The aim of this part is to carry out Optical Character Recognition on text-based images using Naïve Bayes Classifier (Bayes Net) and Hidden Markov Model with MAP inference (Viterbi).

**Preprocessing the Data**

It is suggested that a text training file **bc.train** from Part 1 be used, which is representative of the English language. Since **bc.train** has been taken from Part 1, the file has a number of words and special characters such as parts of speech etc that are not required for Part 3. We handle this in the **TestPreprocessing(file)** function, where the POS tokens, and other characters are replaced.

**Probability Distribution**

The initial probability distribution and the transition probabilities are calculated in the **FindProbDist(text)** function.

**Initial Probability:**

The number of times a sentence begins with a particular number is stored in the form of a dictionary. The values are normalized.

**Transition Probability:**

Here the probability of the occurrence of all letters given a particular letter is calculated and stored. It is stored in the form of a dictionary of dictionaries. These values are also normalized.

**Emission Probability:**

For the emission probability, we used a simple Naïve Bayes Classifier.  If we assume that m% of the pixels are noisy, then a naive Bayes classifier could assume that each pixel of a given noisy image of a letter will match the corresponding pixel in the reference letter with probability (100 - m)%.

**Algorithm**

1. **Bayes Net:** 
   1. This is done using function **NBC(noisy\_img, letters, m).**
   2. Looping through each letter in the noisy image, we iterate through all the letters available and calculate the emission probability for a letter, given the noisy image letter.
   3. The emission probability of each letter is compared, and the letter with the best emission probability is added to a variable holding the best letters/characters.
2. **HMM with Viterbi:**
   1. The initial probabilities are calculated for all the actual letters (hidden states), for the image of the noisy letter observed.
   2. The transition probabilities and the emission probabilities are calculated for each state.
   3. Once we have all the probabilities, we use the Viterbi algorithm to calculate the best sequence of letters.

**Sample Output**