overfitting because of the memory problem. When the model has more than 10 hidden states, it gets extremely slow and the computer is starting to run out of memory.

Section 8. Model Quality

We used the "accuracy average over time" method to calculate the accuracy, which takes average each time when we finish testing ONE sample. It starts to be unstable but gradually converges. We have to say that the hidden Markov model is more accurate than random chance but still not very good. Despite the data is complicated, it is approaching an accuracy bound at about 84% (no change from 8,000 to 12,500). It gets above 80% after 1,000 samples and rarely improves even with some noise file dropped. For future predictions, we believe the accuracy model has a larger t than the word based model but still not robust with a small hidden state. After 10 steps, it decreases dramatically and even becomes less than 50%. Although this model is historically famous, we have to say that its accuracy still have a long way to go.

Section 9. Our Unique Idea for Other Sequence Models

We find stock price prediction to be very attractive. Stock price is an inherently sequential data. We could imagine having stocks from the same industry to predict the general trends of the stocks. Stocks from complementary industry or market index can be used as priors. The format of the data can be the prices of stocks per minute or even per seconds. We can call it a success if the portfolio recommended can beat the market.

Thank You for Your Grading!