## How I have solved the task

### Analysing the zeroth order source statistics

* Create an array the same size as the number of characters in our alphabet (also keep a reference to which character is represented by which slot in the array).
* Go through the text and store the number of times each character (only those in our alphabet) occurs.
* Then divide the stored numbers by the total of characters analysed to get the probability of each.

### Generate text based on the zeroth order statistics

* Uniformly generate a number between 0 and 1 (exclusive)
* Sum the probabilities until the sum is strictly greater than the randomly generated number and append the character whose probability was last added to the sum.
* Example (using a smaller alphabet for simplicity) :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Probabilities: |  | | a | 0.1 | | b | 0.1 | | c | 0.2 | | d | 0.4 | | e | 0.2 | | |  |  | | --- | --- | | Random generated nr: | Result: | | 0.33 | c | | 0.1 | b | | 0.0 | a | | 0.02 | a | | 0.99 | e | |



### Analysing orders > zeroth

* Create a transition matrix with initially zero rows
  + It can store unique prefixes as rows, with columns of possible suffixes and how many times they have occurred.
  + For each prefix, also store the total of transitions recorded
* Start by looking at the character at position , where is equal to the order. If it is not in the alphabet, continue to the next one.
* Then, check if the previous characters are all in the alphabet. Continue to next position if any are not in the alphabet.
* Check if the transition matrix has a row for the current prefix.
  + If it does not, add a new row with the prefix, and the suffix character as a possible transition
  + If it exists, add an occurrence of the suffix character.

### Generate text based on order > zeroth

* Analyse the source statistics from zeroth to the desired order
* Generate the first character based on the zeroth order statistics, second character based on 1st order, third character based on 2nd order etc, up to desired order -1.
* Generate the rest of the text based on the statistics of the given order.
* Characters are generated in a similar manner as described for zeroth order;
  + A random number between 0 and the total occurrences of suffixes is generated.
  + Sum the number of occurrences for each suffix until the sum is greater than the random number.
  + Append the last suffix of which occurrence number was added.

## Observation

* Zeroth order
  + The text does not make any sense, but the frequency of spaces seems to be about correct for Norwegian text.
* 1st order
  + Some parts of the text are actual words, but mostly, it’s not readable
* 2nd order
  + About half of the “words” are actual words
* 3rd order
  + Almost all the text is made up of readable words.

## Are they unifilar?

At order n > 0, they are unifilar because all states produce distinct symbols, which determine the next state. I do not know how the unifilar property is defined for a source which has no state, as with zeroth order.

## Entropy

### Zeroth order

Entropy calculated by  
 where is the alphabet size.   
Result: 3.999233313493003

### 1st Order

Result: 3.024102190660886