**COMP 9318 Data Warehousing**

**and Data Mining**

**T1, 2019**

**Project 1**

Group: Dalao

Wanze LIU (z5137189), Zhou JIANG (z5146092)

School of Computer Science and Engineering

UNSW Sydney

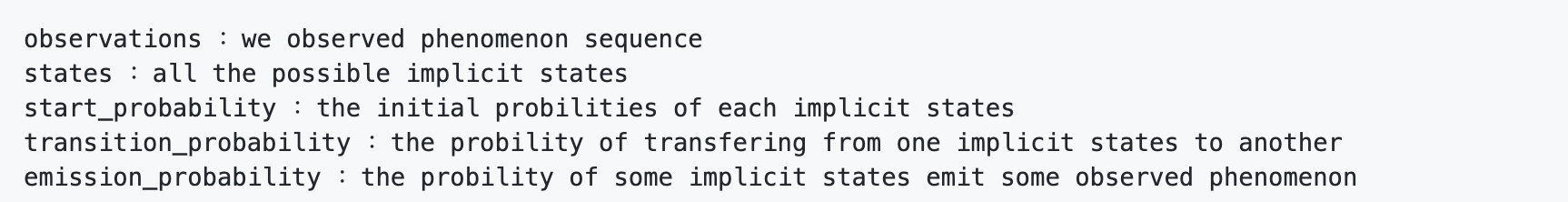
Introduction

This report briefly illustrates the steps we made to accomplish this project. We have three tasks in this project, the first is to find the implicit sequence with the highest probability , the second one is to find top-k path, and the last one is to improve task one

**Task 1**

### Viterbi Algorithm

for HMM, the most useful function is to find the most likely implicit sequence according to its observation, In general, the HMM problem can be described by the following five elements:



If you use the brute-force method to exhaust all possible state sequences and compare their probability values, the time complexity is O(n^m), obviously , this is unacceptable when we want to find a long sequnce with large dataset, however, we can decrease its time complexity by using Viterbi Algorithem,

we can consider this probelm as dynamic programming , the last\_state is the probability of each implicit state corresponding to the previous observed phenomenon, and curr\_pro is the probability of each implicit state corresponding to the current observed phenomenon. Solving cur\_pro actually depends only on last\_state, this is core thinking of Vitberi Algorithem.

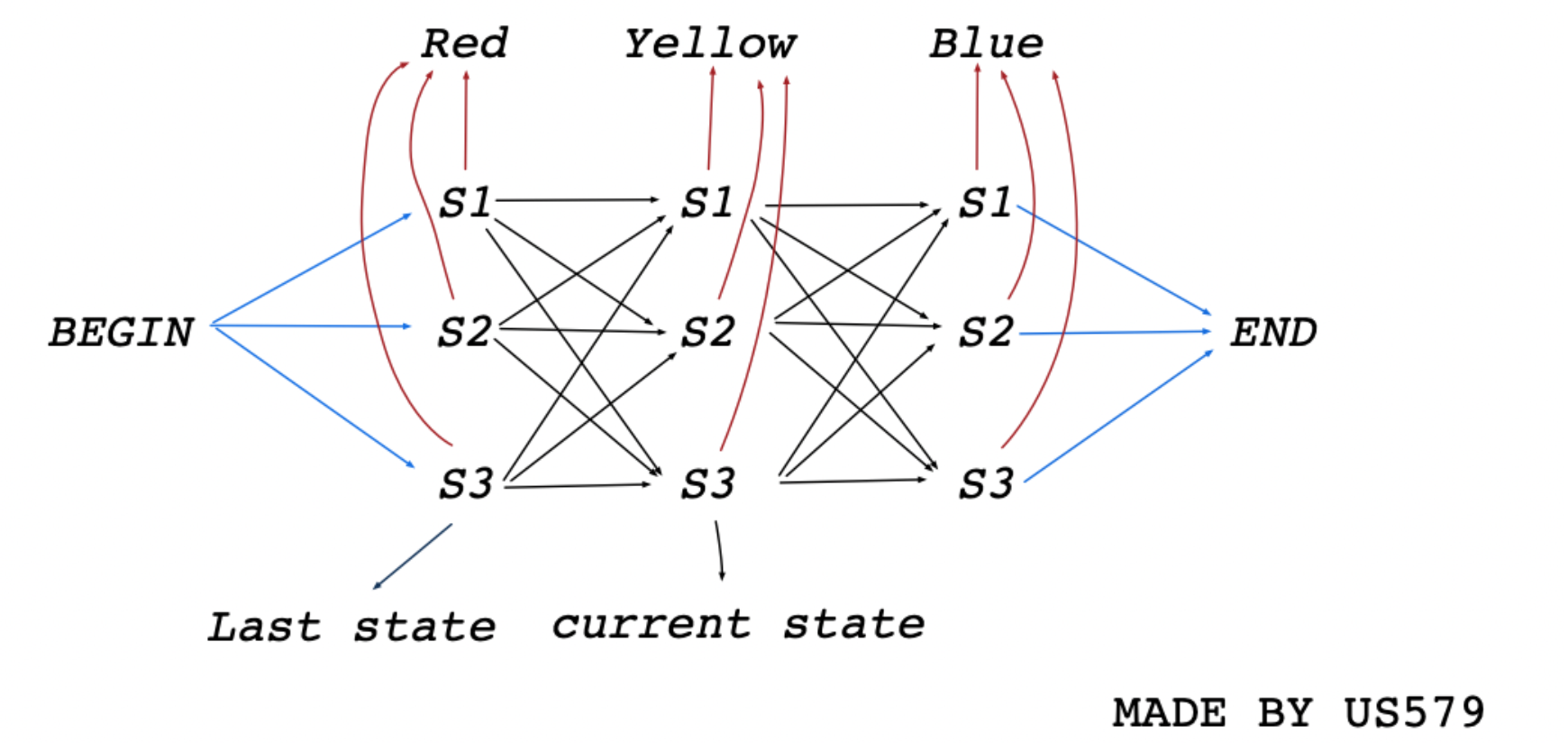
Here, I describe the second sample (toy) as a example to illustrates this algorithem, In this case below, the output of the implicit sequence is

[3, 0, 0, 1, 2, 4, -9.843403381747937]

with log probility -9.843403381747937

##### Parameter breakdown

0:S1 1:S2 2:S3 3:BEGIN 4:END



### 1.Initial Probilities

The blue line represent the initial probability (Pi) which can be deemed as equivalent to transition probabilities from the BEGIN state to all the implicit state

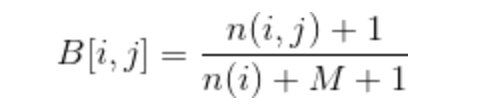
So, we caculate as

for s in states[:-2]:

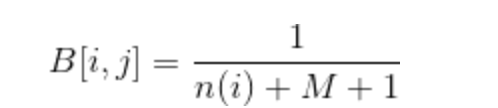
transition\_probability[len(states)-2][states.index(s)])

### 2.Emission Probilities

The red line represent its emission probability from state after smoothing is

****

If the symbol is an unknown symbol, its emission probability from state after smoothing is

****

for i in range(1,len(obs)):

for cur in range(len(states[:-2])):

#if there is no emission from states `cur` to observation `sym.index(obs[i])`(this is the index in the symbol list),we us add one smoothing (in case it is 0)

if str(cur)+'-'+str(sym.index(obs[i])) not in emission\_probability:

emission\_rate = 1.0 / (dic\_distance[cur] + n2 +1)

else:

#otherwise i will use the formula below

emission\_rate = emission\_probability[str(cur)+'-'+str(sym.index(obs[i]))]

### 2.Transition Probilities

for i in range(n1):

for j in range(n1):

transition\_probability[i][j] = (float(distance[i][j])+1) / (sum(distance[i])+n1-1)

the number of state\_i transfer to state\_j divide by the total number of transfering state\_j to any states , i also use add-1 smoothing here

### 3.Viterbi Algorithm

Considering this problem from the problem of dynamic programming, according to the definition of the first figure, last\_state is the probability of each implicit state corresponding to the previous observation phenomenon, and curr\_state is the probability of each implicit state corresponding to the current observation phenomenon. Solving curr\_state actually depends only on last\_state. And their dependencies can be expressed in the python code below

### 