Hand-in 3

Michael Iversen
Student ID: 201505099

December 5, 2022

In this hand-in, I implement a hidden Markov model following the Codon model from the lectures. The model contains 5 hidden states: Non-coding, start codon, coding, stop codon, reverse start codon, reverse coding and reverse stop codon. The non-coding state emits from the alphabet $\mathcal{A} = \{A, G, C, T\}$ while all other hidden states emits from the alphabet $\mathcal{A} \times \mathcal{A} \times \mathcal{A} = \{AAA, AAG, AAC, AAT, AGA, ..., TTT\}$. The final model is illustrated in Fig. 1 along with the non-zero transition and emission probabilities. The transition and emission probabilities which are zero have been omitted in this figure. I obtain the model from training-by-counting without hard-coding any transition or emission probabilities. A priori, every transition between hidden states and emission of any symbols are allowed. However, after training, some transitions and emission have probability zero because they didn't appear during training.

I translate the given annotations of non-coding N, coding C and reverse coding R into a list of hidden states. I represent the non-coding state as "0", start codon as "1", coding state as "2", stop codon as "3", reverse start codon as "4", reverse coding state as "5" and reverse stop codon as "6". The non-coding part of the annotations are directly translated $N \to 0$. The start codons are identified as a shift from N to C. Similarly, the stop codons are identified as a shift from C to N. The coding state is determined as part of the annotations where three letters are C with both the preceding 3 and following 3 letters are all C. The reverse start codons, reverse stop codons and reverse coding states are identified using similar considerations. The code snippet below illustrates this translation.

```
def str_to_list(z: str) -> list[int, ...]:
    z_list = []
    n = 0
    while n < len(z):
        if z[n] == 'N':
            z_list.append(0)
            n += 1
        elif z[n] == 'C':
            if 'N' in z[n - 3: n]:
                z_list.append(1)
            elif 'N' in z[n + 1: n + 4]:
                z_list.append(3)
            else:
                z_list.append(2)
            n += 3
        elif z[n] == 'R':
            if 'N' in z[n - 3: n]:
                z_list.append(4)
            elif 'N' in z[n + 1: n + 4]:
```

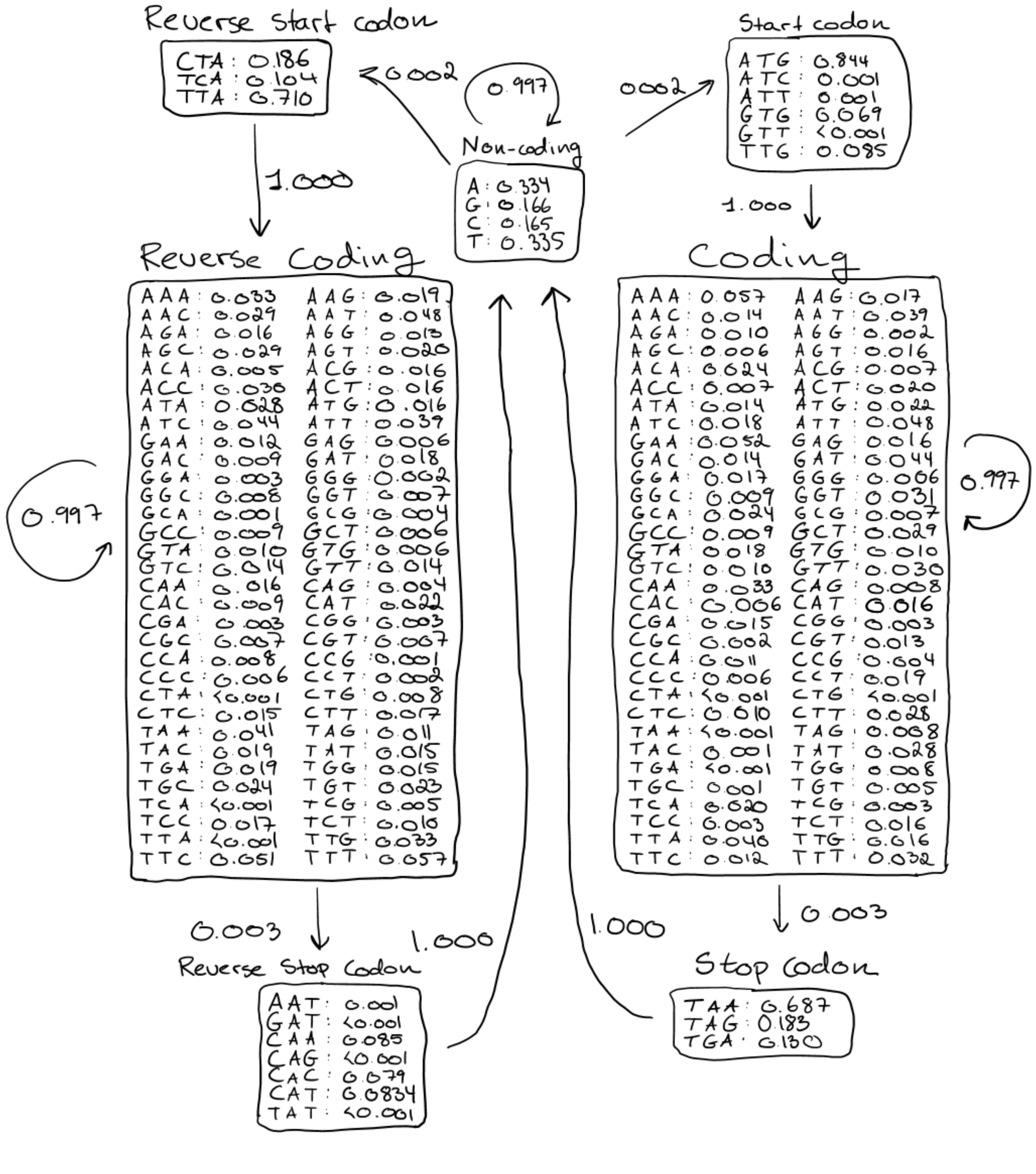

Figure 1: Model of hidden Markov model with transition and emission probabilities. Only non-zero probabilities are shown.

Known genomes	**Only Cs**	**Only Rs**	**Both**
Genome 1	0.7144	0.7675	0.6246
Genome 2	0.7549	0.7411	0.6303
Genome 3	0.7870	0.7751	0.6760
Genome 4	0.7642	0.7379	0.6248
Genome 5	0.7930	0.7390	0.6478

Unknown genomes	**Only Cs**	**Only Rs**	**Both**
Genome 6	0.7439	0.7513	0.6356
Genome 7	0.7630	0.7384	0.6380
Genome 8	0.7497	0.7049	0.5993
Genome 9	0.7209	0.6767	0.5297
Genome 10	0.6525	0.6656	0.4576

Table 1: Performance of hidden Markov model on known genomes (left) and unknown genomes (right)

```
                z_list.append(6)
            else:
                z_list.append(5)
            n += 3
    return z_list
```

Translating a list of hidden states to an annotation in terms of N, C and R is straightforward. I simply map $0 \to N$, $1 \to CCC$, $2 \to CCC$, $3 \to CCC$, $4 \to RRR$, $5 \to RRR$ and $6 \to RRR$. This procedure is illustrated in the code snippet below.

```
def list_to_str(z: list[int, ...]) -> str:
    def decoding(i: int) -> str:
        if i == 0:
            return 'N'
        elif i == 1 or i == 2 or i == 3:
            return 'CCC'
        else:
            return 'RRR'
    return ''.join(map(decoding, z))
```