I spent multiple hours trying many different attempts of getting my lists of tokens and augmented sentence tokens to work, but they kept using a shallow copy, so the references were overwriting the previous lists. I tried googling for answers online but using .copy(), [:] and list() were not working. I didn’t want to import copy.deepcopy because it would require importing a library. I used Clause 3.5-sonnet in Cursor to discover a very efficient usage of list comprehension! I’m not very great at Python so this was cool to see it in such a compact and understandable way. After this I went to Youtube to learn more about List Comprehension.

For example, printing square(i) of 1 to 10 would be print([i \* i *for* i *in* range(1,11)])

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

From these multiple different lines, I was able to further improve them (by myself) by merging them all into a single line in a for loop: 

Later, I also used AI to compact some code, such as this:  


When I was transitioning into using both sentence and paragraph modes, I introduced another layer of array dimensionality, and since the code was becoming so complex it was difficult for me to locate which exact locations needed to be updated or to find parts that I accidentally missed. I had Claude 3.5-sonnet help me identify the locations to help make my sentence and paragraph levels work as I intended. I received some extra changes in these responses that I didn’t want to use, I just used these responses to locate parts of my code that I missed in my transition to using the modes dimension.

A screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a black screen

Description automatically generatedA computer screen shot of a program code

Description automatically generatedA screenshot of a computer error

Description automatically generatedA screen shot of a computer program

Description automatically generatedA screenshot of a black screen

Description automatically generated

Given that my code was taking 8 minutes to calculate the probabilities I investigated my time complexity of the main probability loop. It was also not responding and crashed.

A screenshot of a computer

Description automatically generated

A black background with white text

Description automatically generated

A screen shot of a computer code

Description automatically generated

To which Cursor-small responded that I should precompute my context totals rather than summing every single time, which I didn’t even think about before. I used most of the changes it suggested as they were greatly beneficial for the time reduction, but I did have to modify the results to properly work with my codebase.

A screenshot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

Which effectively reduced the time to less than half a minute! 