Sophia Ju – 260852000
COMP 462 – HW3
Instructions on how to run all code is in src/README.md – need to pip install biopython

Q1.a) All results for this question are also in src/config.json file

Average lengths:

genic: 990.4515103338633
intergenic: 1051.5917566241412

Nucleotide frequencies:

"A": 0.2661070072218186,
"C": 0.24333380461567392,
"G": 0.22576010608985442,
"T": 0.26479908207265307

Start codons:

"AAA": 0.0,	"AAT": 0.0,	"AAG": 0.0,	"AAC": 0.0,	"ATA": 0.0,	"ATT": 0.0,	"ATG": 0.88765235824059 35,	"ATC": 0.0,
"AGA": 0.0,	"AGT": 0.0,	"AGG": 0.0,	"AGC": 0.0,	"ACA": 0.0,	"ACT": 0.0,	"ACG": 0.0,	"ACC": 0.0,
"TAA": 0.0,	"TAT": 0.0,	"TAG": 0.0,	"TAC": 0.0,	"TTA": 0.0,	"TTT": 0.0,	"TTG": 0.0376258 6115527292,	"TTC": 0.0,
"TGA": 0.0,	"TGT": 0.0,	"TGG": 0.0,	"TGC": 0.0,	"TCA": 0.0,	"TCT": 0.0,	"TCG": 0.0,	"TCC": 0.0,
"GAA": 0.0,	"GAT": 0.0,	"GAG": 0.0,	"GAC": 0.0,	"GTA": 0.0,	"GTT": 0.0,	"GTG": 0.0747217 8060413355,	"GTC": 0.0,
"GGA": 0.0,	"GGT": 0.0,	"GGG": 0.0,	"GGC": 0.0,	"GCA": 0.0,	"GCT": 0.0,	"GCG": 0.0,	"GCC": 0.0,
"CAA": 0.0,	"CAT": 0.0,	"CAG": 0.0,	"CAC": 0.0,	"CTA": 0.0,	"CTT": 0.0,	"CTG": 0.0,	"CTC": 0.0,
"CGA": 0.0,	"CGT": 0.0,	"CGG": 0.0,	"CGC": 0.0,	"CCA": 0.0,	"CCT": 0.0,	"CCG": 0.0,	"CCC": 0.0

Middle codons:

"AAA": 0.03594262473602442,	"AAT": 0.01905464072701577,	"AAG": 0.013646872060776092,	"AAC": 0.020269452721549924,	"ATA": 0.0035508648073966404,	"ATT": 0.03106724379647301,	"ATG": 0.023487978523801765,	"ATC": 0.025304549979107815,
"AGA": 0.0025344882382644807,	"AGT": 0.011565720038267384,	"AGG": 0.0008389139935694016,	"AGC": 0.014034063134731201,	"ACA": 0.0075728120881053285,	"ACT": 0.013012846677174603,	"ACG": 0.011301139471064727,	"ACC": 0.02063405764952432,
"TAA": 0.0,	"TAT": 0.015757063413831433,	"TAG": 0.0,	"TAC": 0.014271217667528705,	"TTA": 0.01936439358617986,	"TTT": 0.02574336652959027,	"TTG": 0.02261679860740277,	"TTC": 0.01374689642154783,
"TGA": 0.0,	"TGT": 0.00570622845241341,	"TGG": 0.013140297072351492,	"TGC": 0.004138104602895222,	"TCA": 0.010492878104183439,	"TCT": 0.011183368852736715,	"TCG": 0.009190948118009386,	"TCC": 0.005838518736014739,
"GAA": 0.03927569456432131,	"GAT": 0.037388138078790156,	"GAG": 0.024339798886503002,	"GAC": 0.014484172758204013,	"GTA": 0.011059145049842784,	"GTT": 0.016592750815117876,	"GTG": 0.028448864158851592,	"GTC": 0.014487399350486974,
"GGA": 0.007446974989069918,	"GGT": 0.02697108489325626,	"GGG": 0.008615001395501162,	"GGC": 0.025244858021873068,	"GCA": 0.019390206324443535,	"GCT": 0.02106480771929938,	"GCG": 0.030596161323160963,	"GCC": 0.022297365971389805,
"CAA": 0.033346831244383715,	"CAT": 0.012793438401933374,	"CAG": 0.018444814785536476,	"CAC": 0.010667114087463236,	"CTA": 0.008844089447591269,	"CTT": 0.012788598513508935,	"CTG": 0.029221633010620328,	"CTC": 0.01458097052669279,
"CGA": 0.005231919386818403,	"CGT": 0.020313011717369874,	"CGG": 0.002803908693891577,	"CGC": 0.01763171353023075,	"CCA": 0.012799891586499293,	"CCT": 0.011254353882961818,	"CCG": 0.010749392190678697,	"CCC": 0.0058175458861755504

stop codons:

"AAA": 0.0,	"AAT": 0.0,	"AAG": 0.0,	"AAC": 0.0,	"ATA": 0.0,	"ATT": 0.0,	"ATG": 0.0,	"ATC": 0.0,
"AGA": 0.0,	"AGT": 0.0,	"AGG": 0.0,	"AGC": 0.0,	"ACA": 0.0,	"ACT": 0.0,	"ACG": 0.0,	"ACC": 0.0,
"TAA": 0.643879173290938,	"TAT": 0.0,	"TAG": 0.166401695813460 5,	"TAC": 0.0,	"TTA": 0.0,	"TTT": 0.0,	"TTG": 0.0,	"TTC": 0.0,
"TGA": 0.189719130895601 5,	"TGT": 0.0,	"TGG": 0.0,	"TGC": 0.0,	"TCA": 0.0,	"TCT": 0.0,	"TCG": 0.0,	"TCC": 0.0,
"GAA": 0.0,	"GAT": 0.0,	"GAG": 0.0,	"GAC": 0.0,	"GTA": 0.0,	"GTT": 0.0,	"GTG": 0.0,	"GTC": 0.0,
"GGA": 0.0,	"GGT": 0.0,	"GGG": 0.0,	"GGC": 0.0,	"GCA": 0.0,	"GCT": 0.0,	"GCG": 0.0,	"GCC": 0.0,
"CAA": 0.0,	"CAT": 0.0,	"CAG": 0.0,	"CAC": 0.0,	"CTA": 0.0,	"CTT": 0.0,	"CTG": 0.0,	"CTC": 0.0,
"CGA": 0.0,	"CGT": 0.0,	"CGG": 0.0,	"CGC": 0.0,	"CCA": 0.0,	"CCT": 0.0,	"CCG": 0.0,	"CCC": 0.0

Q1.b & c) Instructions on how to run all code is in src/README.md

For this section:

run “python **viterbi.py** -f fasta.file -c config.json -o output.file”
exact command I used is:
“python **viterbi.py** -f ../data/Vibrio_vulnificus.ASM74310v1.dna.toplevel.fa -c config.json -o vulnificus/predictions.gff3”

The GFF3 file with the gene predictions is **vulnificus/predictions.gff3**

Q1.d)

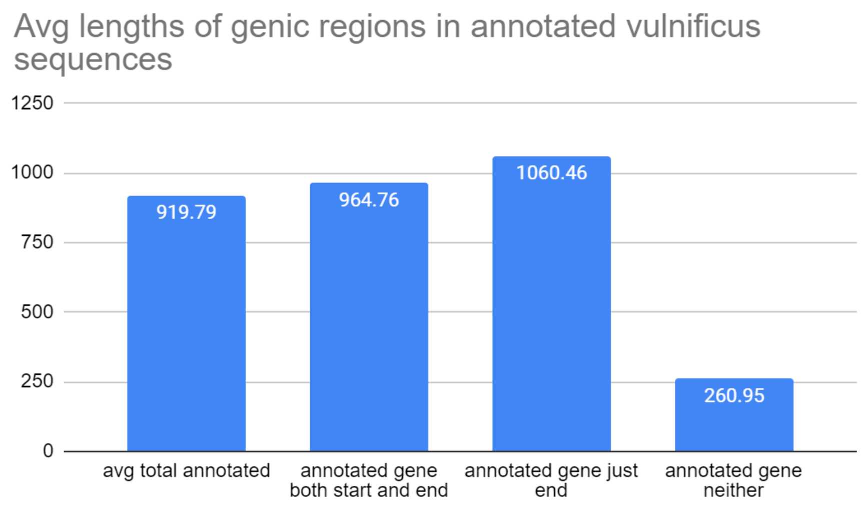
fraction of annotated genes that:

perfectly match both ends of one of predicted genes:	0.37605126151381657
match the start but not the end of a predicted gene:	0.0
match the end but not the start of a predicted gene:	0.45694833800560675
do not match either the start or end of a predicted gene:	0.16700040048057668

fraction of predicted genes that:

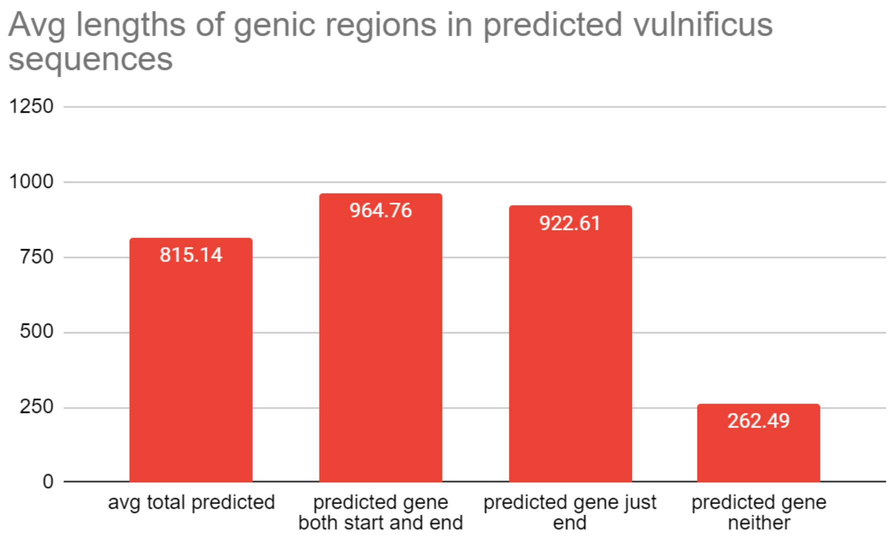
perfectly match both ends of one of annotated genes:	0.3626882966396292
match the start but not the end of an annotated gene:	0.0
match the end but not the start of an annotated gene:	0.44071069911162614
do not match either the start or end of an annotated gene:	0.1966010042487447

Q1.e) What properties of annotated genes are associated to an elevated risk of being partially or completely missed by your predictor?



I first calculated the average lengths of annotated genes for each category that had non-zero frequency in the previous question. Here, we can see that the annotated genes that have no match to the start or end of a predicted gene are on average much shorter in length. This is probably because we calculate transition probabilities using a much higher genic length, and so my model cannot easily find shorter genes. Additionally, from the previous question we know that the model found no annotated genes that matched the start but not the end of a predicted gene, but in many cases found just the end. This leads me to believe that when my model predicts a gene, it will sometimes mess up finding the start of the gene especially if the gene is shorter in length, but will always be able to predict the end.

What are the properties of genes predicted by your predictor that do not match an annotated gene?



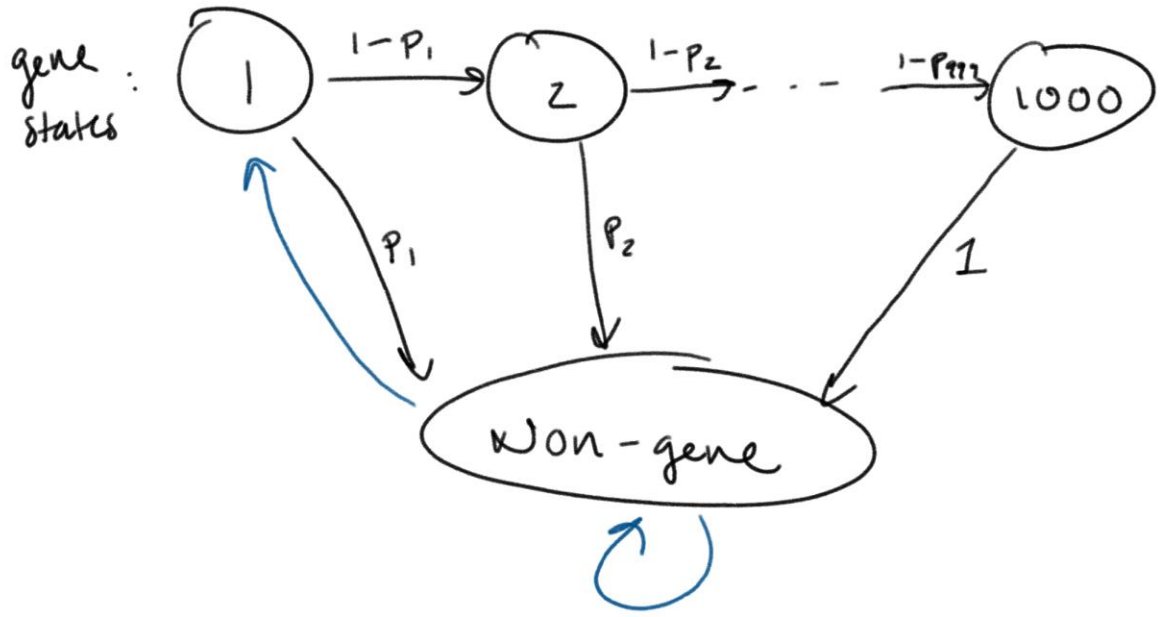
I found similar results when calculating the average lengths for predicted genes of each category. The predicted genes were about 100bp shorter on average, but followed a similar pattern as the annotated genes, suggesting that my model does okay with longer predictions but messes up often with shorter predictions. I also investigated the start and stop codon frequencies for the predicted genes that did not match the start or end of the annotated genes:

Start codons:	Emission freq	Mismatch predicted freq	Stop codons:	Emission freq	Mismatch predicted freq
ATG	0.8876	0.7075	TAA	0.6438	0.5376
TTG	0.0376	0.1785	TAG	0.1664	0.1935
GTG	0.0747	0.1140	TGA	0.1897	0.2688

From looking at the start and stop codon frequencies, mismatched predicted genes had higher frequencies of TTG or GTG as a start codon (as opposed to ATG) compared to the original frequencies from Q1.a that we used for the emission probabilities. Mismatched predicted genes also had higher frequencies of TAG and TGA as a stop codon (as opposed to TAA) compared to the emission probabilities.

Q2) For this question, I assume that since we only have distribution for up to length 1000, the maximum length of a gene region in this model is 1000bp, and that after that the model must return to the non-gene state. I also assume that since you want the distribution over the duration of stay in the gene state to be the exact target length distribution, I should not split up the gene state by codon as we did for the HMM in Q1, and the observations are all individual nucleotides.

We could modify the given HMM by splitting the Gene state into 1000 different gene substates, one for each length from 1 to 1000. The non-gene state will only be able to transition itself and to the gene-length-1 state, and the gene-length-1 state will only be able to transition to non-gene or gene-length-2 and so on and so forth until gene-length-1000, which will only be able to transition to the non-gene state. The transition probabilities from each gene-length-n state to the non-gene state would be Pr[length = n] = p_n, and the transition probabilities from a gene-length-n state to a gene-length-(n+1) state would be 1- $p_{n.}$ An exception would be for the gene-length-1000 state, which would have transmission probability = 1 to the non-gene state.


gene states :
1
2
1000
$1-p_1$
$1-p_2$
$1-p_{999}$
p_1
p_2
1
Non-gene