A Shift Cipher Cracking Algorithm

The Shift Cipher

The shift cipher, also known as Caesar's cipher, takes a plaintext m and shifts each letter forward a set number of times k. If $m =$ "Stop: Hammer time!" then its encryption with key $k = 4$ produces the ciphertext $Enc_k(m) =$ "WXSTLEQQIVXMQI". Notice that all letters are capitalized and all spaces and punctuation are removed. My implementation of the shift cipher in Clojure uses the code below to psuedo-randomly generate a key.

```
(+ 1 (rand-int 25))
```

The Cracking Algorithm

The cracking algorithm cycles through each possible value $[1, 25]$ for k and finds the one that produces the text which has letter distribution that most closely matches the average letter distribution in English, as explained below. Let p_i represent the average frequency of the ith letter of the English alphabet (e.g., $p_1 =$ "a" $= 0.08167$) and let q_i represent the frequency of occurence of the ith letter in our ciphertext. To find which value k will shift the ciphertext such that it's letter frequency distribution most closely matches English it calculates

$I_j = \sum_{i=1}^{26} p_i * q_{i+j}$ for all j in $[1, 25]$.

Then it determines for which j the value I_j is closest to

$\sum_{i=1}^{26} p_i^2 \approx 0.0655$.

This is chosen as the value for k and then it reproduces the original text, albeit uniformly capitalized and sans punctuation and spacing.