CS 271: Topics in Machine Learning

Assignment 1

Name: Mrunal Kotkar

SJSU ID: 018681091

Date: 09/02/2025

---

1. Problem 2.1

Solution:

a) Direct Computation:

Based on the observation sequence O = (1,0,2) and X = (X0,X1,X2).

$$P(X, \mathcal{O}) = \pi_{x_0}\, b_{x_0}(O_0)\, a_{x_0,x_1}\, b_{x_1}(O_1)\, a_{x_1,x_2}\, b_{x_2}(O_2)$$

Based on the formula above,

P($\mathcal{O}$, X=HHH) = (0.0)·(0.4)·(0.7)·(0.1)·(0.7)·(0.5) = 0
P($\mathcal{O}$, X=HHC) = (0.0)·(0.4)·(0.7)·(0.1)·(0.3)·(0.1) = 0
P($\mathcal{O}$, X=HCH) = (0.0)·(0.4)·(0.3)·(0.7)·(0.4)·(0.5) = 0
P($\mathcal{O}$, X=HCC) = (0.0)·(0.4)·(0.3)·(0.7)·(0.6)·(0.1) = 0
P($\mathcal{O}$, X=CHH) = (1.0)·(0.2)·(0.4)·(0.1)·(0.7)·(0.5) = 0.0028
P($\mathcal{O}$, X=CHC) = (1.0)·(0.2)·(0.4)·(0.1)·(0.3)·(0.1) = 0.00024
P($\mathcal{O}$, X=CCH) = (1.0)·(0.2)·(0.6)·(0.7)·(0.4)·(0.5) = 0.0168
P($\mathcal{O}$, X=CCC) = (1.0)·(0.2)·(0.6)·(0.7)·(0.6)·(0.1) = 0.00504

To calculate the desired probability, we take sum of all the eight calculated probabilities.

$$P(\mathcal{O} \mid \lambda) = \sum_{X \in \mathcal{X}} P(\mathcal{O}, X \mid \lambda)$$

P(O| λ) = 0 + 0 + 0 + 0 + 0.0028 + 0.00024 + 0.0168 + 0.00504 = **0.02488**

b) Using Alpha Pass:

Based on the formula of alpha pass, the initial

$$\alpha_0(i) = \pi_i b_i(\mathcal{O}_0), \text{ for } i = 0, 1, \ldots, N-1$$

and recurrence

$$\alpha_t(i) = \left(\sum_{j=0}^{N-1} \alpha_{t-1}(j) a_{ji} \right) b_i(\mathcal{O}_t)$$

α0(0) = π0.b0(O0) = (0.0).(0.4) = 0
α0(1) = π1.b1(O0) = (1.0).(0.2) = 0.2
α1(0) = (α0(0).a00 + α0(1).a10).b0(O1) = [(0.0).(0.7) + (0.2).(0.4)].(0.1) = 0.008
α1(1) = (α0(0).a01 + α0(1).a11).b1(O1) = [(0.0).(0.3) + (0.2).(0.6)].(0.7) = 0.084
α2(0) = (α1(0).a00 + α1(1).a10).b0(O2) = [(0.008).(0.7) + (0.084).(0.4)].(0.5) = 0.0196
α2(1) = (α1(0).a01 + α1(1).a11).b1(O2) = [(0.008).(0.3) + (0.084).(0.6)].(0.1) = 0.00528

The desired probability is calculated by adding these values

$$P(\mathcal{O} \mid \lambda) = \sum_{i=0}^{N-1} \alpha_{T-1}(i).$$

P(O| λ) = α2(0) + α2(1) = 0.0196 + 0.00528 = **0.02488**

c) Work factor for method A = 2TN^T = 2*3*2^3 = 2*3*8 = 48
Work factor for method B = N^2T = 2^2*3 = 4*3 = 12
Hence, forward algorithm i.e. method B is more suitable and efficient.

2. Problem 2.3
Solution:

a) <u>Direct Calculation:</u>
As per the problem, the observation symbols are {0,1,2}. Hence M = 3
The observation sequence O is of length 4. Hence T = 4
Total possible observation sequences = M^T = 3^4 = 81.

Attached file **Question3A.cpp** is the corresponding code solution for this question.

Basic code flow is like this:

$$P(X) = \pi_{x_0} b_{x_0}(\mathcal{O}_0) a_{x_0,x_1} b_{x_1}(\mathcal{O}_1) a_{x_1,x_2} b_{x_2}(\mathcal{O}_2) a_{x_2,x_3} b_{x_3}(\mathcal{O}_3)$$

Based on the formula above,
First state is calculated as = πx0*bx0(O0)
For others recursive formula is applied = Ax(t-1),x(t) * Bx(t)(Ot) for t = 1 to T-1
Finally, all the values are multiplied and it gives the
P(X,O) = Probability of a state for given observation sequence.
We sum these up to calculate the
P(O) = To find probability of all states for given observation sequence.
The total probability sums up to 1.

Steps to run the code (on windows machine):
(1) g++ Question3A.cpp -o Question3A
(2) Question3A.exe

Output: The output of the code is in the file **Question3AOutput.txt**

b) <u>Using forward algorithm:</u>
Attached file **Question3B.cpp** is the corresponding code solution for this question.
Output: The output of the code is in the file **Question3BOutput.txt**

Both the values from method A and method B match. Also, the final sums up to 1 in both the cases.

3. Problem 2.10
Solution: Used the same code across by just modifying the value of N.

In this problem, based on the information in section 2.9 of textbook,
M = 27 (all lowercase symbols and space)
N = 2 (will start from lowest number of states) this will vary for the other parts
T = 50000 (consecutive observations from the Brown Corpus)
A, B, pi matrices are initialized at first using 1/N, 1/M, 1/N approximate values respectively.
Iterations = 100

a) For N=2, Final logProb = -136423.46191
The output is stored under file **10AOutput.txt**
Based on the Final B Matrix and highest probabilities,
State 1 is most likely – vowels (a,e,i,o,u) and space
State 2 is most likely – consonants (c,d,f,g,h,s,t,etc)

```
Final B^T:
a 0.09818 0.07692
b 0.00000 0.02172
c 0.00210 0.05141
d 0.00000 0.06455
e 0.21053 0.00000
f 0.00000 0.03325
g 0.00053 0.02587
h 0.00074 0.06790
i 0.12106 0.00000
j 0.00000 0.00345
k 0.00100 0.00690
l 0.00368 0.06502
m 0.00000 0.03647
n 0.00000 0.10773
o 0.12997 0.00000
p 0.00056 0.03474
q 0.00000 0.00138
r 0.00000 0.09562
s 0.00000 0.10417
t 0.00962 0.13639
u 0.04432 0.00000
v 0.00000 0.01509
w 0.00000 0.02142
x 0.00000 0.00433
y 0.00001 0.02432
z 0.00000 0.00100
  0.37771 0.00034
```

10A Output

```
Final B^T:
a 0.13851 0.00851 0.09905
b 0.03199 0.00011 0.00000
c 0.07450 0.00416 0.00087
d 0.04300 0.06463 0.00000
e 0.00644 0.20106 0.11323
f 0.01809 0.03831 0.00000
g 0.01701 0.02706 0.00000
h 0.00654 0.11527 0.00137
i 0.03558 0.00142 0.12457
j 0.00507 0.00003 0.00000
k 0.00397 0.00934 0.00001
l 0.04616 0.06415 0.00284
m 0.03562 0.01882 0.00295
n 0.07433 0.07687 0.02210
o 0.05250 0.06424 0.06988
p 0.04167 0.01024 0.00206
q 0.00124 0.00099 0.00000
r 0.08530 0.06284 0.00495
s 0.08275 0.08776 0.00002
t 0.13875 0.09339 0.00007
u 0.00772 0.00097 0.05045
v 0.01679 0.00680 0.00000
w 0.03161 0.00002 0.00000
x 0.00153 0.00093 0.00404
y 0.00187 0.04206 0.00000
z 0.00146 0.00001 0.00000
  0.00000 0.00003 0.50156
```

10B Output

```
Final B^T:
a 0.09589 0.08578 0.08270 0.08363
b 0.01087 0.01062 0.01194 0.01197
c 0.02992 0.02641 0.02760 0.02739
d 0.03125 0.03263 0.03987 0.03118
e 0.09355 0.10353 0.09749 0.10812
f 0.01647 0.01649 0.02099 0.01555
g 0.01353 0.01240 0.01569 0.01347
h 0.03819 0.03713 0.03189 0.03585
i 0.06022 0.05652 0.06110 0.05354
j 0.00164 0.00171 0.00222 0.00163
k 0.00431 0.00373 0.00384 0.00444
l 0.03937 0.03769 0.03327 0.03226
m 0.01796 0.01807 0.02106 0.01915
n 0.05477 0.05576 0.06536 0.04914
o 0.05817 0.06414 0.06751 0.05875
p 0.01716 0.01754 0.01558 0.02340
q 0.00059 0.00065 0.00092 0.00073
r 0.04974 0.04512 0.04730 0.05769
s 0.05744 0.05000 0.04967 0.06043
t 0.07777 0.08132 0.06912 0.07474
u 0.01873 0.02192 0.02256 0.02159
v 0.00770 0.00700 0.00867 0.00819
w 0.01215 0.01073 0.01102 0.01080
x 0.00236 0.00209 0.00185 0.00275
y 0.01265 0.01263 0.01256 0.01296
z 0.00053 0.00056 0.00052 0.00048
  0.17704 0.18782 0.17770 0.18017
```

10C Output

b) For N=3, Final logProb = -133837.79557
The output is stored under file **10BOutput.txt**
Based on the Final B Matrix and high probabilities,
State 1 is most likely – high frequency starting letters (a & t)
State 2 is most likely – middle parts of the words (e,h,s,r,n,o)
State 3 is most likely – spaces

c) For N=4, Final logProb = -140956.52691
The output is stored under file **10COutput.txt**
Based on the Final B Matrix and high probabilities,
State 1 is most likely – short word sequences or start of words
State 2 is most likely – spaces, a, s (most repeating characters)
State 3 is most likely – common sequences inside words (in, on)
State 4 is most likely – ending of word sequences (er, st)

```
Final B^T:
a 0.09358 0.09329 0.08209 0.09642 0.06845 0.07956 0.07806 0.08249 0.09070 0.06859 0.09663 0.08594 0.08524 0.08529 0.09020 0.07183 0.12926 0.07507 0.09218
0.08633 0.09006 0.07921 0.09353 0.09194 0.09338 0.08778 0.08386
b 0.01405 0.01258 0.01055 0.01115 0.00866 0.01018 0.01141 0.01283 0.01254 0.00902 0.01289 0.01320 0.01002 0.01150 0.01165 0.00762 0.01202 0.01058 0.00965
0.01256 0.01042 0.01156 0.01094 0.01246 0.01261 0.01298 0.01074
c 0.03140 0.03036 0.02646 0.02741 0.02816 0.02569 0.02780 0.03164 0.02764 0.02358 0.02806 0.02790 0.02816 0.02876 0.02579 0.02436 0.02811 0.03072 0.02699
0.02801 0.02848 0.02643 0.02725 0.02886 0.02974 0.02839 0.02569
d 0.03214 0.03611 0.03085 0.03122 0.02970 0.03695 0.03528 0.03687 0.03068 0.02684 0.02937 0.03687 0.03354 0.03938 0.03534 0.03137 0.03977 0.03063 0.03962
0.03695 0.03360 0.03731 0.03061 0.02940 0.03322 0.03592 0.03123
e 0.08564 0.09647 0.09964 0.09180 0.10754 0.10020 0.09084 0.10414 0.08932 0.09814 0.09000 0.10463 0.10554 0.11030 0.10515 0.11006 0.10661 0.10772 0.10938
0.09848 0.10442 0.10435 0.11229 0.09718 0.09078 0.08997 0.10592
f 0.01818 0.02017 0.01583 0.01582 0.01404 0.01697 0.01901 0.01887 0.01700 0.01454 0.01830 0.01741 0.01623 0.01848 0.01652 0.01510 0.02005 0.01690 0.01762
0.02048 0.01698 0.02012 0.01527 0.01713 0.01898 0.01646 0.01663
g 0.01538 0.01489 0.01367 0.01414 0.01156 0.01309 0.01309 0.01512 0.01340 0.01166 0.01311 0.01522 0.01267 0.01528 0.01395 0.01297 0.01396 0.01489 0.01372
0.01520 0.01257 0.01470 0.01368 0.01381 0.01515 0.01315 0.01172
h 0.02994 0.03138 0.03649 0.02966 0.03909 0.03908 0.03595 0.03682 0.03473 0.03627 0.03176 0.03858 0.04203 0.04073 0.04037 0.03662 0.03022 0.03370 0.03868
0.03404 0.03473 0.04308 0.03752 0.03332 0.03186 0.03819 0.03213
i 0.05526 0.05497 0.05656 0.06093 0.05724 0.05968 0.06224 0.05883 0.05399 0.05978 0.05412 0.06101 0.06237 0.05326 0.05996 0.05713 0.05349 0.05801 0.05553
0.05157 0.05768 0.05591 0.06526 0.05932 0.06349 0.05572 0.05893
j 0.00210 0.00180 0.00153 0.00181 0.00118 0.00169 0.00186 0.00196 0.00199 0.00140 0.00202 0.00229 0.00172 0.00189 0.00174 0.00129 0.00208 0.00176 0.00145
0.00193 0.00159 0.00201 0.00171 0.00205 0.00214 0.00214 0.00149
k 0.00416 0.00395 0.00355 0.00376 0.00367 0.00399 0.00430 0.00436 0.00441 0.00385 0.00383 0.00428 0.00401 0.00445 0.00460 0.00371 0.00424 0.00396 0.00407
0.00427 0.00442 0.00432 0.00389 0.00418 0.00429 0.00421 0.00350
l 0.03904 0.03489 0.03163 0.03405 0.03702 0.03788 0.03813 0.03346 0.03590 0.03762 0.03319 0.03936 0.03997 0.03424 0.03681 0.03479 0.03394 0.03661 0.03554
0.03532 0.03744 0.03449 0.03234 0.03397 0.03410 0.03765 0.03478
m 0.02028 0.01937 0.01780 0.01799 0.01763 0.01896 0.02099 0.01986 0.01815 0.01908 0.02166 0.01840 0.01710 0.01942 0.02032 0.01713 0.01742 0.01963 0.01737
0.01827 0.01856 0.01989 0.01919 0.02050 0.01996 0.01944 0.01985
n 0.05170 0.05945 0.05844 0.05780 0.05987 0.05538 0.05594 0.05907 0.05336 0.06019 0.05095 0.05220 0.05918 0.05898 0.05434 0.05677 0.05071 0.05633 0.05136
0.06365 0.06193 0.06297 0.05114 0.05253 0.05511 0.05383 0.05558
o 0.06479 0.05803 0.05949 0.06087 0.05645 0.06079 0.06935 0.06062 0.06363 0.05948 0.06582 0.06044 0.06690 0.06369 0.06345 0.06063 0.06174 0.06345 0.06036
0.05929 0.06366 0.06584 0.06569 0.05782 0.06424 0.05733 0.06384
p 0.01973 0.01734 0.01748 0.01826 0.01612 0.01798 0.02055 0.01970 0.02043 0.01653 0.02197 0.02249 0.01671 0.02040 0.01737 0.01387 0.02077 0.01679 0.01618
0.01941 0.01635 0.01788 0.01681 0.01900 0.02000 0.01828 0.01865
```

10D Output

d) For N=27, Final logProb = -140952.38423
The output is stored under file **10DOutput.txt**

4. Problem 2.11
Solution:

a) Question11A.cpp is the code for this question.
   (1) First read 50000 characters from BrownCorpus file. This should be only alphabets. They are converted into lower case and then stored into **plaintext** file. All numbers, other symbols and spaces are ignored.
   (2) I have considered the key for encryption as 18. While reading characters, the cipher conversion for the same is stored in **ciphertext** file.
   (3) Based on this encryption (SHIFT = 18), this is the encryption table.

Plaintext	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
ciphertext	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r

   (4) Above conversion tells that, the vowels maps to
   a → s, e → w, i → a, o → g, u → m
   (5) The output files are generated in the same folder – **plaintext.txt, ciphertext.txt**

b) I have used the same hmm code from 2.10
N = 2
M = 26
T = 50000 (length of ciphertext)
Iterations = 200
While running the code, input file provided was ciphertext.
To run the code = Question11B.exe ciphertext.txt
The output is stored under 11BOutput.txt
Iteration 1 logProb=-162139.10651 → Iteration 200 logProb=-141615.20645
Based on the final B Matrix given below, the maximum probabilities again defines 2 hidden states – vowels and consonants.
The characters with highest probabilities represent ciphertext letters that corresponds to the vowels from plaintext.
Vowels = a,g,s,w – which as we know corresponds to vowels from plaintext (i,o,a,e)
Others we can consider as another hidden state i.e. consonants.

```
Final B^T:
a 0.11307 0.02894
b 0.00114 0.00300
c 0.00457 0.00562
d 0.02388 0.06267
e 0.00700 0.04006
f 0.02465 0.11028
g 0.13248 0.02007
h 0.01584 0.02771
i 0.00091 0.00077
j 0.01958 0.10160
k 0.04165 0.08983
l 0.08710 0.09937
m 0.03124 0.02001
n 0.00089 0.01874
o 0.00570 0.02274
p 0.00054 0.00454
q 0.01621 0.01429
r 0.00005 0.00134
s 0.17065 0.04177
t 0.00741 0.02009
u 0.01320 0.05370
v 0.02701 0.05403
w 0.21108 0.03789
x 0.00639 0.03527
y 0.01579 0.01791
z 0.02200 0.06776
```

11B Output

c) Again, from BrownCorpus, 1,000,000 characters are read (only lowercase letters) and digraph frequency matrix A is generated.
Code – Question11C.cpp
Command – g++ Question11C.cpp -o Question11C
Question11C.exe
Output is stored in 11COutput.txt

```
{
  {0.020318, 0.026740, 0.047397, 0.039540, 0.004099, 0.016241, 0.022882, 0.009490, 0.039079, 0.002597, 0.010433, 0.093710, 0.036318, 0.171562, 0.009863,
0.022784, 0.000679, 0.103671, 0.094882, 0.148219, 0.012274, 0.019255, 0.015079, 0.002389, 0.028997, 0.001501},
  {0.100728, 0.024486, 0.016399, 0.010648, 0.218348, 0.011861, 0.004538, 0.014332, 0.064920, 0.005796, 0.001752, 0.089541, 0.013119, 0.007188, 0.101402,
0.012310, 0.000674, 0.048387, 0.031045, 0.059889, 0.079252, 0.003504, 0.019993, 0.000225, 0.059349, 0.000314},
  {0.135266, 0.008445, 0.024494, 0.006313, 0.136640, 0.005752, 0.002329, 0.146797, 0.067282, 0.001291, 0.032547, 0.034819, 0.006706, 0.003928, 0.184030,
0.006594, 0.001122, 0.035156, 0.015937, 0.100558, 0.025925, 0.001543, 0.008137, 0.000196, 0.007912, 0.000281},
  {0.125616, 0.049482, 0.033063, 0.028663, 0.156241, 0.021749, 0.015639, 0.025344, 0.117520, 0.004325, 0.001760, 0.018857, 0.026702, 0.016117, 0.074701,
0.017902, 0.001735, 0.031354, 0.056924, 0.093257, 0.030700, 0.006437, 0.029242, 0.000126, 0.016217, 0.000327},
  {0.092317, 0.025804, 0.053652, 0.086211, 0.035990, 0.022938, 0.014107, 0.014148, 0.029119, 0.002476, 0.004636, 0.042320, 0.036928, 0.100816, 0.024575,
0.027499, 0.003124, 0.140785, 0.108035, 0.057141, 0.006954, 0.019307, 0.026834, 0.011498, 0.012254, 0.000532},
  {0.109212, 0.015097, 0.025070, 0.013192, 0.081920, 0.058573, 0.007752, 0.016547, 0.108804, 0.003491, 0.001813, 0.025705, 0.014779, 0.008251, 0.183607,
0.013419, 0.000680, 0.079518, 0.021897, 0.153187, 0.037265, 0.002221, 0.011878, 0.000227, 0.005486, 0.000408},
  {0.122489, 0.026086, 0.021616, 0.013096, 0.160776, 0.012307, 0.014673, 0.112023, 0.086463, 0.002367, 0.001052, 0.029347, 0.013148, 0.029136, 0.092721,
0.010939, 0.000631, 0.094983, 0.035185, 0.067003, 0.031556, 0.001999, 0.013411, 0.000263, 0.006469, 0.000263},
  {0.166832, 0.007542, 0.009541, 0.004375, 0.465933, 0.003187, 0.001821, 0.005364, 0.124312, 0.000752, 0.000534, 0.003523, 0.006037, 0.006889, 0.092521,
0.003642, 0.000178, 0.018726, 0.009858, 0.038679, 0.015183, 0.000990, 0.007364, 0.000099, 0.005998, 0.000119},
  {0.031280, 0.010656, 0.077385, 0.037640, 0.041234, 0.017774, 0.028654, 0.001306, 0.000856, 0.000351, 0.005251, 0.054852, 0.030213, 0.248105, 0.077006,
0.009968, 0.000870, 0.035744, 0.127043, 0.121876, 0.001446, 0.029848, 0.001530, 0.002036, 0.000112, 0.006964},
  {0.153202, 0.005911, 0.005419, 0.004433, 0.152709, 0.002956, 0.003941, 0.005911, 0.029064, 0.003448, 0.002463, 0.002956, 0.003941, 0.002956, 0.288670,
0.002956, 0.002463, 0.027094, 0.004926, 0.003448, 0.275862, 0.002463, 0.005419, 0.002463, 0.002463, 0.002463},
  {0.087480, 0.019092, 0.020188, 0.011268, 0.328638, 0.013459, 0.005321, 0.030360, 0.141471, 0.004695, 0.002504, 0.025196, 0.010798, 0.054773, 0.051017,
0.008764, 0.001252, 0.010329, 0.086385, 0.041628, 0.006103, 0.001721, 0.021127, 0.000782, 0.014867, 0.000782},
  {0.125686, 0.020106, 0.015708, 0.064957, 0.165487, 0.016215, 0.004591, 0.006718, 0.125976, 0.000870, 0.005341, 0.136029, 0.013557, 0.004519, 0.076799,
0.012566, 0.000580, 0.008337, 0.038448, 0.034146, 0.023296, 0.007395, 0.009086, 0.000121, 0.083202, 0.000266},
  {0.194144, 0.044677, 0.009590, 0.004951, 0.240458, 0.005185, 0.001481, 0.005692, 0.111536, 0.001248, 0.000780, 0.003509, 0.041090, 0.003898, 0.118514,
0.062883, 0.000273, 0.028108, 0.032240, 0.023742, 0.043078, 0.000819, 0.007329, 0.000195, 0.014346, 0.000234},
  {0.084680, 0.018933, 0.056928, 0.145653, 0.088198, 0.014374, 0.110978, 0.012138, 0.058053, 0.003205, 0.007365, 0.011853, 0.011739, 0.018192, 0.064307,
0.008989, 0.001168, 0.006397, 0.068353, 0.161239, 0.011710, 0.006254, 0.014403, 0.000256, 0.014018, 0.000613},
  {0.021885, 0.020351, 0.022876, 0.024234, 0.007956, 0.112494, 0.012395, 0.009055, 0.013305, 0.001493, 0.008621, 0.044694, 0.064203, 0.180811, 0.029108,
0.028592, 0.000516, 0.137122, 0.040811, 0.058976, 0.091492, 0.022184, 0.039657, 0.001168, 0.005471, 0.000529},
  {0.135726, 0.005057, 0.003204, 0.003354, 0.172725, 0.004205, 0.001101, 0.031491, 0.058476, 0.000651, 0.000751, 0.097627, 0.009212, 0.000751, 0.133423,
0.052118, 0.000250, 0.178282, 0.022780, 0.040553, 0.039902, 0.000300, 0.003955, 0.000250, 0.003605, 0.000250},
  {0.004570, 0.004570, 0.005484, 0.004570, 0.005484, 0.004570, 0.004570, 0.005484, 0.004570, 0.004570, 0.004570, 0.004570, 0.004570, 0.004570, 0.004570,
0.005484, 0.004570, 0.004570, 0.004570, 0.005484, 0.879342, 0.005484, 0.004570, 0.005484, 0.004570, 0.004570},
  {0.112224, 0.016000, 0.027313, 0.035571, 0.225225, 0.011184, 0.015337, 0.011006, 0.098649, 0.001648, 0.015159, 0.018973, 0.028670, 0.025438, 0.099457,
0.013123, 0.000630, 0.018052, 0.077736, 0.075425, 0.020202, 0.009648, 0.011491, 0.000178, 0.031272, 0.000388},
  {0.117529, 0.034245, 0.044853, 0.019025, 0.106997, 0.017001, 0.006255, 0.051444, 0.092507, 0.002146, 0.007366, 0.016057, 0.020882, 0.013424, 0.078200,
```

11C Output

d) Used the same code as 11B.
N = 26
M = 26
A matrix calculated from 11C
No re-estimation for A rest all is same
T = 1000 (characters from ciphertext file)
Iterations = 200

Code – Question11D.cpp
Command to run – Question11D.exe ciphertext.txt
Output – 11DOutput.txt
After training the model and looking at the B matrix,

This is the mapping of letters
a->s, b->t, c->u, d->v, e->w, f->x, g->v, h->z, i->a, j->x, k->l, l->j, m->e, n->f, o->g, p->h, q->b, r->j, s->k, t->l, u->m, v->n, w->o, x->k, y->q, z->r

Putative key = 18
Correct positions: 20/26
Fraction of putative key elements that match actual key: 0.7692

5. Watch this video by Rick Beato: So It Begins...Is This A Real Band Or AI?

a. How did Mr. Beato determine that the "new artist" referred to in the title of the video was AI-generated?
Ans. Mr. Beato determined that “The Velvet Sundown” band is AI generated by testing their song using Logic Pro. It is Apple’s AI software offering a collection of tools related to music. Beato used Logic Pro to break down a couple of real songs into multiple separate tracks like vocals, guitar, keyboards, and other kinds. It worked pretty smoothly. But when he tried it with the band’s “Dust on Wind” song, the software struggled badly. This was expected because it is an AI generated track. Beato also said that AI generated songs are full of weird artifacts and garbled sounds because the music models which generate such songs are trained no low-quality MP3s, so the end result just doesn’t separate like a real performance would.

What could be done to make it more difficult to detect AI generated music?

Ans. To make AI-generated music harder to detect, the first step would be improving the audio quality. Mr. Beato pointed out that AI companies like suno, etc. need high quality music like multi-tracks, and also the effects used in the mix if possible. This will improve the music model output and would split into tracks nicely. Another way is to make the song and lyrics more human-like. No human is perfect, but these AI generated songs seem to be perfect. So, we can add some irregularities like timing variations, pitch changes, etc. Opposite to this, a real artist can write some parts, and then AI will fill in with supporting material.