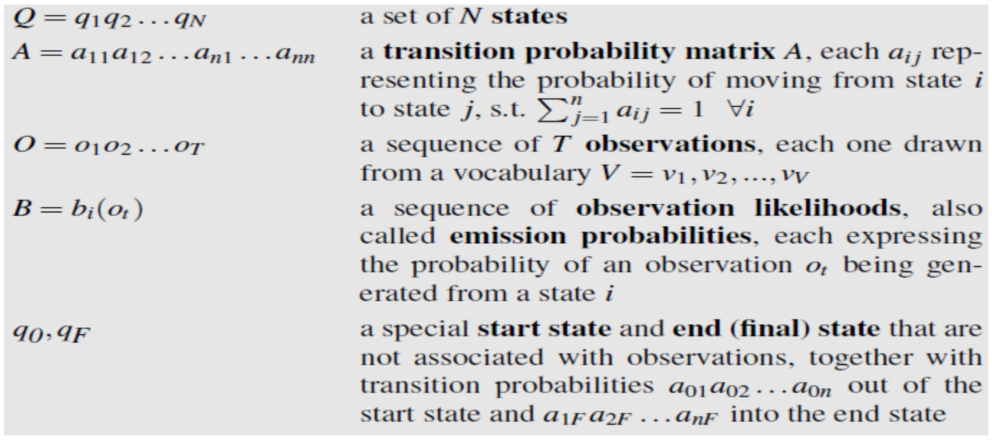
Viterbi Algorithm

The Hidden Markov Model

A process can be identified as a Markov process if one can make predictions for the future of the process based solely on its present states. This property is called as the Markov property. A Markov chain is a type of Markov process that has a discrete state space.

A Markov chain can be used to compute a probability for a sequence of events that we can observe in the world. But most of the time we can't observe those events directly to map with a Markov Chain. As an example we can't observe part of the speech tags in sentences directly. Thus we have to find out the correct tag sequence from the word sequence. The tags are hidden because they aren't directly observed. The HMM is a sequence model which allows us to map the observed events and hidden events. The HMM is a probabilistic sequence model, given a sequence of units (words, letters, morphemes, sentences, whatever), they compute a probability distribution over possible sequences of labels and choose the best label sequence.

An HMM is specified by the following components,



$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, \ldots, v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i
q_0, q_F	a special **start state** and **end (final) state** that are not associated with observations, together with transition probabilities $a_{01} a_{02} \ldots a_{0n}$ out of the start state and $a_{1F} a_{2F} \ldots a_{nF}$ into the end state

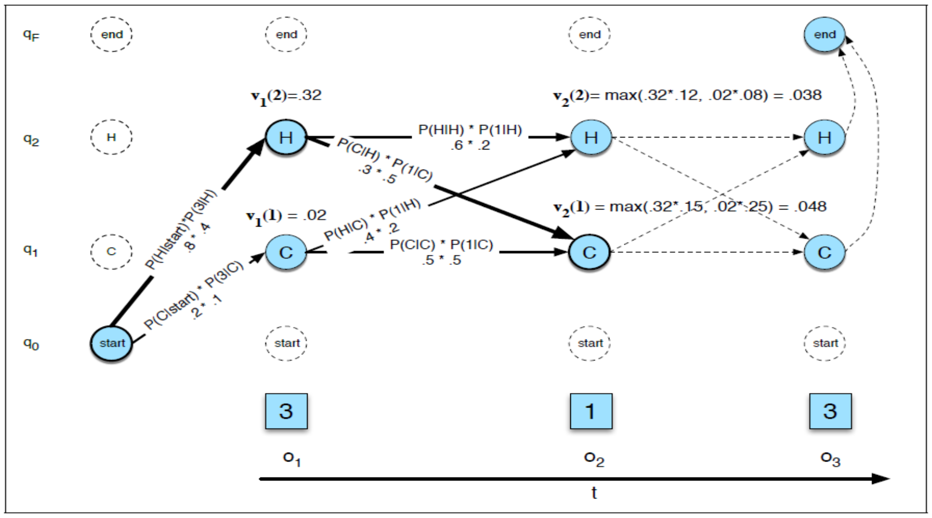
An HMM can be characterized into three parts,

- **Likelihood computation** - Given an HMM **λ = (A,B)** and an observation sequence O, determine the likelihood **P(O| λ)**.
- **Decoding** - Given an observation sequence **O** and an HMM **λ = (A,B)** discover the best hidden state sequence **Q**.
- **Learning** - Given an observation sequence **O** and the set of states in the HMM, learn the HMM parameters **A** and **B**.

The Viterbi Algorithm

In the decoding problem, brute force method is to compute the likelihood of the observation sequence for all possible hidden state sequences. Then we could choose the hidden state sequence with the maximum observation likelihood as the answer. For an HMM with **N** hidden states and an observation sequence of **T** observations, there are $\mathbf{N^T}$ possible hidden sequences. In a real world application, where **N** and **T** are both large, $\mathbf{N^T}$ is a very large number, so we cannot compute all the observation likelihoods for each hidden state sequence.

Instead of using such an extremely exponential algorithm, we use an efficient $\mathbf{O(N^2T)}$ algorithm called the Viterbi Algorithm. This is a is a kind of dynamic programming algorithm which uses intermediate probability values of the hidden state paths and fold these multiple paths into a graph called **Viterbi trellis**.



The above Viterbi trellis shows how to compute the best path through the hidden state space for the observation sequence $\mathbf{o_1, o_2, o_3}$ and state sequence $\mathbf{q_1, q_2}$ where $\mathbf{q_0, q_F}$ are start and end states. Hidden states are in circles, observations in squares. White (unfilled) circles indicate illegal transitions. The figure shows the computation of $\mathbf{v_t(j)}$ for two states at two time steps. The computation in each cell is done by choosing the most probable path that could lead to that cell. Like other dynamic programming algorithms, Viterbi fills each cell recursively. The resulting probability expressed in each cell is as follows.

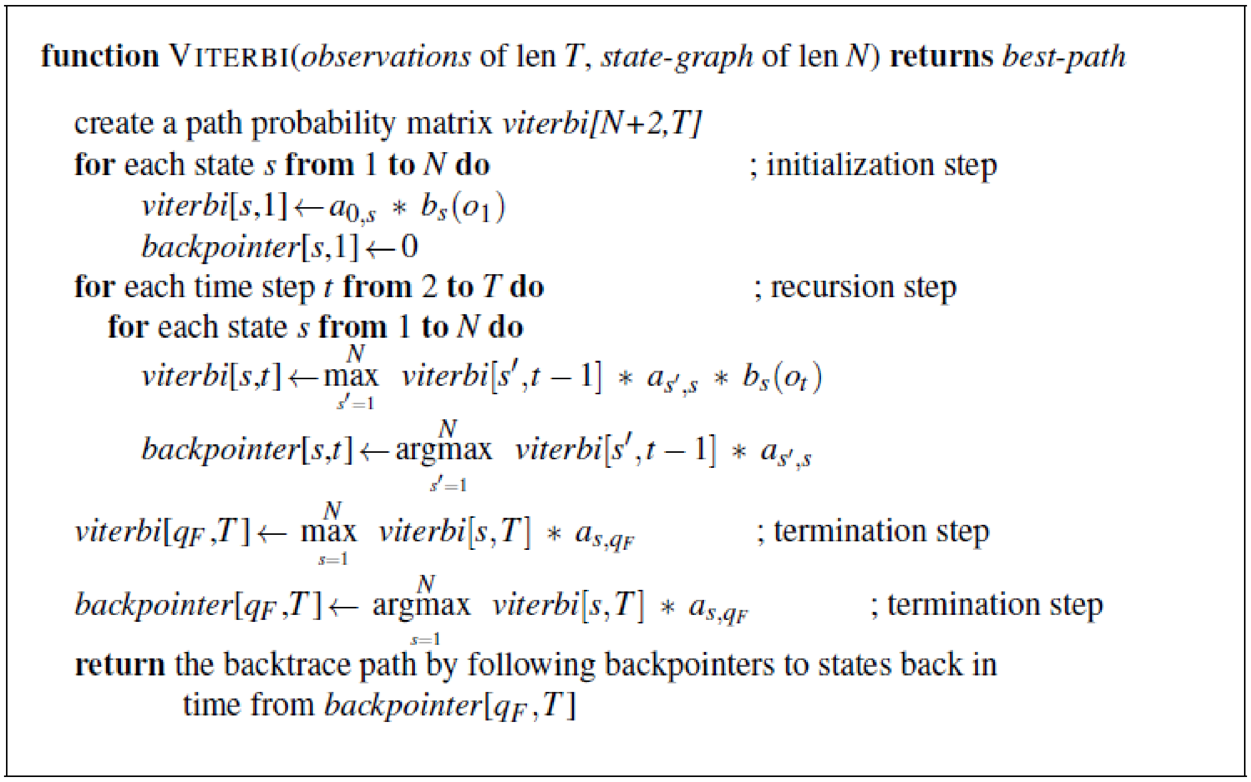
$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

Where,

$\mathbf{V_{t-1}(i)}$	- previous Viterbi path probability from the previous time step
$\mathbf{a_{i\,j}}$	- the transition probability from previous state $\mathbf{q_i}$ to current state $\mathbf{q_j}$
$\mathbf{b_j(o_t)}$	- the state observation likelihood of the observation symbol $\mathbf{o_t}$ given the current state **j**

The best state sequence is computed while keeping the track of the path of hidden states that led to each state. This is called the **Viterbi back-trace.**

Pseudo code of the Viterbi algorithm is as follows,



```
function VITERBI(observations of len T, state-graph of len N) returns best-path

  create a path probability matrix viterbi[N+2,T]
  for each state s from 1 to N do                              ; initialization step
      $viterbi[s,1] \leftarrow a_{0,s} * b_s(o_1)$
      $backpointer[s,1] \leftarrow 0$
  for each time step t from 2 to T do                          ; recursion step
    for each state s from 1 to N do
      $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
      $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s}$
  $viterbi[q_F,T] \leftarrow \max_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$          ; termination step
  $backpointer[q_F,T] \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$          ; termination step
  return the backtrace path by following backpointers to states back in
          time from $backpointer[q_F,T]$
```

References

- Speech and Language Processing. Daniel Jurafsky & James H. Martin
- 50.007 Machine Learning, Fall 2015, Singapore University of technology and design
- Video from Coursera - Columbia University - Course: Natural Language Processing: https://www.coursera.org/course/nlangp