NLP HW4

Xinzhuo Cheng, Dongkyu Kim, Shiyi Zhang

1. Complete Data, Incomplete Model

$$P(T \mid T) = 1/3$$

$$P(T\,O \mid T) = 2/3$$

2. Complete Model, Incomplete Data

B O A T

$$\begin{tikzcd}
B \arrow[d] & OW \arrow[d] \arrow[dr] & & T \arrow[d] \arrow[dr] & \\
B & O & O & T & O
\end{tikzcd}$$

$$\underbrace{1 \quad 0.8 \quad 0.5}_{0.4}$$

$$\frac{40}{42}$$

$$\frac{20}{21}$$

B O A T

$$\begin{tikzcd}
B \arrow[d] & OW \arrow[d] & & T \arrow[dl] \arrow[d] \arrow[dr] & \\
B & O & O & T & O
\end{tikzcd}$$

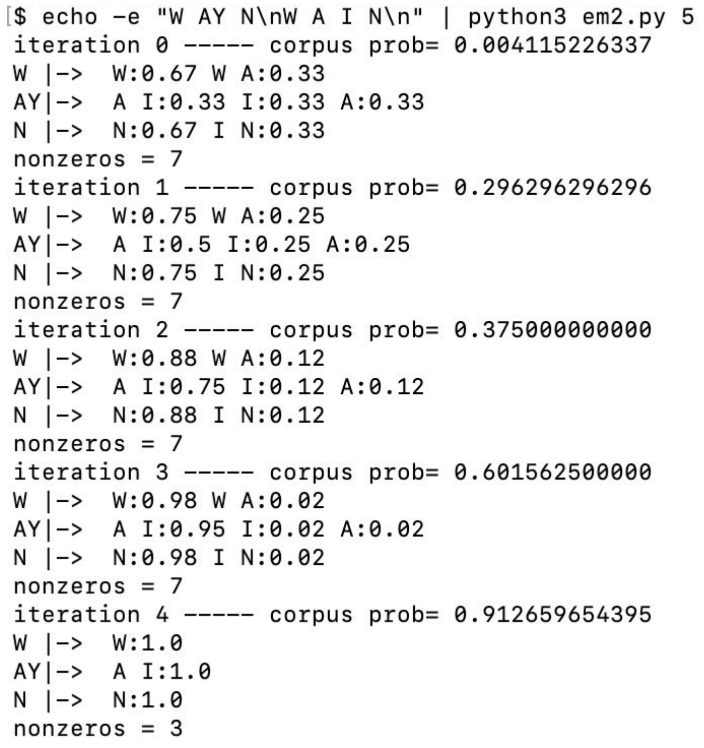
$$\underbrace{1 \quad 0.2 \quad 0.1}_{0.02}$$

$$\frac{2}{42}$$

$$\frac{1}{21}$$

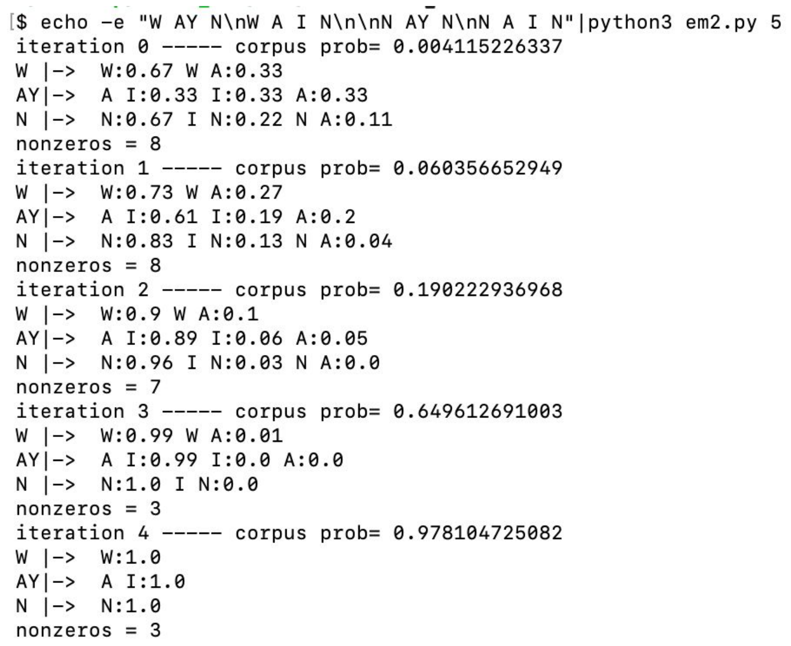
3. EM, Implementation Ⅰ: Enumerating All Alignments

1.



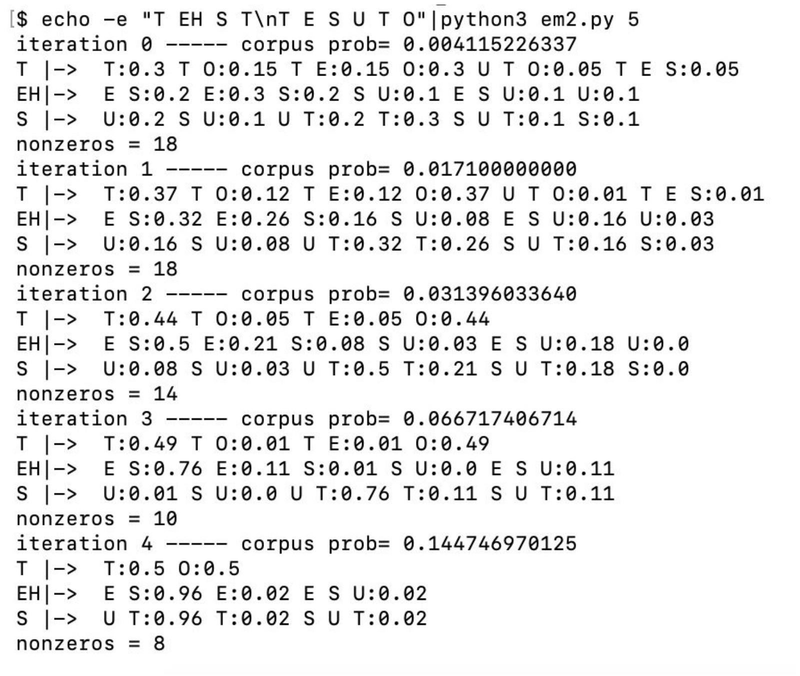
```
[$ echo -e "W AY N\nW A I N\n" | python3 em2.py 5
 iteration 0 ----- corpus prob= 0.004115226337
 W |->  W:0.67 W A:0.33
 AY|->  A I:0.33 I:0.33 A:0.33
 N |->  N:0.67 I N:0.33
 nonzeros = 7
 iteration 1 ----- corpus prob= 0.296296296296
 W |->  W:0.75 W A:0.25
 AY|->  A I:0.5 I:0.25 A:0.25
 N |->  N:0.75 I N:0.25
 nonzeros = 7
 iteration 2 ----- corpus prob= 0.375000000000
 W |->  W:0.88 W A:0.12
 AY|->  A I:0.75 I:0.12 A:0.12
 N |->  N:0.88 I N:0.12
 nonzeros = 7
 iteration 3 ----- corpus prob= 0.601562500000
 W |->  W:0.98 W A:0.02
 AY|->  A I:0.95 I:0.02 A:0.02
 N |->  N:0.98 I N:0.02
 nonzeros = 7
 iteration 4 ----- corpus prob= 0.912659654395
 W |->  W:1.0
 AY|->  A I:1.0
 N |->  N:1.0
 nonzeros = 3
```

2.



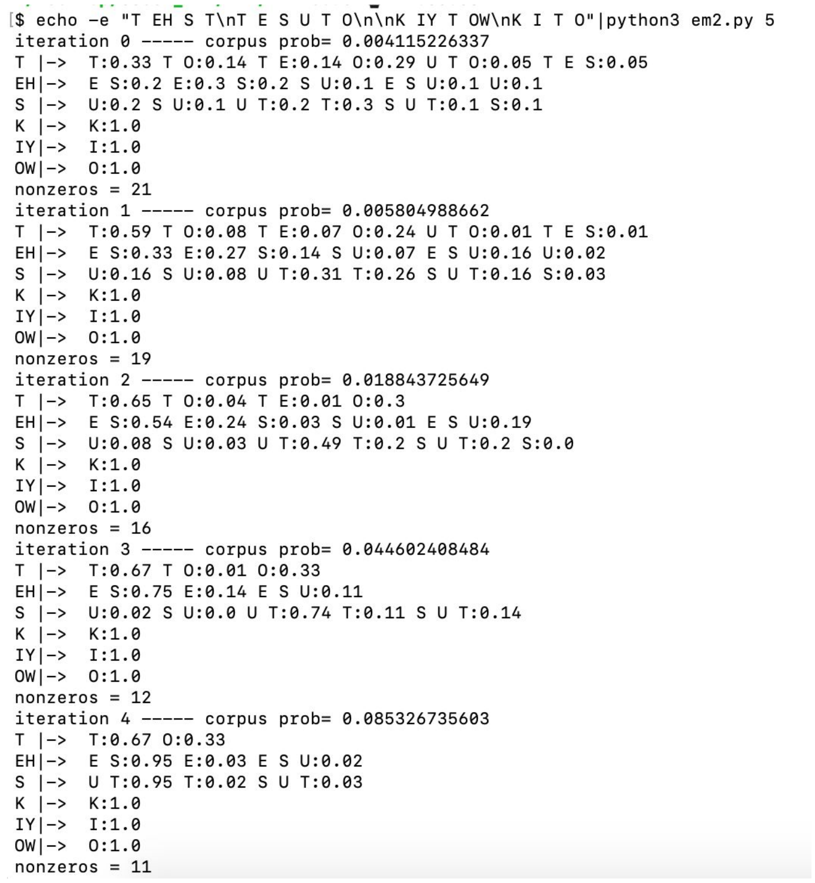
```
[$ echo -e "W AY N\nW A I N\n\nN AY N\nN A I N"|python3 em2.py 5
 iteration 0 ----- corpus prob= 0.004115226337
 W |->  W:0.67 W A:0.33
 AY|->  A I:0.33 I:0.33 A:0.33
 N |->  N:0.67 I N:0.22 N A:0.11
 nonzeros = 8
 iteration 1 ----- corpus prob= 0.060356652949
 W |->  W:0.73 W A:0.27
 AY|->  A I:0.61 I:0.19 A:0.2
 N |->  N:0.83 I N:0.13 N A:0.04
 nonzeros = 8
 iteration 2 ----- corpus prob= 0.190222936968
 W |->  W:0.9 W A:0.1
 AY|->  A I:0.89 I:0.06 A:0.05
 N |->  N:0.96 I N:0.03 N A:0.0
 nonzeros = 7
 iteration 3 ----- corpus prob= 0.649612691003
 W |->  W:0.99 W A:0.01
 AY|->  A I:0.99 I:0.0 A:0.0
 N |->  N:1.0 I N:0.0
 nonzeros = 3
 iteration 4 ----- corpus prob= 0.978104725082
 W |->  W:1.0
 AY|->  A I:1.0
 N |->  N:1.0
 nonzeros = 3
```

3.For epron “T”, there is a wrong learning that impossible align “T” to “O” which has same probability with “T” to “T”.



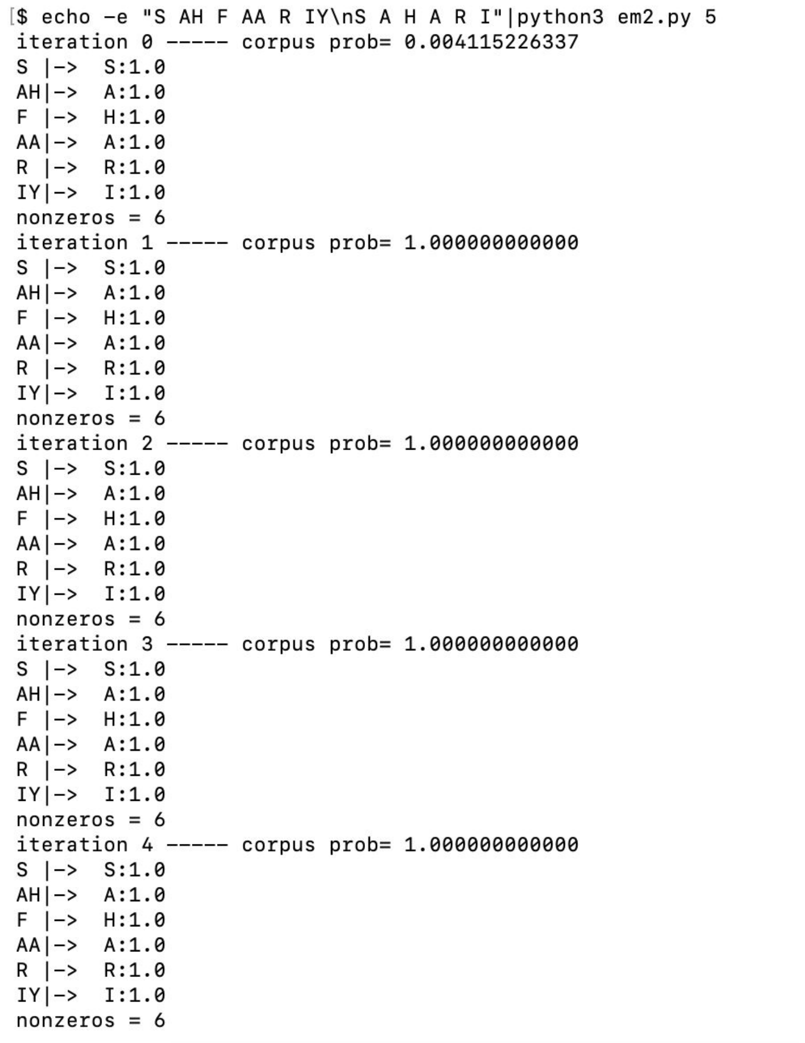
```
[$ echo -e "T EH S T\nT E S U T O"|python3 em2.py 5
 iteration 0 ----- corpus prob= 0.004115226337
 T |->  T:0.3 T O:0.15 T E:0.15 O:0.3 U T O:0.05 T E S:0.05
 EH|->  E S:0.2 E:0.3 S:0.2 S U:0.1 E S U:0.1 U:0.1
 S |->  U:0.2 S U:0.1 U T:0.2 T:0.3 S U T:0.1 S:0.1
 nonzeros = 18
 iteration 1 ----- corpus prob= 0.017100000000
 T |->  T:0.37 T O:0.12 T E:0.12 O:0.37 U T O:0.01 T E S:0.01
 EH|->  E S:0.32 E:0.26 S:0.16 S U:0.08 E S U:0.16 U:0.03
 S |->  U:0.16 S U:0.08 U T:0.32 T:0.26 S U T:0.16 S:0.03
 nonzeros = 18
 iteration 2 ----- corpus prob= 0.031396033640
 T |->  T:0.44 T O:0.05 T E:0.05 O:0.44
 EH|->  E S:0.5 E:0.21 S:0.08 S U:0.03 E S U:0.18 U:0.0
 S |->  U:0.08 S U:0.03 U T:0.5 T:0.21 S U T:0.18 S:0.0
 nonzeros = 14
 iteration 3 ----- corpus prob= 0.066717406714
 T |->  T:0.49 T O:0.01 T E:0.01 O:0.49
 EH|->  E S:0.76 E:0.11 S:0.01 S U:0.0 E S U:0.11
 S |->  U:0.01 S U:0.0 U T:0.76 T:0.11 S U T:0.11
 nonzeros = 10
 iteration 4 ----- corpus prob= 0.144746970125
 T |->  T:0.5 O:0.5
 EH|->  E S:0.96 E:0.02 E S U:0.02
 S |->  U T:0.96 T:0.02 S U T:0.02
 nonzeros = 8
```

After adding an english word ‘kettle’: K IY T OW / K I T O, it looks better than before



```
[$ echo -e "T EH S T\nT E S U T O\n\nK IY T OW\nK I T O"|python3 em2.py 5
 iteration 0 ----- corpus prob= 0.004115226337
 T |->  T:0.33 T O:0.14 T E:0.14 O:0.29 U T O:0.05 T E S:0.05
 EH|->  E S:0.2 E:0.3 S:0.2 S U:0.1 E S U:0.1 U:0.1
 S |->  U:0.2 S U:0.1 U T:0.2 T:0.3 S U T:0.1 S:0.1
 K |->  K:1.0
 IY|->  I:1.0
 OW|->  O:1.0
 nonzeros = 21
 iteration 1 ----- corpus prob= 0.005804988662
 T |->  T:0.59 T O:0.08 T E:0.07 O:0.24 U T O:0.01 T E S:0.01
 EH|->  E S:0.33 E:0.27 S:0.14 S U:0.07 E S U:0.16 U:0.02
 S |->  U:0.16 S U:0.08 U T:0.31 T:0.26 S U T:0.16 S:0.03
 K |->  K:1.0
 IY|->  I:1.0
 OW|->  O:1.0
 nonzeros = 19
 iteration 2 ----- corpus prob= 0.018843725649
 T |->  T:0.65 T O:0.04 T E:0.01 O:0.3
 EH|->  E S:0.54 E:0.24 S:0.03 S U:0.01 E S U:0.19
 S |->  U:0.08 S U:0.03 U T:0.49 T:0.2 S U T:0.2 S:0.0
 K |->  K:1.0
 IY|->  I:1.0
 OW|->  O:1.0
 nonzeros = 16
 iteration 3 ----- corpus prob= 0.044602408484
 T |->  T:0.67 T O:0.01 O:0.33
 EH|->  E S:0.75 E:0.14 E S U:0.11
 S |->  U:0.02 S U:0.0 U T:0.74 T:0.11 S U T:0.14
 K |->  K:1.0
 IY|->  I:1.0
 OW|->  O:1.0
 nonzeros = 12
 iteration 4 ----- corpus prob= 0.085326735603
 T |->  T:0.67 O:0.33
 EH|->  E S:0.95 E:0.03 E S U:0.02
 S |->  U T:0.95 T:0.02 S U T:0.03
 K |->  K:1.0
 IY|->  I:1.0
 OW|->  O:1.0
 nonzeros = 11
```

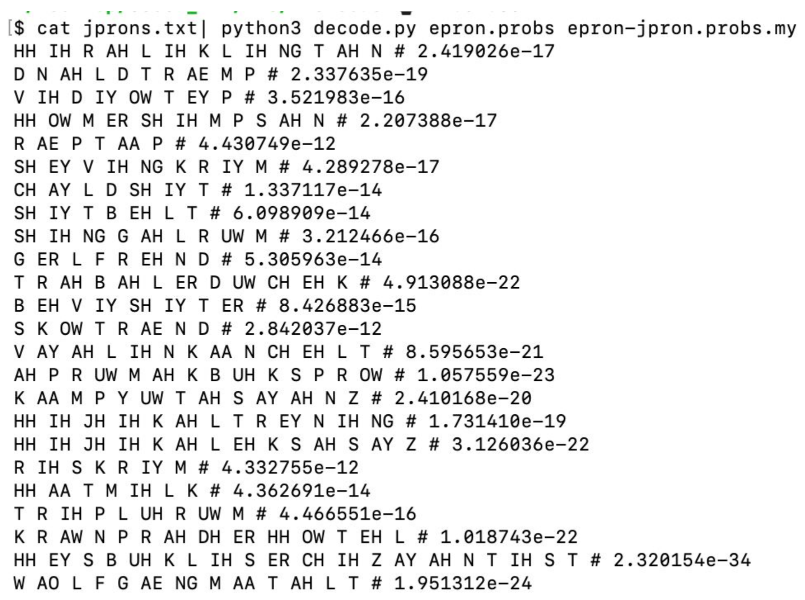
4.safari:S AH F AA R IY / S A H A R I



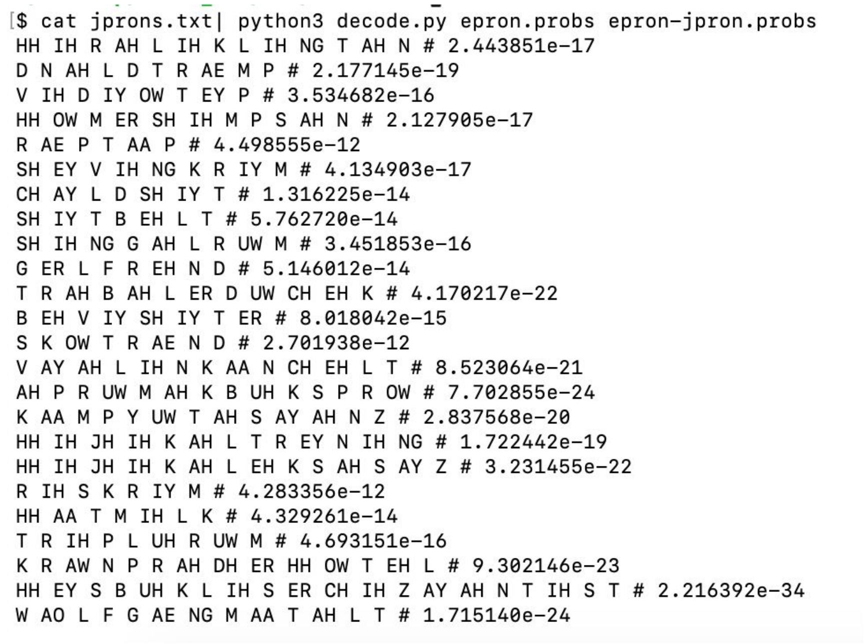
```
[$ echo -e "S AH F AA R IY\nS A H A R I"|python3 em2.py 5
 iteration 0 ----- corpus prob= 0.004115226337
 S |->  S:1.0
 AH|->  A:1.0
 F |->  H:1.0
 AA|->  A:1.0
 R |->  R:1.0
 IY|->  I:1.0
 nonzeros = 6
 iteration 1 ----- corpus prob= 1.000000000000
 S |->  S:1.0
 AH|->  A:1.0
 F |->  H:1.0
 AA|->  A:1.0
 R |->  R:1.0
 IY|->  I:1.0
 nonzeros = 6
 iteration 2 ----- corpus prob= 1.000000000000
 S |->  S:1.0
 AH|->  A:1.0
 F |->  H:1.0
 AA|->  A:1.0
 R |->  R:1.0
 IY|->  I:1.0
 nonzeros = 6
 iteration 3 ----- corpus prob= 1.000000000000
 S |->  S:1.0
 AH|->  A:1.0
 F |->  H:1.0
 AA|->  A:1.0
 R |->  R:1.0
 IY|->  I:1.0
 nonzeros = 6
 iteration 4 ----- corpus prob= 1.000000000000
 S |->  S:1.0
 AH|->  A:1.0
 F |->  H:1.0
 AA|->  A:1.0
 R |->  R:1.0
 IY|->  I:1.0
 nonzeros = 6
```

5.please see file epron-jpron.probs and epron-jpron.logs

6.Two almost have same output of English pronunciation. But the probability of them are slightly different.



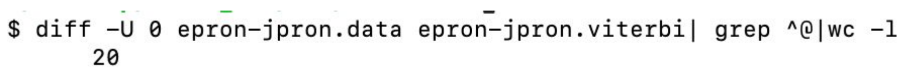
```
[$ cat jprons.txt| python3 decode.py epron.probs epron-jpron.probs.my
 HH IH R AH L IH K L IH NG T AH N # 2.419026e-17
 D N AH L D T R AE M P # 2.337635e-19
 V IH D IY OW T EY P # 3.521983e-16
 HH OW M ER SH IH M P S AH N # 2.207388e-17
 R AE P T AA P # 4.430749e-12
 SH EY V IH NG K R IY M # 4.289278e-17
 CH AY L D SH IY T # 1.337117e-14
 SH IY T B EH L T # 6.098909e-14
 SH IH NG G AH L R UW M # 3.212466e-16
 G ER L F R EH N D # 5.305963e-14
 T R AH B AH L ER D UW CH EH K # 4.913088e-22
 B EH V IY SH IY T ER # 8.426883e-15
 S K OW T R AE N D # 2.842037e-12
 V AY AH L IH N K AA N CH EH L T # 8.595653e-21
 AH P R UW M AH K B UH K S P R OW # 1.057559e-23
 K AA M P Y UW T AH S AY AH N Z # 2.410168e-20
 HH IH JH IH K AH L T R EY N IH NG # 1.731410e-19
 HH IH JH IH K AH L EH K S AH S AY Z # 3.126036e-22
 R IH S K R IY M # 4.332755e-12
 HH AA T M IH L K # 4.362691e-14
 T R IH P L UH R UW M # 4.466551e-16
 K R AW N P R AH DH ER HH OW T EH L # 1.018743e-22
 HH EY S B UH K L IH S ER CH IH Z AY AH N T IH S T # 2.320154e-34
 W AO L F G AE NG M AA T AH L T # 1.951312e-24
```



```
[$ cat jprons.txt| python3 decode.py epron.probs epron-jpron.probs
 HH IH R AH L IH K L IH NG T AH N # 2.443851e-17
 D N AH L D T R AE M P # 2.177145e-19
 V IH D IY OW T EY P # 3.534682e-16
 HH OW M ER SH IH M P S AH N # 2.127905e-17
 R AE P T AA P # 4.498555e-12
 SH EY V IH NG K R IY M # 4.134903e-17
 CH AY L D SH IY T # 1.316225e-14
 SH IY T B EH L T # 5.762720e-14
 SH IH NG G AH L R UW M # 3.451853e-16
 G ER L F R EH N D # 5.146012e-14
 T R AH B AH L ER D UW CH EH K # 4.170217e-22
 B EH V IY SH IY T ER # 8.018042e-15
 S K OW T R AE N D # 2.701938e-12
 V AY AH L IH N K AA N CH EH L T # 8.523064e-21
 AH P R UW M AH K B UH K S P R OW # 7.702855e-24
 K AA M P Y UW T AH S AY AH N Z # 2.837568e-20
 HH IH JH IH K AH L T R EY N IH NG # 1.722442e-19
 HH IH JH IH K AH L EH K S AH S AY Z # 3.231455e-22
 R IH S K R IY M # 4.283356e-12
 HH AA T M IH L K # 4.329261e-14
 T R IH P L UH R UW M # 4.693151e-16
 K R AW N P R AH DH ER HH OW T EH L # 9.302146e-23
 HH EY S B UH K L IH S ER CH IH Z AY AH N T IH S T # 2.216392e-34
 W AO L F G AE NG M AA T AH L T # 1.715140e-24
```

7.See epron-jpron.viterbi file

8.



```
$ diff -U 0 epron-jpron.data epron-jpron.viterbi| grep ^@|wc -l
      20
```

4. EM, Implementation Ⅱ: Forward-Backward (DP)

1.Assume each epron maximally align 3 jpron.
Then, assume $d = m - n$, and $\exists\, i,j \in N,\ d = i*1 + j*2$ (i is the number of epron to 2 jprons, and j is the number of epron to 3 jprons). We have:

Number of alignments = $\sum_{i,j}(C_n^i * C_{n-i}^j)$

The highest number of alignments is 10, and pairs such as AH B AW T / A B A U T O, AE K T ER / A K U T A A, and AE F T ER / A H U T A A have 10 possible alignments.

2. **Pseudo code**

```
# forward
For each epron from the start:
    For each jpron:
        For each sequence of the pair:
            Calculate the score
            Sum score and store in forward table
Update totalprob

# backward
For each epron from the end:
    For each jpron:
        For each sequence of the pair:
            If each previous epron is in english-japanae dictionary:
                Probability of the pair is the probability of the previous epron and japanage pair.
            Else:
                Probability of the pair is 0.001
            Calculate score
            Sum score and store in backward table

# calculate fraccount following the formula in the slide.
```

Time complexity:
N is the length of English pronunciation set
M is the length of Japanese pronunciation set
T is the length of the paired pronunciation sequence.

Thus, the time complexity is O(N*M*T)

3. The forward-backward algorithm is an inference algorithm for hidden Markov models which computes the posterior marginals of all hidden state variables. This inference task is generally called smoothing. The algorithm makes use of the principle of dynamic programming to efficiently compute the values that are required to obtain the posterior marginal distributions in two passes.
In this experiment, by applying the forward-backward algorithm, we can avoid the enumeration of the Viterbi algorithm. With respect to the Viterbi algorithm, it finds the path with the max possibility. However, with respect to the forward-backward algorithm, the forward is just like Viterbi, and the backward is like a reversed Viterbi. We replaced finding the max possibility with summing up each possibility of transition as the idea of dynamic programming.
We believe this is the reason why the forward-backward algorithm based EM is faster than the slow version EM.

5. EM for Decipherment