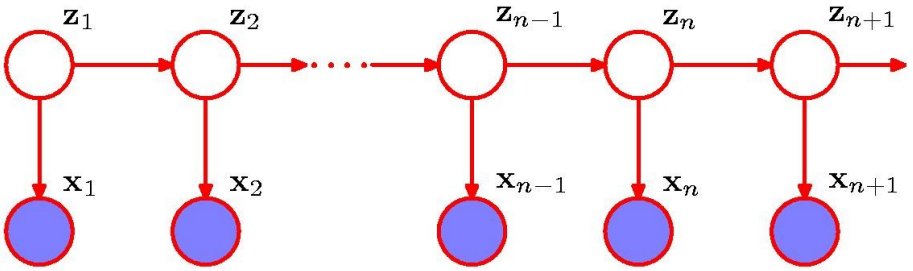
Hidden Markov Models

Hand In 3 - Gene finding



C: coding left-to-right

- A gene is a substring of the DNA sequence of A,C,G,T's
- A gene starts with a start-codon **atg**
- A gene ends with a stop-codon **taa**, **tag** or **tga**
- The number of nucle
  is a multiplum of 3

N: No

Even more b

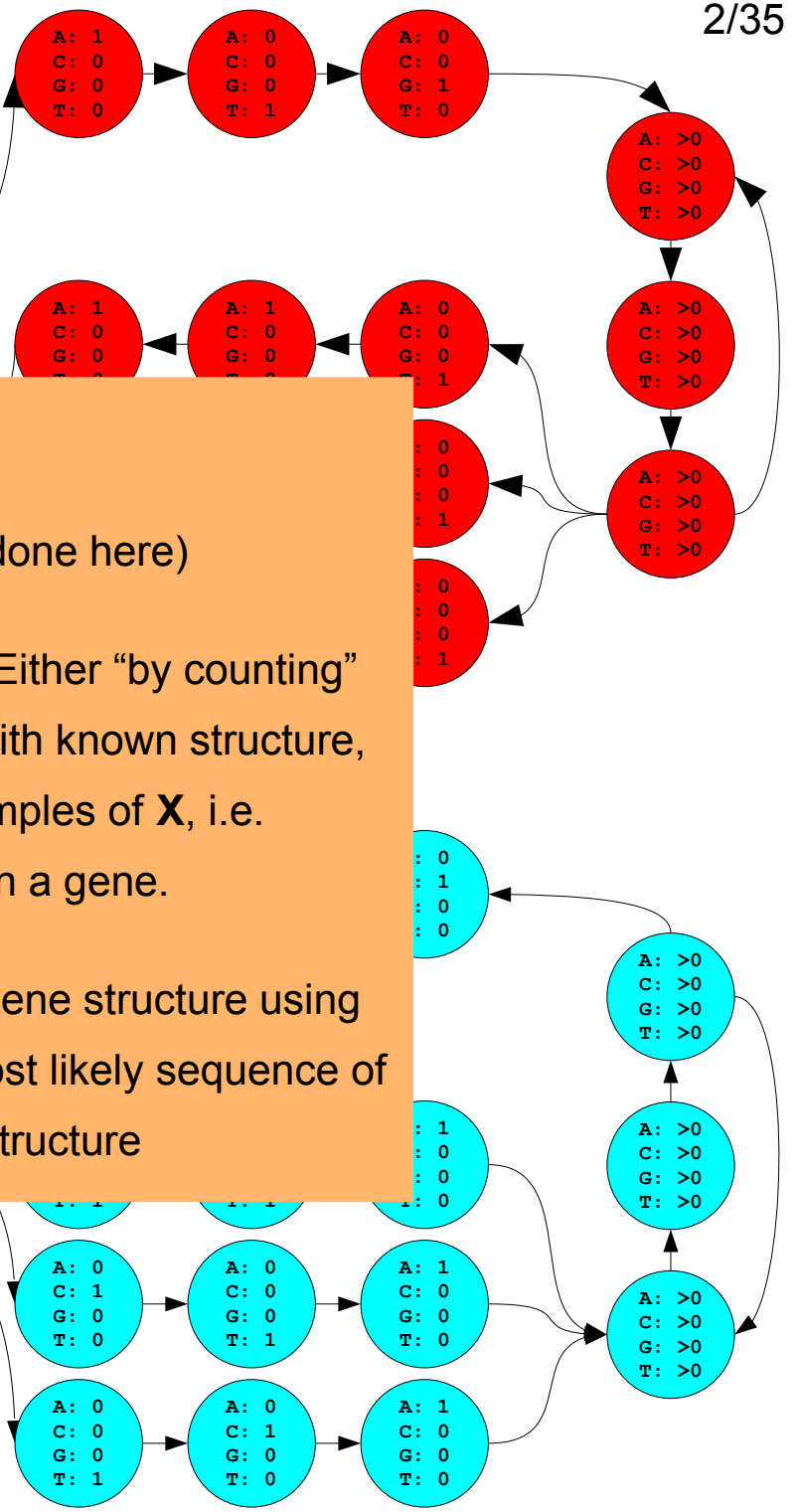
There can be genes in

$\pi_N = 1$

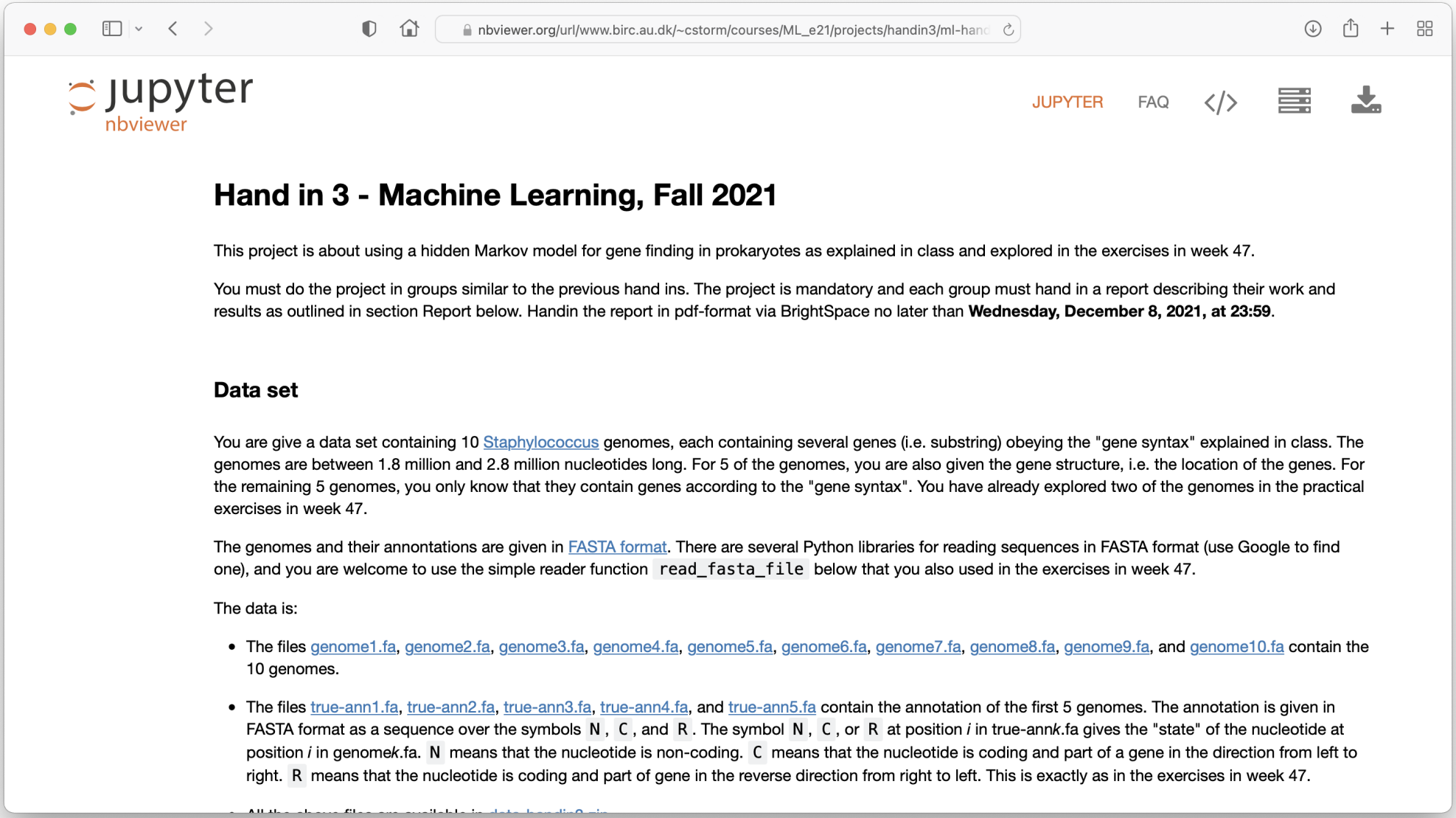
$\pi_C = 0$

Gene finding

- Select initial model structure (e.g. as done here)
- Select model parameters by training. Either “by counting” from examples of (**X**,**Z**)'s, i.e. genes with known structure, or by EM- or Viterbi-training from examples of **X**, i.e. sequences which are known to contain a gene.
- Given a new sequence **X**, predict its gene structure using the Viterbi algorithm for finding the most likely sequence of underlying latent states, i.e. its gene structure



R: coding right-to-left



https://www.birc.au.dk/~cstorm/courses/ML_e21/

Data

You are give a data set containing 10 Staphylococcus genomes, each containing several genes (i.e. substring) obeying the "gene syntax" explained in class. The genomes are between 1.8 million and 2.8 million nucleotides long. For 5 of the genomes, you are also given the gene structure, i.e. the location of the genes. For the remaining 5 genomes, you only know that they contain genes according to the "gene syntax". You have already explored two of the genomes in the practical exercises in week 47.

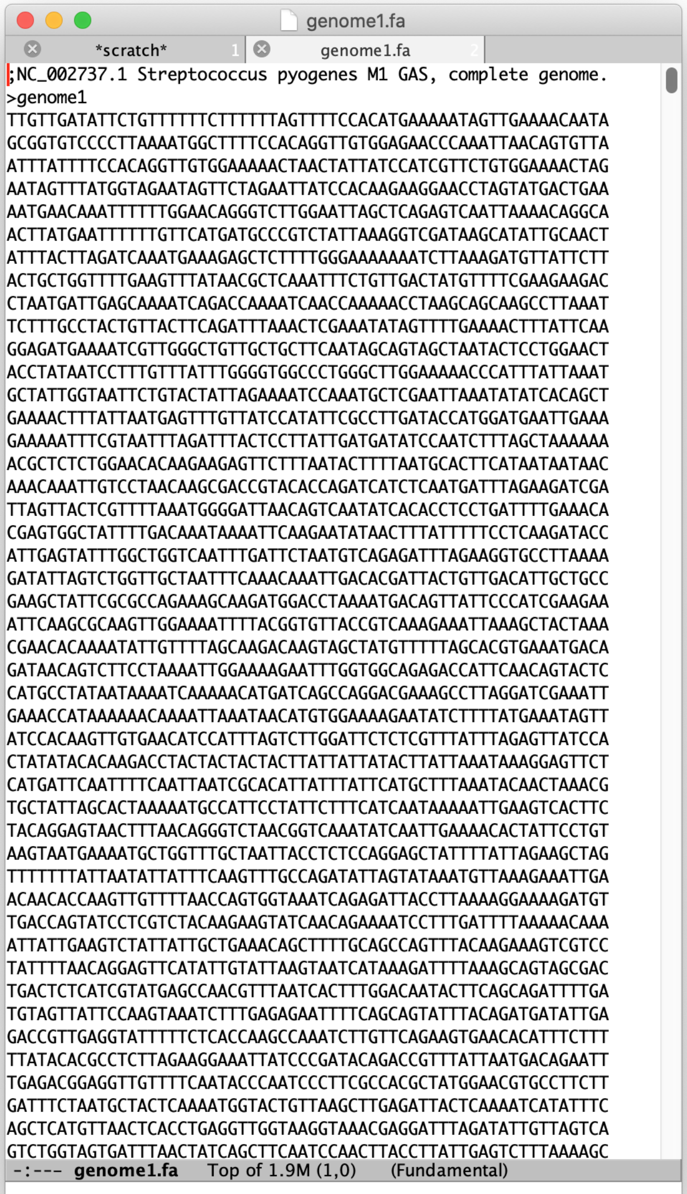
The genomes and their annontations are given in FASTA format. There are several Python libraries for reading sequences in FASTA format (use Google to find one), and you are welcome to use the simple reader function `read_fasta_file` below that you also used in the exercises in week 47.

The data is:

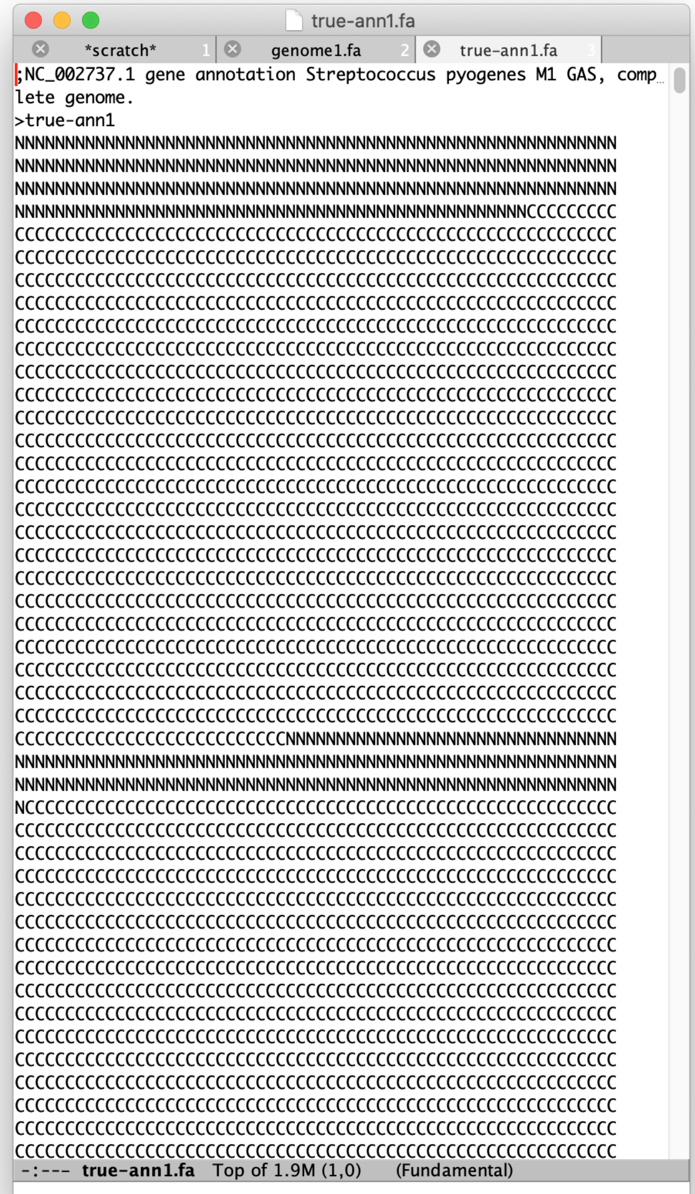
- The files genome1.fa, genome2.fa, genome3.fa, genome4.fa, genome5.fa, genome6.fa, genome7.fa, genome8.fa, genome9.fa, and genome10.fa contain the 10 genomes.
- The files true-ann1.fa, true-ann2.fa, true-ann3.fa, true-ann4.fa, and true-ann5.fa contain the annotation of the first 5 genomes. The annotation is given in FASTA format as a sequence over the symbols `N`, `C`, and `R`. The symbol `N`, `C`, or `R` at position *i* in true-ann_k_.fa gives the "state" of the nucleotide at position *i* in genome_k_.fa. `N` means that the nucleotide is non-coding. `C` means that the nucleotide is coding and part of a gene in the direction from left to right. `R` means that the nucleotide is coding and part of gene in the reverse direction from right to left. This is exactly as in the exercises in week 47.
- All the above files are available in data-handin3.zip

A simple analysis of the annotations and the corresponding genomes, it is clear that fx several start-codons are possible (and not only atg as modelled in class).

Data



genome1.fa



true-ann1.fa

Analysis of data

```
Length of genome1: 1852441 (1852441)
Length of genome2: 2211485 (2211485)
Length of genome3: 2499279 (2499279)
Length of genome4: 1796846 (1796846)
Length of genome5: 2685015 (2685015)
Length of genome6: 2127839 (2127839)
Length of genome7: 2742531 (2742531)
Length of genome8: 2046115 (2046115)
Length of genome9: 2388435 (2388435)
Length of genome10: 1570485 (1570485)
Length of genome11: 2096309 (2096309)
```

We observe

3 typical start-codons

3 stop-codons

We may ignore rare start and stop codons, i.e. a gene that starts (or ends) with a ignored start (or stop) codon does not correspond to a path in your model.

```
Start-codon in normal genes:
ATG [8423, 'NCCC']
ATC [3, 'NCCC']
ATA [1, 'RCCC']
GTG [713, 'NCCC']
ATT [3, 'NCCC']
CTG [2, 'NCCC']
GTT [1, 'NCCC']
CTC [1, 'NCCC']
TTA [1, 'NCCC']
TTG [1020, 'NCCC']

Stop-codon in normal genes:
TAG [1949, 'CCCN']
TGA [1531, 'CCCN']
TAA [6686, 'CCCN']

Reversed stop-codon in reversed genes:
TTA (reverse-complement: TAA) [6596, 'NRRR']
CTA (reverse-complement: TAG) [2014, 'NRRR']
TCA (reverse-complement: TGA) [1148, 'NRRR']

Reversed start-codon in reversed genes:
TAT (reverse-complement: ATA) [2, 'RRRN']
ATG (reverse-complement: CAT) [1, 'RRRN']
GAT (reverse-complement: ATC) [1, 'RRRN']
CAT (reverse-complement: ATG) [8077, 'RRRN']
AAT (reverse-complement: ATT) [4, 'RRRN']
TAC (reverse-complement: GTA) [1, 'RRRN']
CAC (reverse-complement: GTG) [715, 'RRRN']
CAA (reverse-complement: TTG) [953, 'RRRN']
CAG (reverse-complement: CTG) [4, 'RRRN']
```

Data

You are give a data set containing 10 Staphylococcus genomes, each containing several genes (i.e. substring) obeying the "gene syntax" explained in class. The genomes are between 1.8 million and 2.8 million nucleotides long. For 5 of the genomes, you are also given the gene structure, i.e. the location of the genes. For the remaining 5 genomes, you only know that they contain genes according to the "gene syntax". You have already explored two of the genomes in the practical exercises in week 47.

The genomes and their annontations are given in FASTA format. There are several Python libraries for reading sequences in FASTA format (use Google to find one), and you are welcome to use the simple reader function `read_fasta_file` below that you also used in the exercises in week 47.

The data is:

- The files genome1.fa, genome2.fa, genome3.fa, genome4.fa, genome5.fa, genome6.fa, genome7.fa, genome8.fa, genome9.fa, and genome10.fa contain the 10 genomes.
- The files true-ann1.fa, true-ann2.fa, true-ann3.fa, true-ann4.fa, and true-ann5.fa contain the annotation of the first 5 genomes. The annotation is given in FASTA format as a sequence over the symbols `N`, `C`, and `R`. The symbol `N`, `C`, or `R` at position *i* in true-ann_k_.fa gives the "state" of the nucleotide at position *i* in genome_k_.fa. `N` means that the nucleotide is non-coding. `C` means that the nucleotide is coding and part of a gene in the direction from left to right. `R` means that the nucleotide is coding and part of gene in the reverse direction from right to left. This is exactly as in the exercises in week 47.
- All the above files are available in data-handin3.zip

A simple analysis of the annotations and the corresponding genomes, it is clear that fx several start-codons are possible (and not only atg as modelled in class).

Problem

Your task is to predict the gene structure of the 5 unannotated genomes, i.e. genome6.fa, genome7.fa, genome8.fa, genome9.fa, and genome10.fa, in the best possible manner using an HMM based approach.

As explained in class, predicting gene structure using an HMM involves:

- Deciding on an initial model structure, i.e. the number of hidden states and which transitions and emission should have a fixed probability (e.g. 0 for "not possible", or 1 for "always the case"). You might get inspiration from, or even use, one of the model structures discussed in class. However, be aware that the models presented in class model only one start-codon atg. This is not the case in the presented data, where multiple start-codons are possible (as also mentioned in the lecture/slides about this project).

- Tune model parameters by training, i.e. set the non-fixed emission and transition probabilities. This can be done by several methods as explained in class. You can use (1) Training-by-Counting using the 5 genomes with know gene structure because you know the underlying hidden state for each observation (i.e. symbol) in these sequences, or (2) Viterbi- or EM-training using all the 10 genomes, including the 5 genomes with unknown gene structure. You can of course also combine the two approaches. You can evaluate the performance of your gene finder by computing the approximate correlation (AC) between your predictions and the true structure using this small python program compare_anns.py (see this paper for details).

  In this project, Training-by-Counting is mandatory, while Viterbi- or EM-training is optional. If you implement EM-training you should of course be aware of the numerical problems of the forward- and backward-algorithms unless you use scaling or similar techniqes.

- Evaluate the performance of your gene predictor. In this project, you must do a 5-fold cross validation on the 5 genomes with known gene structure, i.e.:

  You consider genome 1 to 5 in five rounds. In a each you train your model using Training-by-Counting on the remaining 4 genomes, predict the gene structure of the genome you consider, and compute the approximate correlation coefficient (AC) between your predicted gene structure and the true gene structure using the python program compare_anns.py.

- Use your best model to predict the gene structure for the 5 unannotated genomes using the Viterbi algorithm with subsequent backtracking. I.e. for each unannotated genome you must find a most likely sequence of states in your model generating it, and convert this sequence of states into a FASTA file giving the annotation of each nucleotide as `N`, `C`, or `R`. You should make files, `pred-ann6.fa`, ..., `pred-ann10.fa`, giving the annotation for the 5 unannotated genomes.

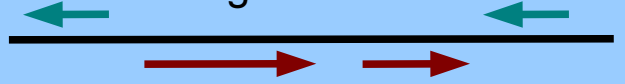
  You can use the www-service GeneFinder Verifier to compare your predictions on Genome 6-10 against their true structures.

Problem - Selecting an initial model

- A gene is a substring of the DNA sequence of A,C,G,T's
- A gene starts with a start-codon **atg**
- A gene ends with a stop-codon **taa**, **tag** or **tga**
- The number of nucleotides in a gene is a multiplum of 3

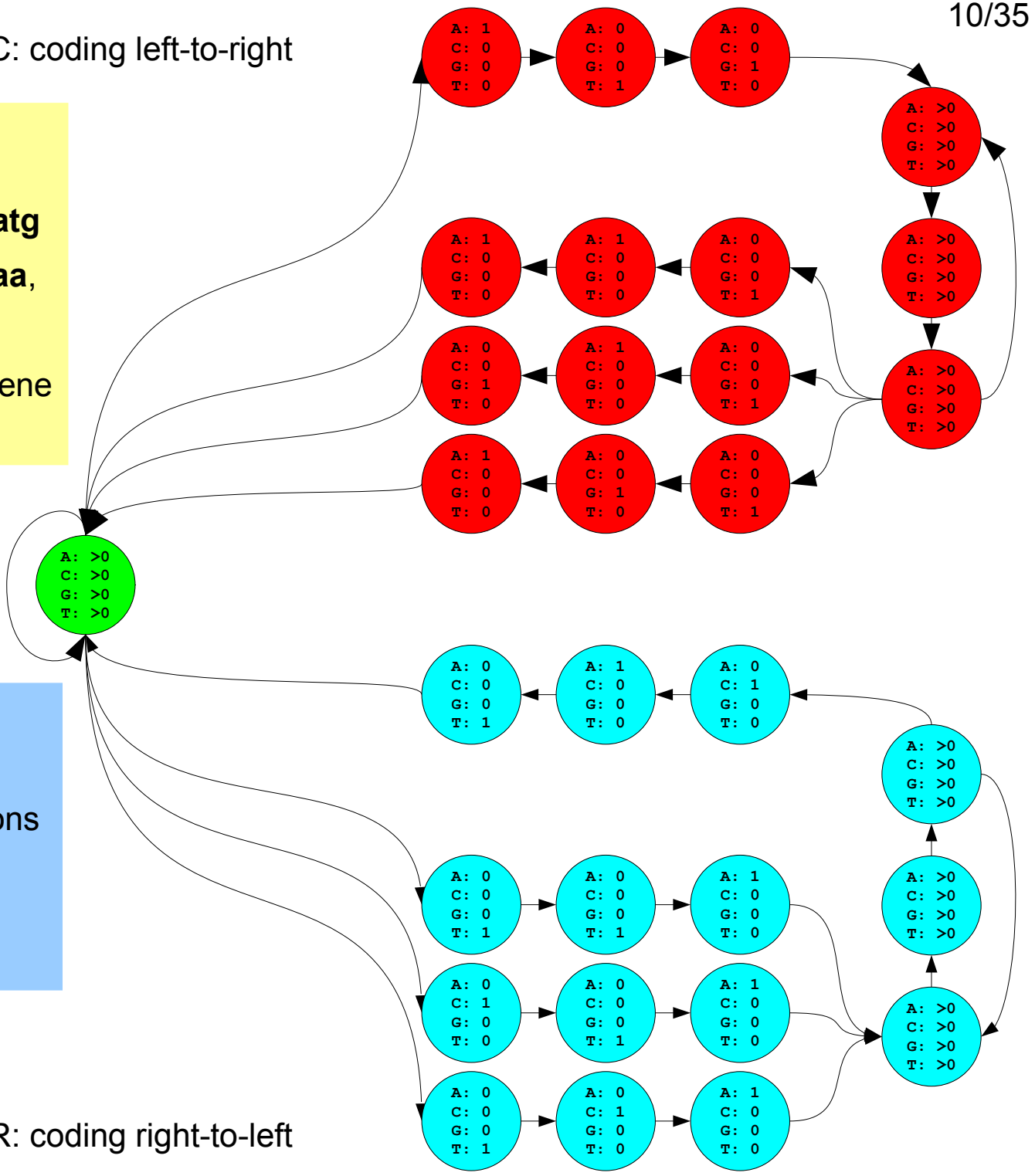
Even more biology

There can be genes in both directions

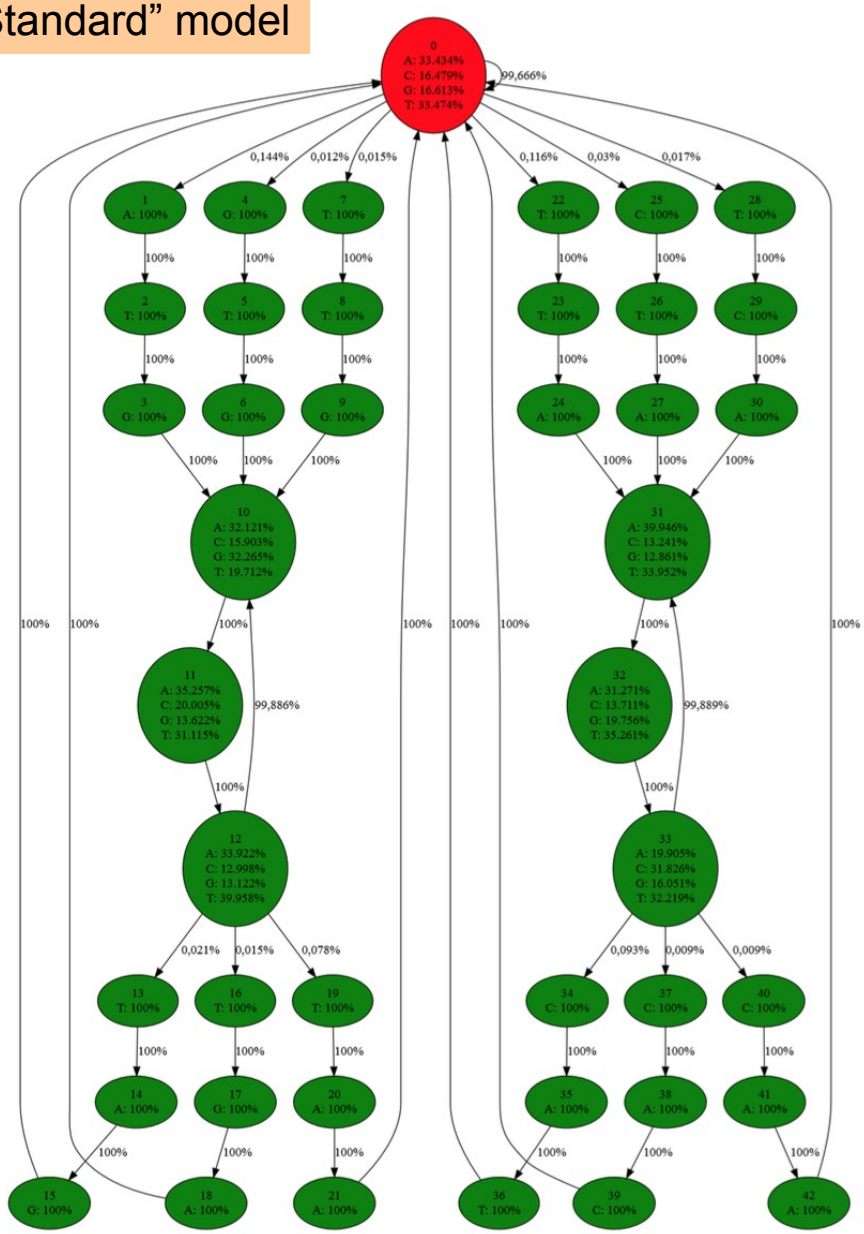


$\pi_N = 1$

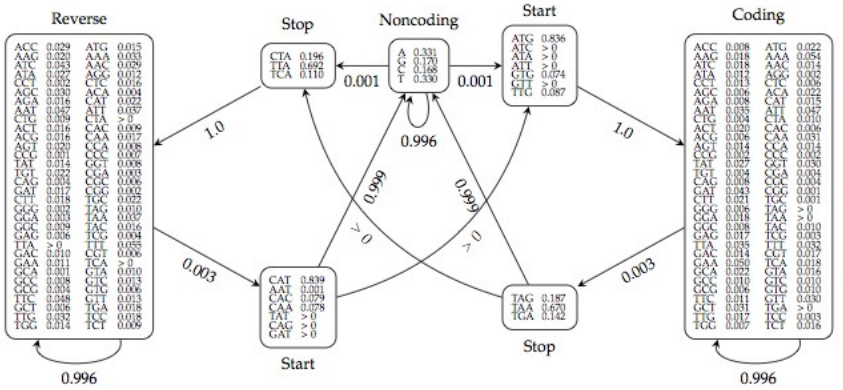
$\pi_C = 0$



“Standard” model



“Codon” model



Problem - 5-fold cross validation

You consider genome 1 to 5 in five rounds. In a each you train your model using Training-by-Counting on the remaining 4 genomes, predict the gene structure of the genome you consider, and compute the approximate correlation coefficient (AC) between your predicted gene structure and the true gene structure using the python program `compare_anns.py`.

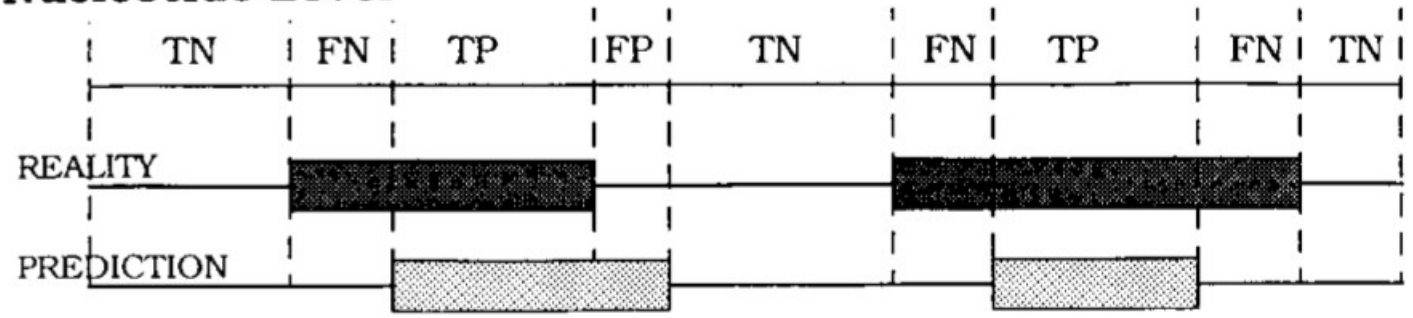
	Round 1	Round 2	Round 3	Round 4	Round 5
Genome 1	Validate	Train	Train	Train	Train
Genome 2	Train	Validate	Train	Train	Train
Genome 3	Train	Train	Validate	Train	Train
Genome 4	Train	Train	Train	Validate	Train
Genome 5	Train	Train	Train	Train	Validate

	AC		
	Only Cs	Only Rs	Both
Genome 1	0.5983	0.6470	0.4347
Genome 2	0.6305	0.6529	0.4510
Genome 3	0.6552	0.6510	0.4805
Genome 4	0.6240	0.6002	0.4079
Genome 5	0.6550	0.6027	0.4302

Evaluating performance

Evaluation of Gene Structure Prediction Programs (Burset and Guigo, 1996)

Nucleotide Level



REALITY

PREDICTION	coding	no coding	
coding	TP	FP	TP+FP
no coding	FN	TN	FN+TN
	TP+FN	TF+TN	

$$Sn = \frac{TP}{TP + FN}$$

Sensitivity

$$Sp = \frac{TP}{TP + FP}$$

Specificity

$$CC = \frac{(TP \times TN) - (FN \times FP)}{\sqrt{(TP + FN) \times (TN + FP) \times (TP + FP) \times (TN + FN)}}$$

Correlation Coefficient

$$ACP = \frac{1}{4}\left[\frac{TP}{TP + FN} + \frac{TP}{TP + FP} + \frac{TN}{TN + FP} + \frac{TN}{TN + FN}\right]$$

$$AC = (ACP - 0.5) \times 2$$

Approximate Correlation

compare_anns.py

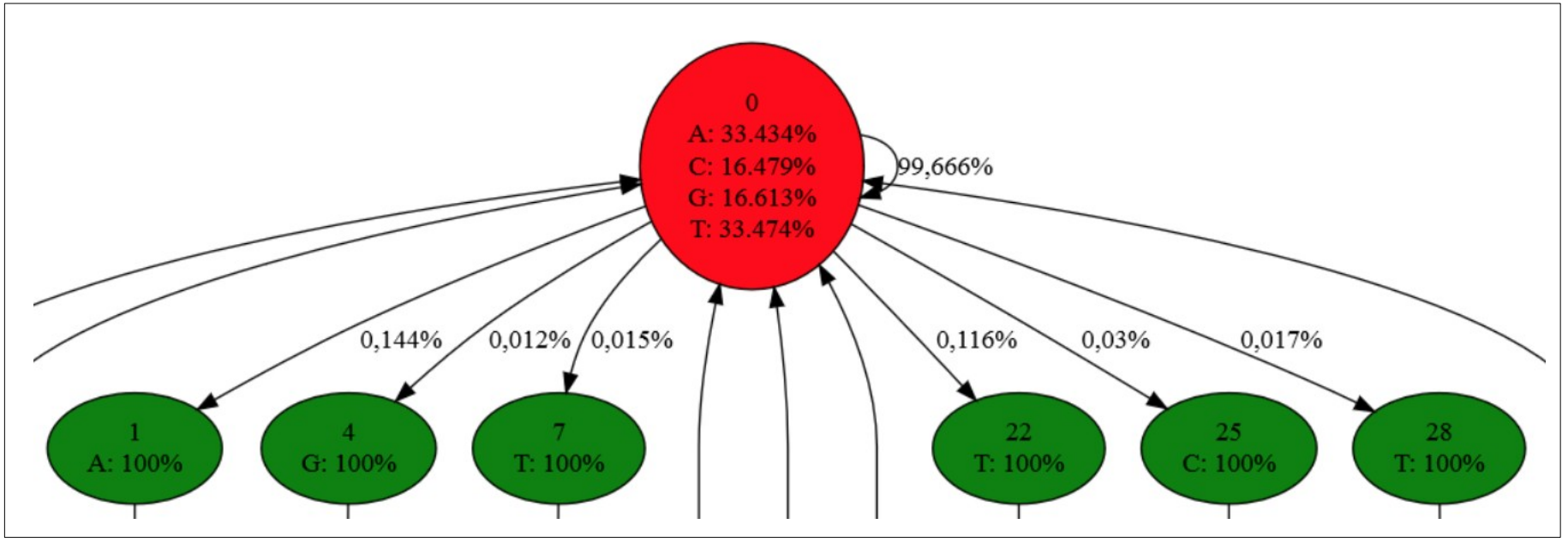
```
$ python compare_anns.py true-ann6.fa pred-ann6.fa
Cs   (tp=757332, fp=164766, tn=305197, fn=57217): Sn = 0.9298, Sp = 0.8213, AC = 0.6213
Rs   (tp=715865, fp=127462, tn=304830, fn=57584): Sn = 0.9255, Sp = 0.8489, AC = 0.6603
Both (tp=1473197, fp=292228, tn=247613, fn=114801): Sn = 0.9277, Sp = 0.8345, AC = 0.4520
```

compare_anns.py

```
$ python compare_anns.py true-ann6.fa pred-ann6.fa
Cs   (tp=757332, fp=164766, tn=305197, fn=57217): Sn = 0.9298, Sp = 0.8213, AC = 0.6213
Rs   (tp=715865, fp=127462, tn=304830, fn=57584): Sn = 0.9255, Sp = 0.8489, AC = 0.6603
Both (tp=1473197, fp=292228, tn=247613, fn=114801): Sn = 0.9277, Sp = 0.8345, AC = 0.4520
```

Problem – Training the model

Training by counting



In theory, we are given (**X**,**Z**) pairs, but we are given **(X**, "**Z**") pairs, where the NCR-annotations have to be translated to **Z**s.

Also, we may ignore rare start and stop codons, i.e. a gene that starts (or ends) with a ignored start (or stop) codon does not correspond to a path in your model.

Training by counting – Typical solution

Example: How to to set the transition probabilities:

N → N	N → CCC, where CCC emits ATG	N → RRR, where RRR emits TTA
	N → CCC, where CCC emits GTG	N → RRR, where RRR emits CTA
	N → CCC, where CCC emits TTG	N → RRR, where RRR emits TCA

We count:

#(N->N) = no. of occurrences of “NN” our annotations.

#(N → CCC, were CCC emits XYZ) = no. of occurrence “NCCC” in our annotations, where CCC is an annotation of XYZ (in our training data).

#(N → RRR, were RRR emits XYZ) = no. of occurrence “NRRR” in our annotations where, RRR is an annotation of XYZ (in our training data).

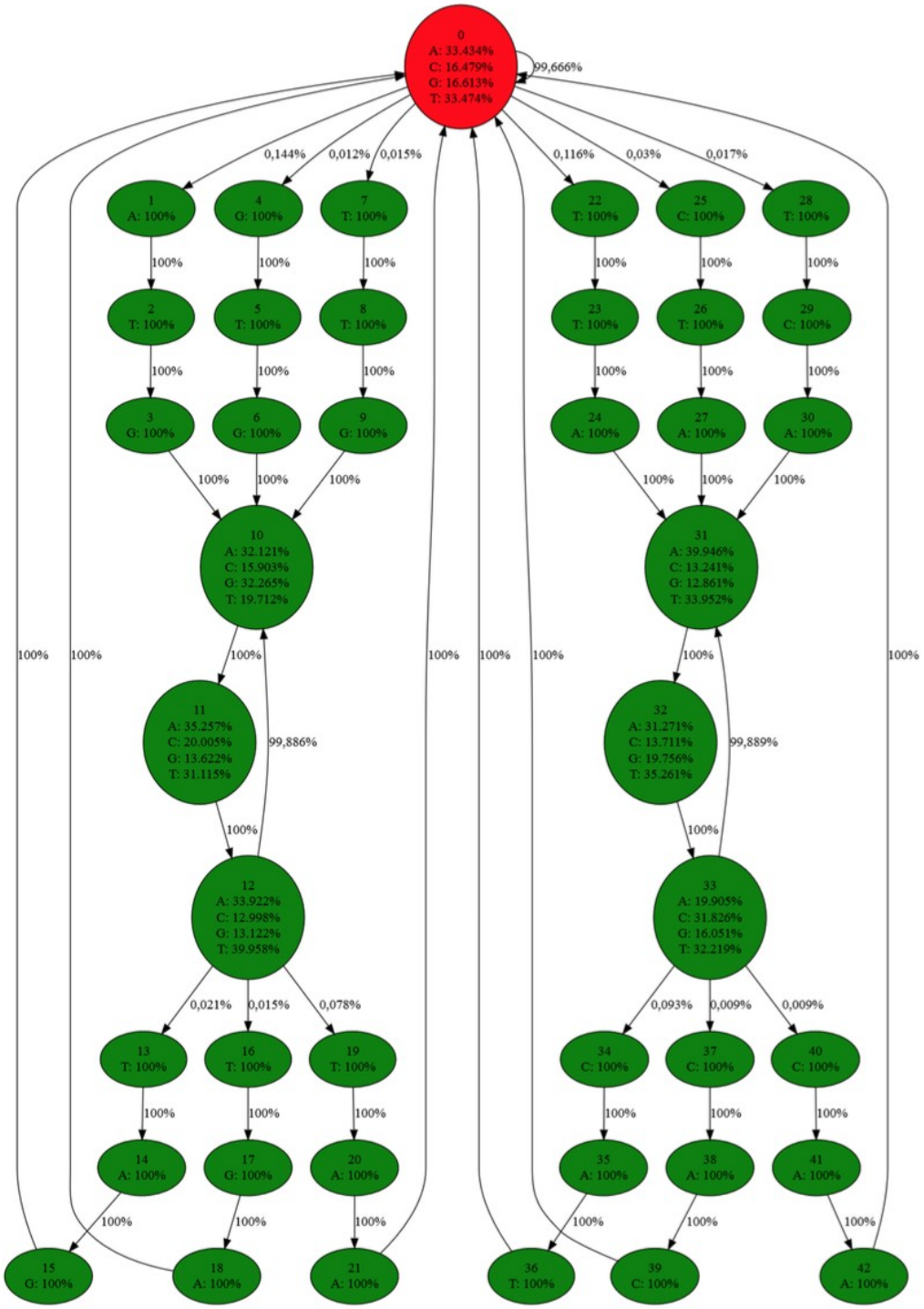
We compute:

Total = #(N → N) + #(N → CCC where CCC emits XYZ) + #(N → RRR, where RRR emits XYZ)

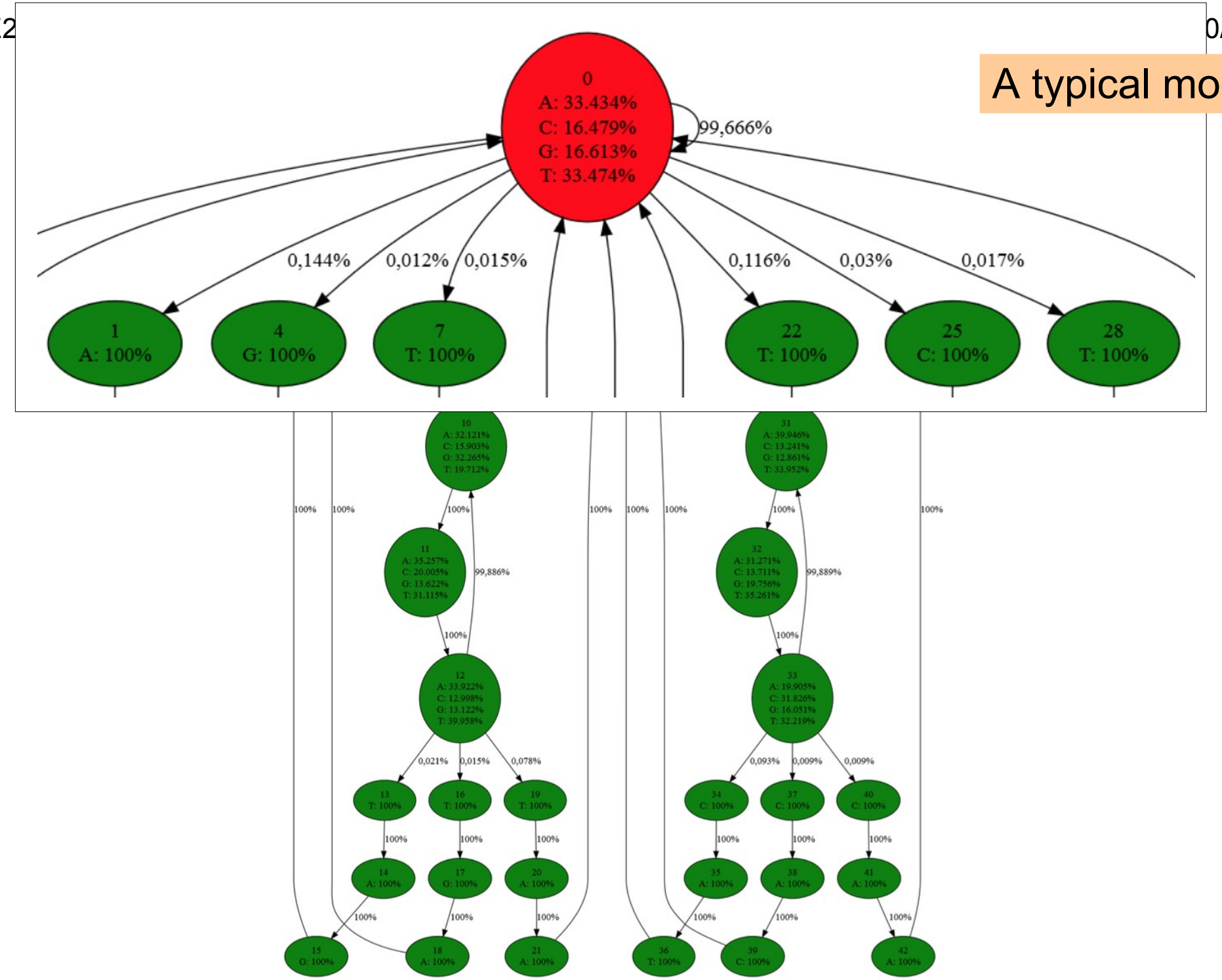
We set:

P(N → X) = #(N → X) / Total for each of the 7 transitions

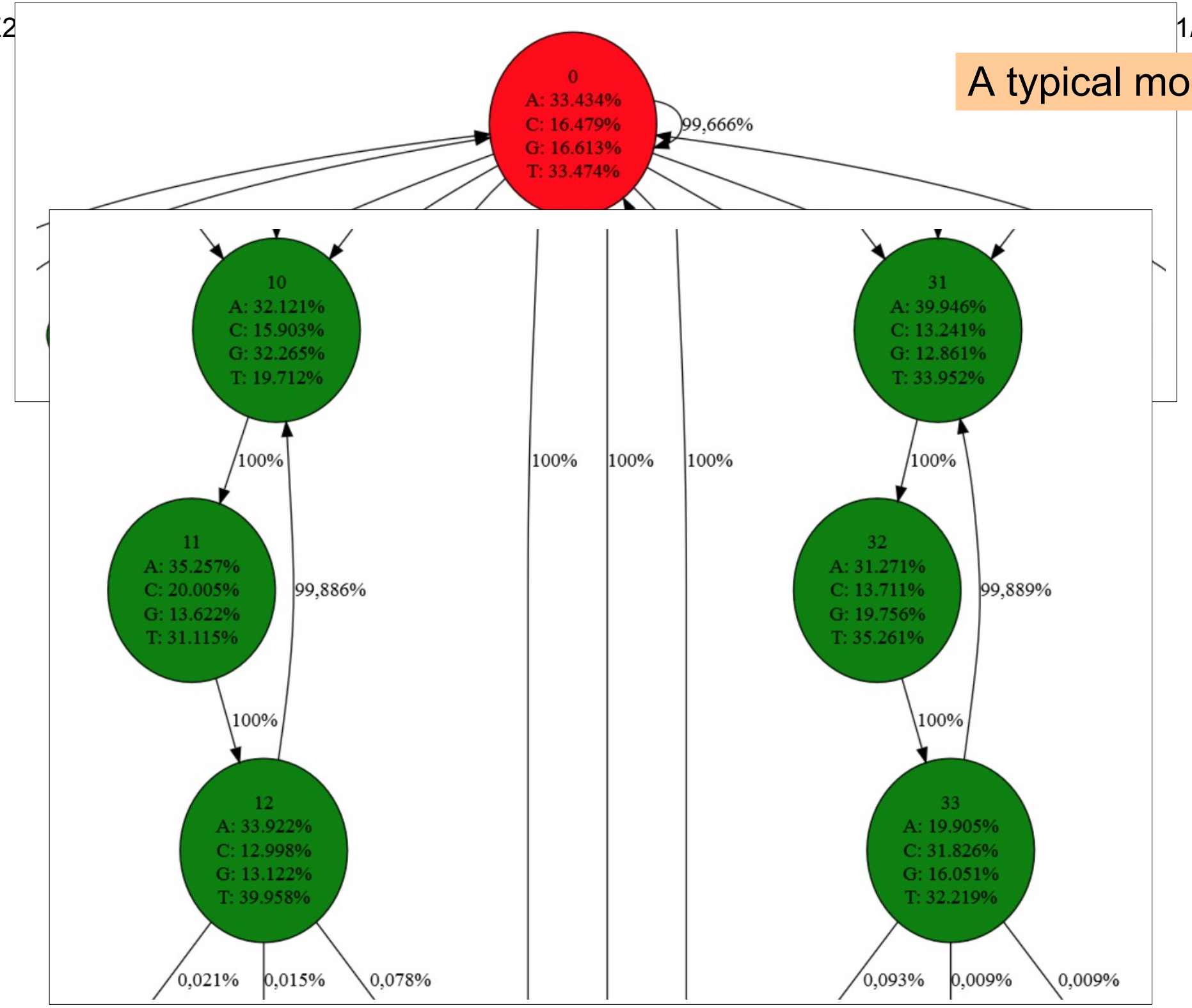
A typical model



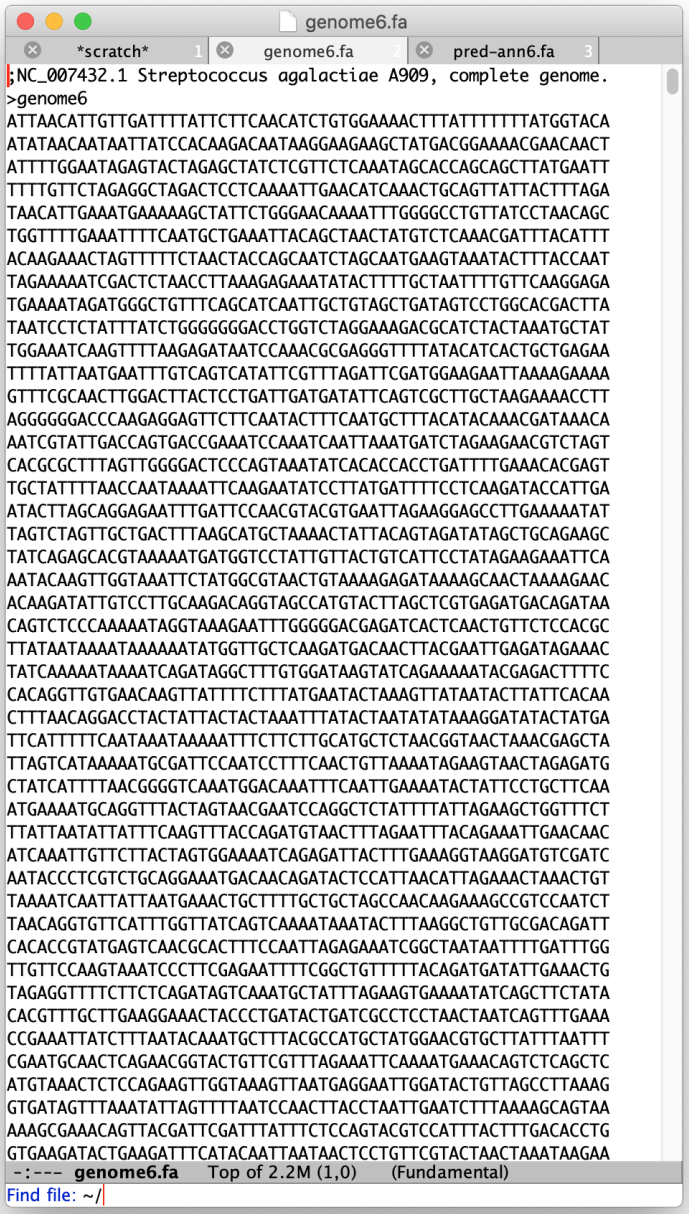
A typical model



A typical model



Problem – Predict on genome 6-10



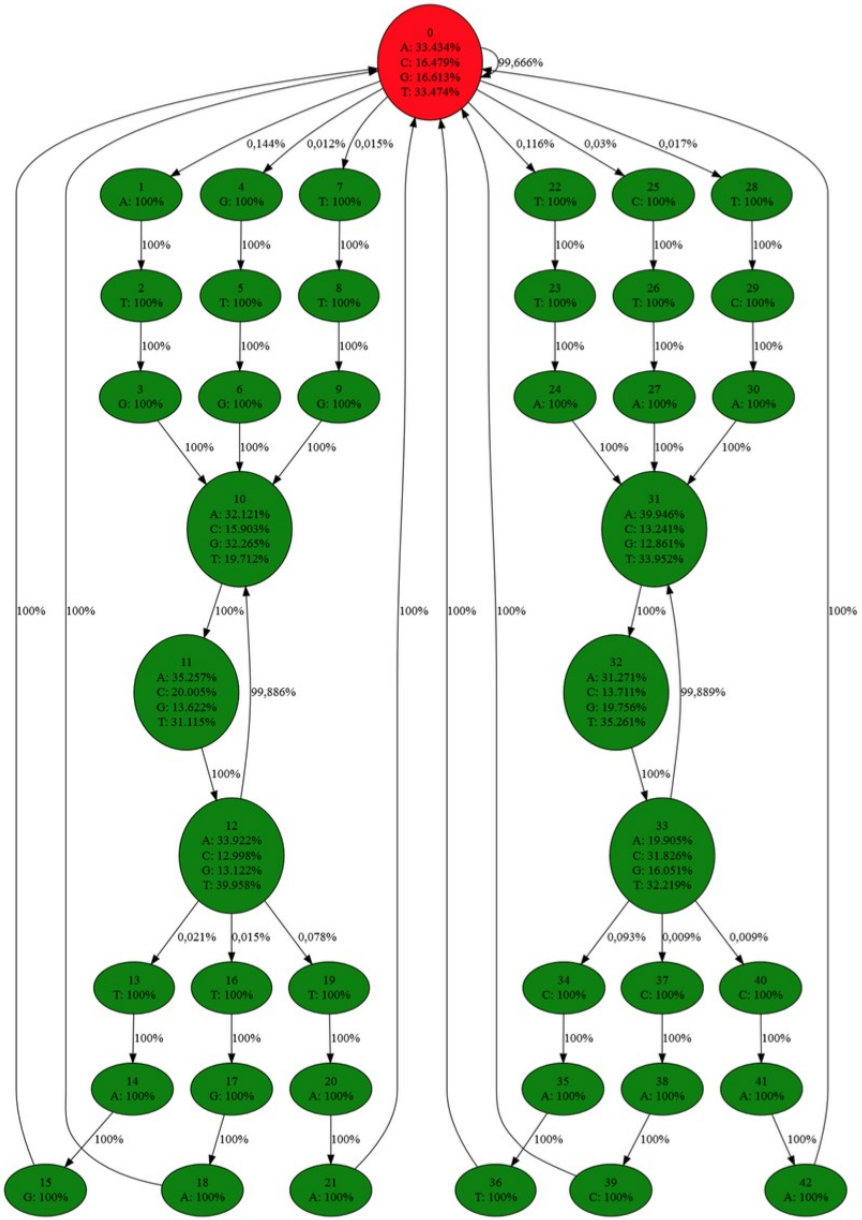
Viterbi decoding and converting to N, C, R annotation

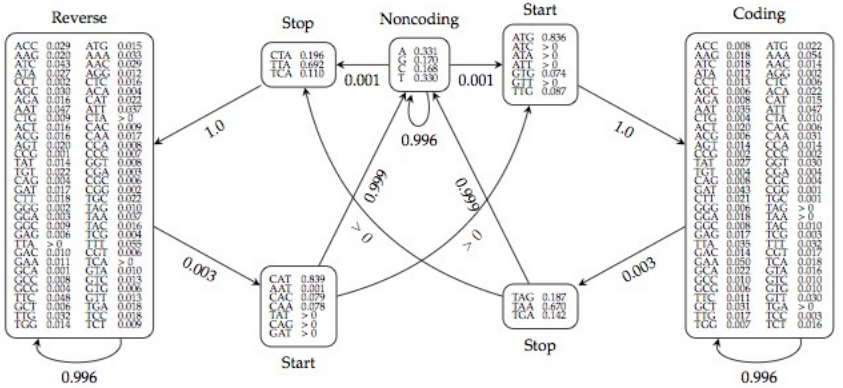
Report

You (i.e. your group) must hand in a short report (in pdf-format) describing your work and results and a zip-archive of the files `pred-ann6.fa` and `pred-ann7.fa`,..., `pred-ann10.fa` containing your predictions on the 5 genomes with unknown structure. The files *must* be in FASTA format similar to the annotation of the other genomes. Handin the report and zip-archive via Blackboard no later than **Wednesday, December 8, 2021, at 23:59**. Your report should be no more than 3 pages. It must cover:

- The status of the work, i.e. does it work, if not, then why.
- An explanation and illustration of your model structure (i.e. a transition diagram). This should include an explanation of how you model start and stop codons, and an explanation of how you have trained your model. For Training-by-Counting you might comment on how you translata the given annotations of `C`, `N`, and `R`'s into corresponding sequences of hidden states. The transition-diagram must contain all non-0 emission- and transition probabilities for the model that you have used for predicting the gene structure of the 5 genomes with unknown gene structure.
- An explanation of how you have predicted the gene structure for the the 5 genomes with unknown gene structure. You should comment on how you translate a most likely sequence of hidden states as returned by Viterbi decoding into a sequence of `C`, `N`, and `R`'s.
- The result of your 5-fold cross valiation on the 5 genomes with known gene structure. The report must include a table with the AC obtained in each of the five rounds of the 5-fold cross validation.
- The result of comparing your predictions on the 5 genomes with unknown gene structure against their true structures via the www-service GeneFinder Verifer.

Report – Transition diagram





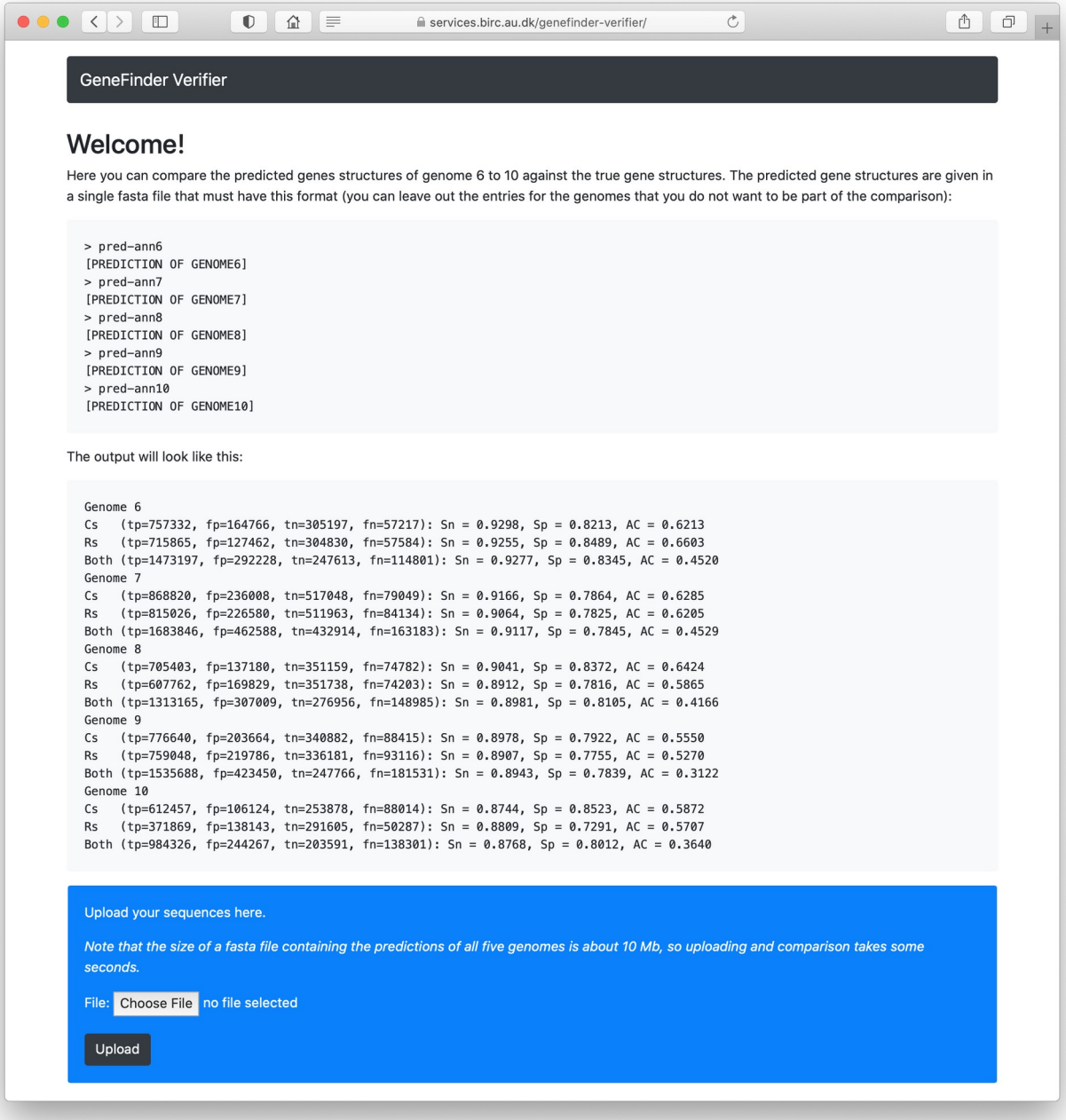
Report - 5-fold cross validation

You consider genome 1 to 5 in five rounds. In a each you train your model using Training-by-Counting on the remaining 4 genomes, predict the gene structure of the genome you consider, and compute the approximate correlation coefficient (AC) between your predicted gene structure and the true gene structure using the python program `compare_anns.py`.

	Round 1	Round 2	Round 3	Round 4	Round 5
Genome 1	Validate	Train	Train	Train	Train
Genome 2	Train	Validate	Train	Train	Train
Genome 3	Train	Train	Validate	Train	Train
Genome 4	Train	Train	Train	Validate	Train
Genome 5	Train	Train	Train	Train	Validate

	AC		
	Only Cs	Only Rs	Both
Genome 1	0.5983	0.6470	0.4347
Genome 2	0.6305	0.6529	0.4510
Genome 3	0.6552	0.6510	0.4805
Genome 4	0.6240	0.6002	0.4079
Genome 5	0.6550	0.6027	0.4302

Report - Performance on genome 6-10



https://services.birc.au.dk/genefinder-verifier/

Report - Performance on genome 6-10

Input is a fasta-file containing your predictions. It is about 10 Mb big!

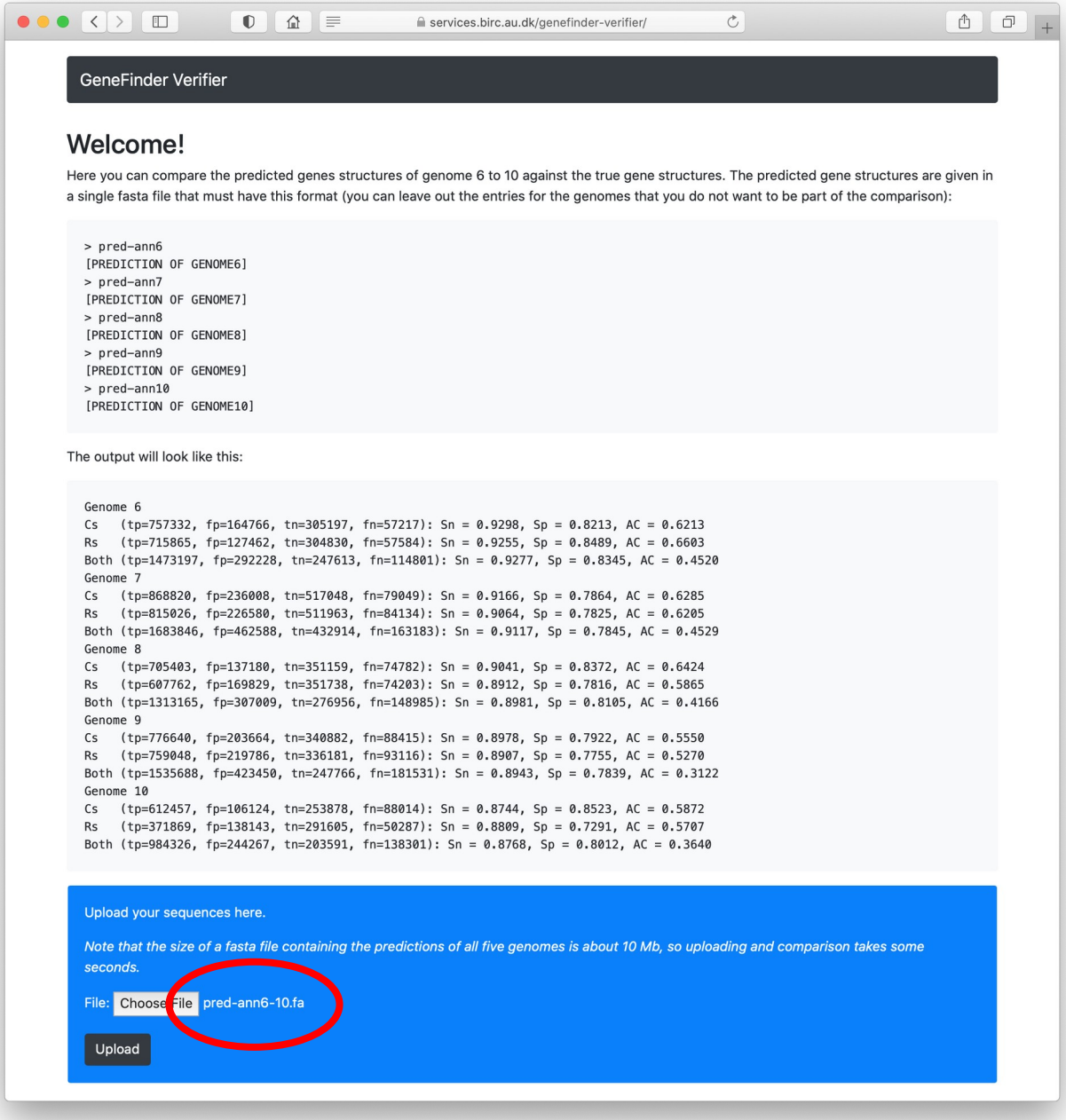
```
> pred-ann6
[PREDICTION OF GENOME6]
> pred-ann7
[PREDICTION OF GENOME7]
> pred-ann8
[PREDICTION OF GENOME8]
> pred-ann9
[PREDICTION OF GENOME9]
> pred-ann10
[PREDICTION OF GENOME10]
```

```
$ cat pred-ann6.fa pred-ann7.fa pred-ann8.fa pred-ann9.fa pred-ann10.fa > pred-ann6-10.fa
```

Beware of the newlines!

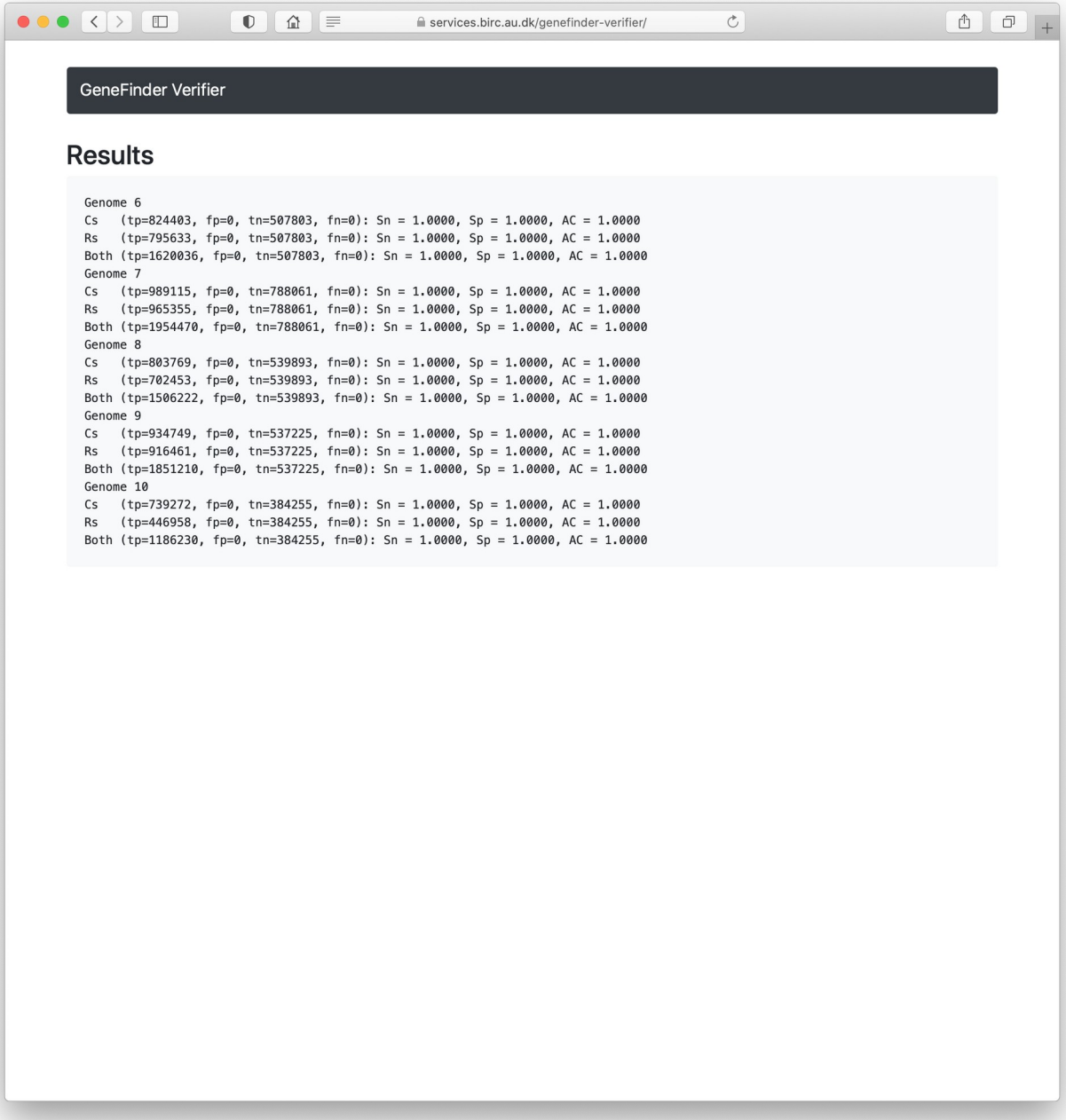
pred-annX.fa should end with a newline in order for the above to work

Report - Performance on genome 6-10



https://services.birc.au.dk/genefinder-verifier/

Report - Performance on genome 6-10



https://services.birc.au.dk/genefinder-verifier/

Previous results

ACs between predictions and true annotations sorted by average

	Genome 6	Genome 7	Genome 8	Genome 9	Genome 10	Average
3	0,6313	0,6378	0,6007	0,5310	0,4762	**0,5754**
22	0,6313	0,6378	0,6007	0,5310	0,4762	**0,5754**
9	0,6053	0,6147	0,5708	0,4919	0,4197	**0,5405**
10	0,4628	0,4587	0,4245	0,3244	0,3611	**0,4063**
21	0,4619	0,4582	0,4219	0,3239	0,3603	**0,4052**
13	0,4539	0,4547	0,4159	0,3121	0,3650	**0,4003**
11	0,4522	0,4504	0,4172	0,3139	0,3643	**0,3997**
1	0,4520	0,4529	0,4165	0,3122	0,3646	**0,3996**
25	0,4543	0,4515	0,4144	0,3133	0,3596	**0,3986**
18	0,4521	0,4481	0,4112	0,3095	0,3619	**0,3966**
15	0,4501	0,4449	0,4187	0,2974	0,3586	**0,3939**
24	0,4237	0,4326	0,4063	0,2769	0,3816	**0,3842**
14	0,4359	0,4383	0,3974	0,2642	0,3467	**0,3765**
2	0,4286	0,4548	0,3725	0,2473	0,3738	**0,3753**
19	0,3856	0,4075	0,3727	0,2926	0,3059	**0,3529**
23	0,3854	0,4073	0,3551	0,2336	0,3710	**0,3505**
5	0,3850	0,4008	0,3619	0,2559	0,3119	**0,3431**
6	0,3821	0,3914	0,3504	0,2365	0,3263	**0,3373**
16	0,2917	0,3462	0,2618	0,2240	0,3213	**0,2890**
7	0,0247	0,0978	0,0311	0,0642	0,1441	**0,0724**
4	0,0254	0,0939	0,0253	0,0631	0,1387	**0,0693**
20	0,0775	0,0526	0,1344	-0,0246	0,0765	**0,0633**
12	0,0344	0,0820	0,0089	0,0549	0,1144	**0,0589**
17	-0,2748	-0,2571	-0,2635	-0,2657	-0,3075	**-0,2737**
8						

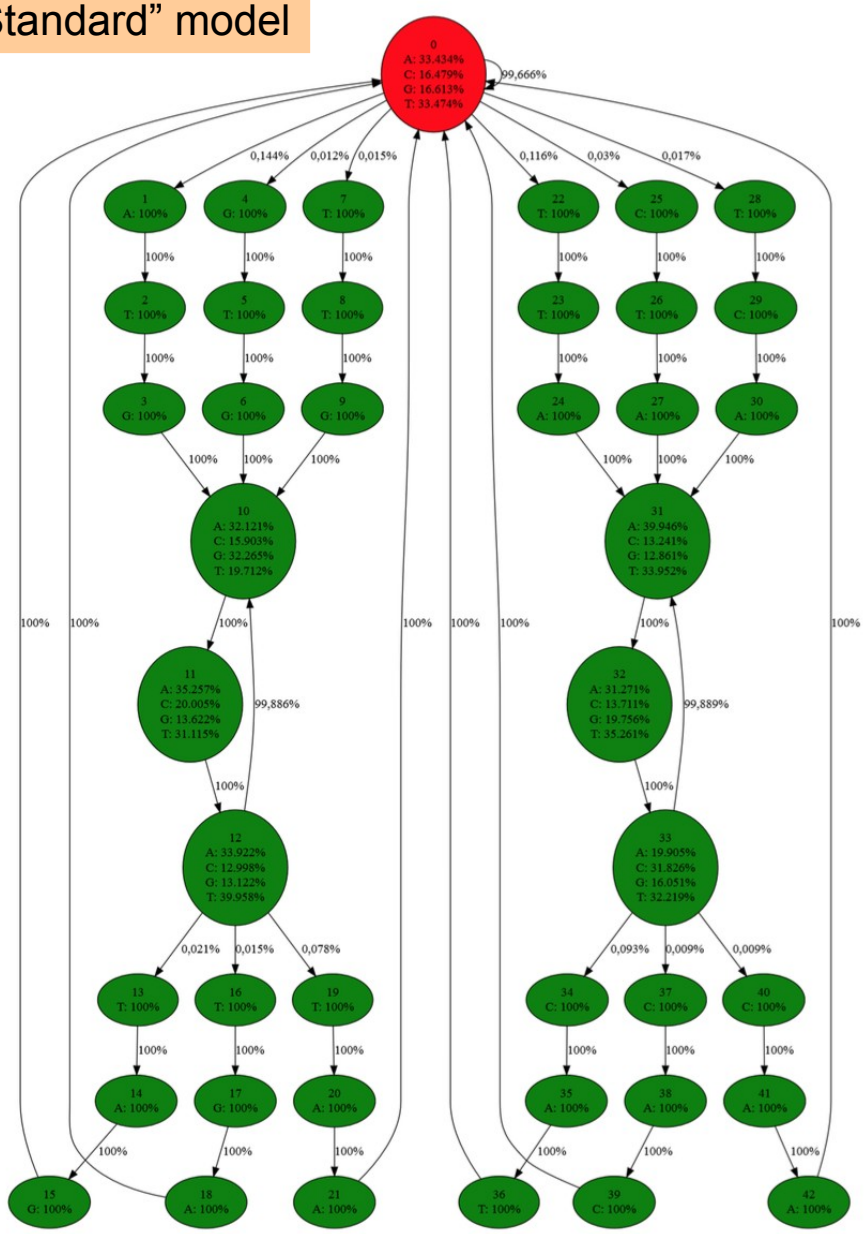
Codon-models

Models inspired by the model presented in class maybe allowing more start/stop codons.

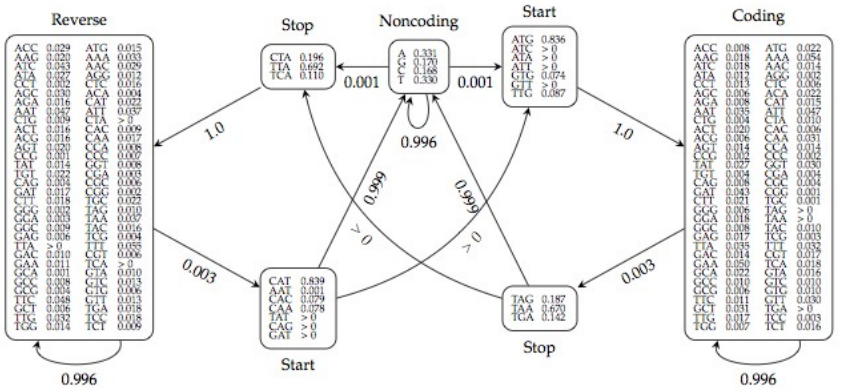
Problems with training or prediction

Examples of models

"Standard" model



"Codon" model



ACs between predictions and true annotations sorted by average

	Genome 6	Genome 7	Genome 8	Genome 9	Genome 10	Average
3	0,6313	0,6378	0,6007	0,5310	0,4762	**0,5754**
22	0,6313	0,6378	0,6007	0,5310	0,4762	**0,5754**
9	0,6053	0,6147	0,5708	0,4919	0,4197	**0,5405**
10	0,4628	0,4587	0,4245	0,3244	0,3611	**0,4063**
21	0,4619	0,4582	0,4219	0,3239	0,3603	**0,4052**
13	0,4539	0,4547	0,4159	0,3121	0,3650	**0,4003**
11						**0,3997**
1						**0,3996**
25						**0,3986**
18						**0,3966**
15						**0,3939**
24	0,4237	0,4326	0,4063	0,2769	0,3816	**0,3842**
14	0,4359	0,4383	0,3974	0,2642	0,3467	**0,3765**
2	0,4286	0,4548	0,3725	0,2473	0,3738	**0,3753**
19	0,3856	0,4075	0,3727	0,2926	0,3059	**0,3529**
23	0,3854	0,4073	0,3551	0,2336	0,3710	**0,3505**
5	0,3850	0,4008	0,3619	0,2559	0,3119	**0,3431**
6	0,3821	0,3914	0,3504	0,2365	0,3268	**0,3373**
16	0,2917	0,3462	0,2618	0,2240	0,3218	**0,2890**
7	0,0247	0,0978	0,0311	0,0642	0,1441	**0,0724**
4	0,0254	0,0939	0,0253	0,0631	0,1387	**0,0693**
20	0,0775	0,0526	0,1344	-0,0246	0,0765	**0,0633**
12	0,0344	0,0820	0,0089	0,0549	0,1144	**0,0589**
17	-0,2748	-0,2571	-0,2635	-0,2657	-0,3075	**-0,2737**
8						

Can you do better than this?

Codon-models

Models inspired by the model presented in class maybe allowing more start/stop codons.

Problems with training or prediction

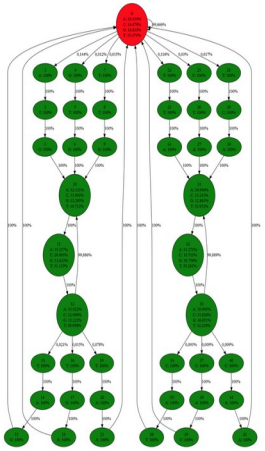
Running time in practice

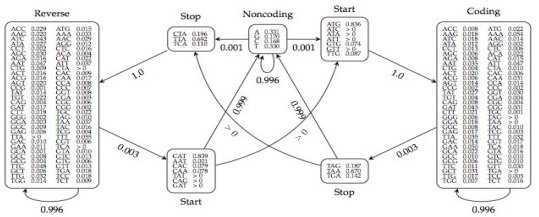
Using the **7-state "Codon-Model",** my implementation of Viterbi with backtracking on genome1.fa (length 1883386) takes **~23 sec** to run on a Macbook Pro.

A back-of-the-envelope calculation gives that it is

43^2 / 7^2 ≈ 38

times slower using the **43-state "standard model"** (since the running time of Viterbi is O(NK^2)). I would thus expect it to take about 38 * 23 = 874 sec ≈ **14,5 minutes** to run Viterbi with backtracking on genome1.fa.





Running time in practice

Using the **7-state "Codon-Model"**, my implementation of Viterbi with

Chr_CodonHMM — ⌥⌘1

```
[~/Dropbox/Work/Teaching/Classes/Current/ML/Code/2021/Chr_CodonHMM% time python codonhmm.py genome1.fa > pred1.fa
python codonhmm.py genome1.fa > pred1.fa  22.31s user 0.21s system 99% cpu 22.538 total
[~/Dropbox/Work/Teaching/Classes/Current/ML/Code/2021/Chr_CodonHMM%
[~/Dropbox/Work/Teaching/Classes/Current/ML/Code/2021/Chr_CodonHMM% python compare_anns.py true-ann1.fa pred1.fa
Cs   (tp=713369, fp=132703, tn=286808, fn=14600): Sn = 0.9799, Sp = 0.8432, AC = 0.7292
Rs   (tp=605003, fp=99958, tn=287970, fn=13438): Sn = 0.9783, Sp = 0.8582, AC = 0.7671
Both (tp=1318372, fp=232661, tn=273370, fn=28038): Sn = 0.9792, Sp = 0.8500, AC = 0.6382
[~/Dropbox/Work/Teaching/Classes/Current/ML/Code/2021/Chr_CodonHMM%
[~/Dropbox/Work/Teaching/Classes/Current/ML/Code/2021/Chr_CodonHMM% python
Python 3.9.7 | packaged by conda-forge | (default, Sep 29 2021, 19:24:02)
[Clang 11.1.0 ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> 43**2 / 7**2
37.734693877551024
[>>>
[>>> 43**2 / 7**2 * 23
867.8979591836735
[>>>
[>>> 43**2 / 7**2 * 23 / 60
14.464965986394558
[>>>
>>>
```