Hidden Markov Models

Arin Ghazarian
Chapman University

HMM

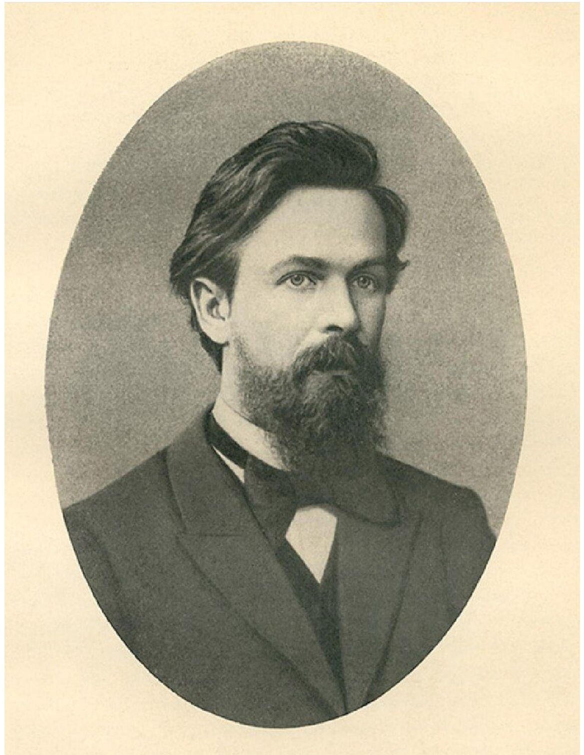
- Introduced in a series of statistical papers by Leonard E. Baum and other authors in the second half of the 1960s.
- One of the first applications of HMMs was speech recognition, starting in the mid-1970s.
- Later it found many applications in the analysis of biological sequences

HMM Applications

- HMMs can solve problems in many different fields where the goal is to recover a data sequence that itself is not immediately observable and instead the other data that depend on the sequence are
    - Computational finance
    - Neuroscience
    - Cryptanalysis
    - Speech recognition
    - Speech synthesis
    - Part-of-speech tagging
    - Machine translation
    - Handwriting recognition
    - Alignment of bio-sequences
    - Time series analysis
    - Activity recognition
    - Protein folding
    - Sequence classification
    - Transportation forecasting

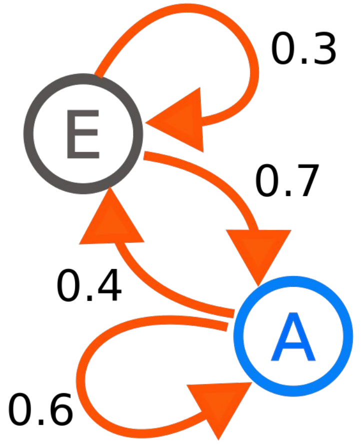
Markov Property

- In probability theory and statistics, the term Markov property refers to the memoryless property of a stochastic process
- Future evolution is independent of its history
- Named after the Russian mathematician Andrey Markov.


А. А. Марков (1886).

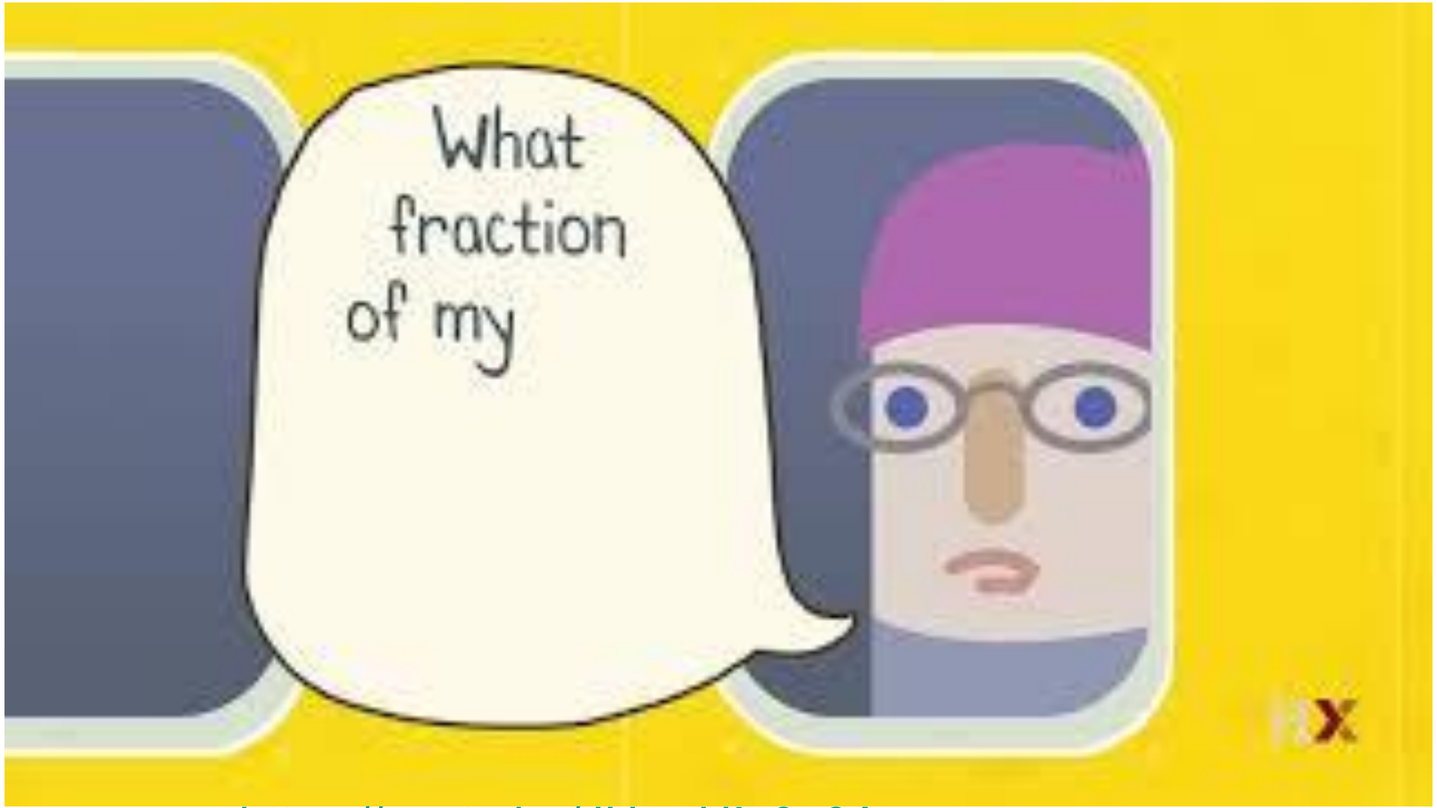
Markov Chain

- A Markov chain or Markov process is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event.
    - Discrete-time Markov chain (DTMC)
    - Continuous-time process is called a continuous-time Markov chain (CTMC)
- What happens next depends only on the state of affairs now

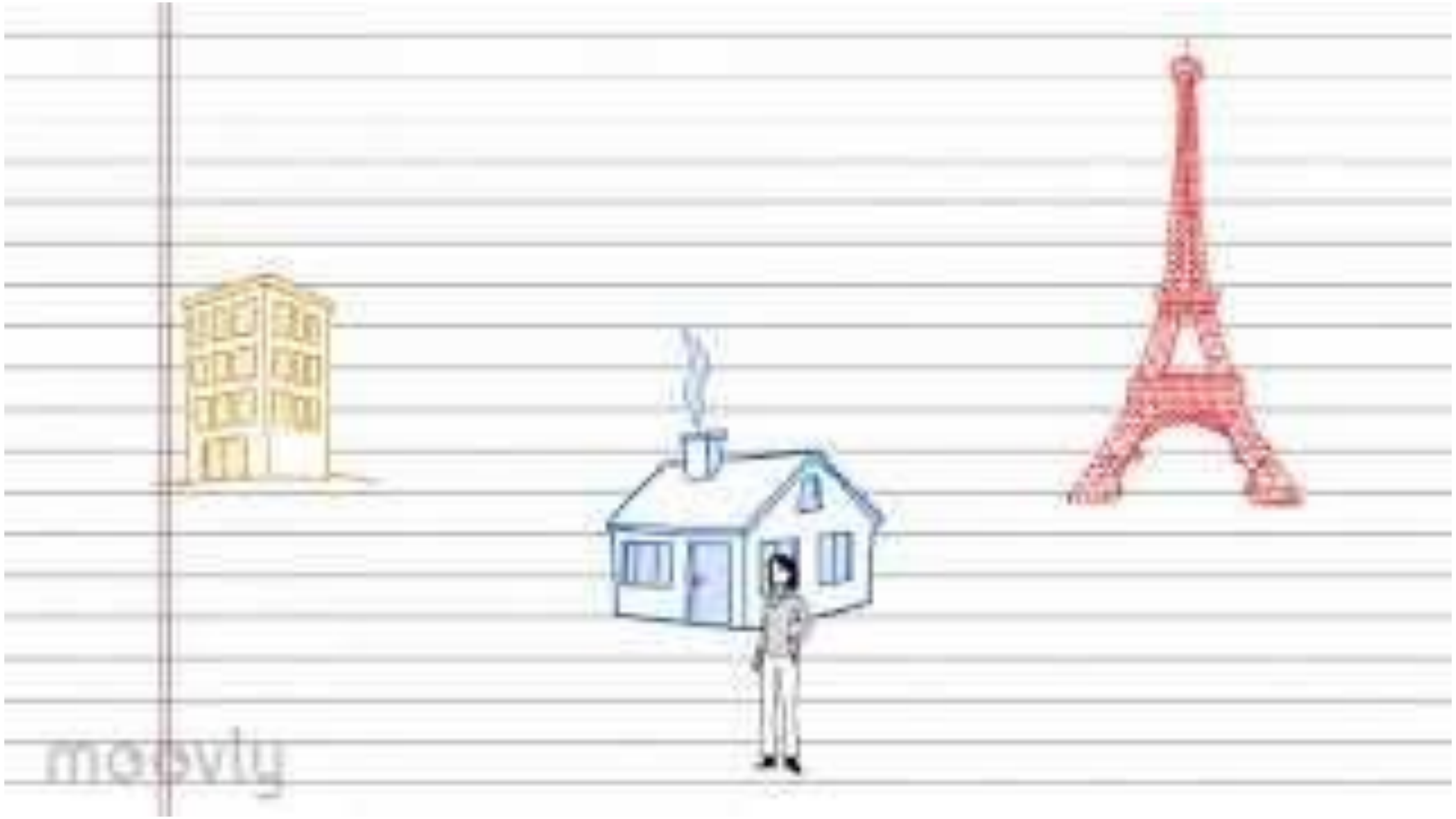

$$\begin{tikzcd} E \arrow[loop above, "0.3"] \arrow[r, bend left, "0.7"] & A \arrow[l, bend left, "0.4"] \arrow[loop below, "0.6"] \end{tikzcd}$$

https://en.wikipedia.org/wiki/Markov_chain

Markov Chain



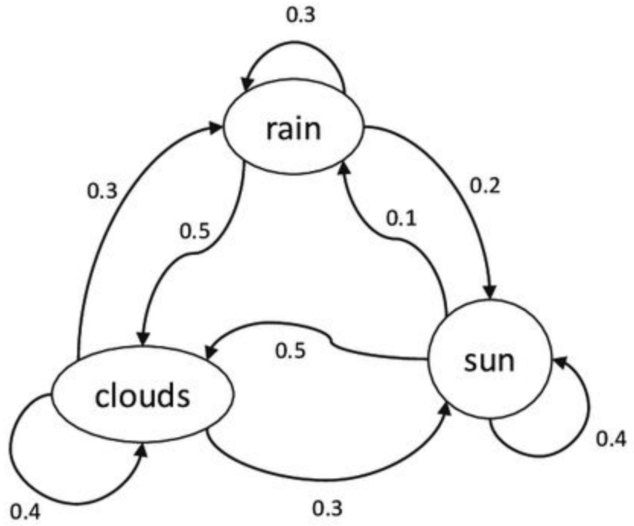
https://youtu.be/JHwyHIz6a8A



https://youtu.be/EqUfuT3CC8s

Transition Matrix

- A transition matrix (also known as stochastic matrix or Markov matrix) is a square matrix used to describe the transitions of a Markov chain

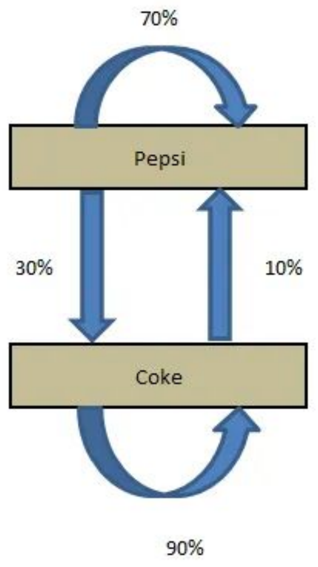


source

$$
\begin{array}{c|ccc}
 & \text{clouds} & \text{rain} & \text{sun} \\
\hline
\text{clouds} & 0.4 & 0.3 & 0.3 \\
\text{rain} & 0.5 & 0.3 & 0.2 \\
\text{sun} & 0.5 & 0.1 & 0.4
\end{array}
$$

Transition Matrix

- Similar to adjacency matrix in Graph theory



source

$$\begin{array}{cc} \text{Pepsi} & \text{Coke} \\ 55\% & 45\% \end{array} \quad \times \quad \begin{array}{ccc} & \overset{\text{Final state}}{\overset{\downarrow}{\text{Pepsi}}} & \text{Coke} \\ \text{Pepsi} & 70\% & 30\% \\ \underset{\underset{\text{Initial state}}{\uparrow}}{\text{Coke}} & 10\% & 90\% \end{array} \quad = \quad \begin{array}{cc} \text{Pepsi} & \text{Coke} \\ 43\% & 57\% \end{array}$$

Transition Matrix

- The probability of ending up in any given state after 1,2, and 3 steps:

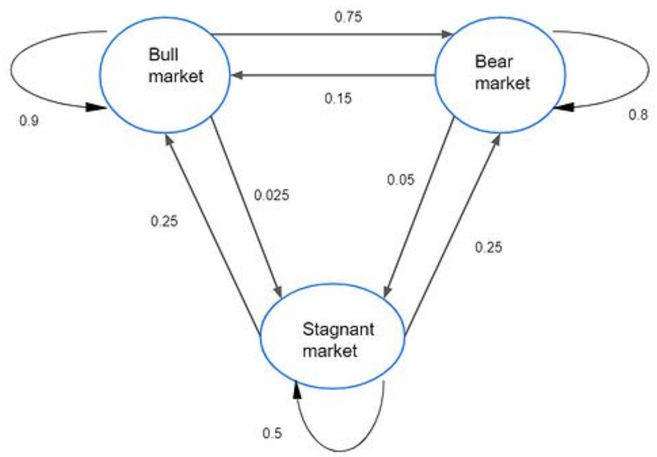
$$\begin{aligned}\pi_1 &= \pi_0 P\\ \pi_2 &= \pi_1 P = \pi_0 P^2\\ \pi_3 &= \pi_2 P = \pi_0 P^3,\end{aligned}$$

- The probability of ending up in any given state after n steps:

$$\pi_n = \pi_0 P^n.$$

Transition Matrix

- Stock markets trend state space:
  - **Bull markets:** prices are rising due to the actors having optimistic hopes of the future.
  - **Bear markets:** prices are declining due to the actors having a pessimistic view of the future.
  - **Stagnant markets:** neither a decline nor a rise

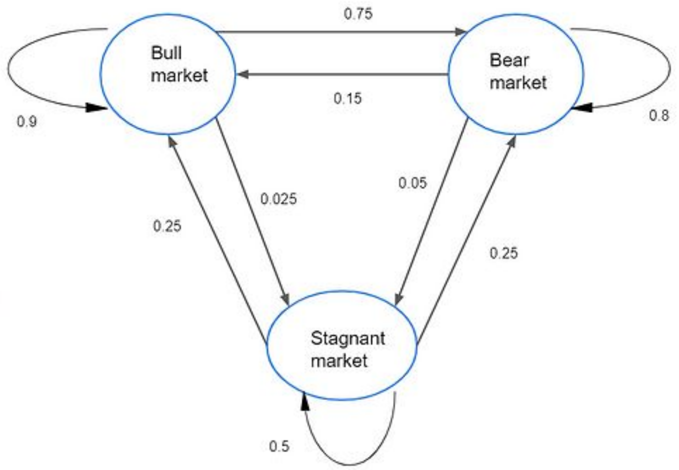


source

Transition Matrix

To / *From*	Bull	Bear	Stagnant
Bull	0.9	0.075	0.025
Bear	0.15	0.8	0.05
Stagnant	0.25	0.25	0.5

$$= \begin{bmatrix} 0.9 & 0.075 & 0.025 \\ 0.15 & 0.8 & 0.05 \\ 0.25 & 0.25 & 0.5 \end{bmatrix} = M$$



source

Stationary Distribution

- In a regular Markov chain, transition probabilities will converge to a steady state over a large enough number of iterations
- After a sufficient number of iterations, the probability of ending up in any given state is the same, regardless of where you start

$$\text{One week from now: } C * M^1 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0.9 & 0.075 & 0.025 \\ 0.15 & 0.8 & 0.05 \\ 0.25 & 0.25 & 0.5 \end{bmatrix}^1 = \begin{bmatrix} 0.15 & 0.8 & 0.05 \end{bmatrix}$$

$$\text{5 weeks from now: } C * M^5 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0.9 & 0.075 & 0.025 \\ 0.15 & 0.8 & 0.05 \\ 0.25 & 0.25 & 0.5 \end{bmatrix}^5 = \begin{bmatrix} 0.48 & 0.45 & 0.07 \end{bmatrix}$$

$$\text{52 weeks from now: } C * M^{52} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0.9 & 0.075 & 0.025 \\ 0.15 & 0.8 & 0.05 \\ 0.25 & 0.25 & 0.5 \end{bmatrix}^{52} = \begin{bmatrix} 0.63 & 0.31 & 0.05 \end{bmatrix}$$

$$\text{99 weeks from now: } C * M^{99} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0.9 & 0.075 & 0.025 \\ 0.15 & 0.8 & 0.05 \\ 0.25 & 0.25 & 0.5 \end{bmatrix}^{99} = \begin{bmatrix} 0.63 & 0.31 & 0.05 \end{bmatrix}$$

source

Stationary Distribution

- A stationary distribution π is a vector, whose entries are non-negative and sum to 1, is unchanged by the operation of transition matrix P: $\pi = \pi P$
- Long Term behavior of a markov chain
- Does not depend on the initial state

Stationary Distribution: Power Method

- If we keep taking powers of the transition matrix, it will eventually converge and will stay constant. The values in columns of the transition matrix i will be the same and represent the probability of ending up in the state:

$$\lim_{k\to\infty} \mathbf{P}^k$$

$$A^{\infty} = \begin{bmatrix} 0.4444 & 0.3333 & 0.2222 \\ 0.4444 & 0.3333 & 0.2222 \\ 0.4444 & 0.3333 & 0.2222 \end{bmatrix}$$

Stationary Distribution: EigenVector Method

- The stationary distribution is a left eigenvector of the transition matrix for eigenvalue=1

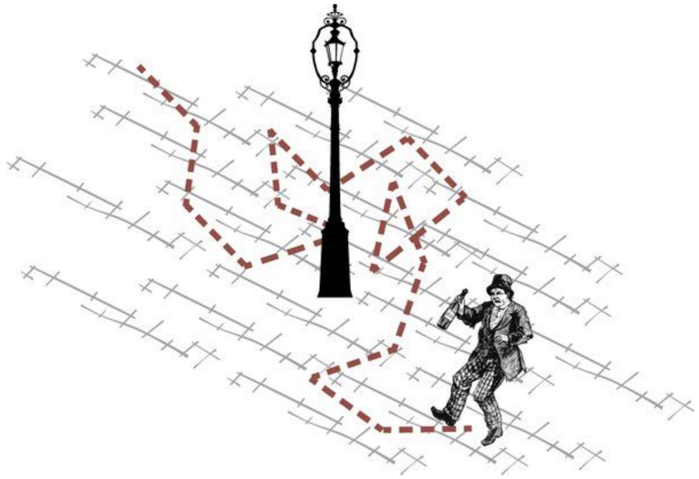
$$\pi P = \pi$$

$$\pi P = 1 \times \pi$$

$$\mathbf{X}_L \, \mathsf{A} = \lambda_L \, \mathbf{X}_L \, .$$

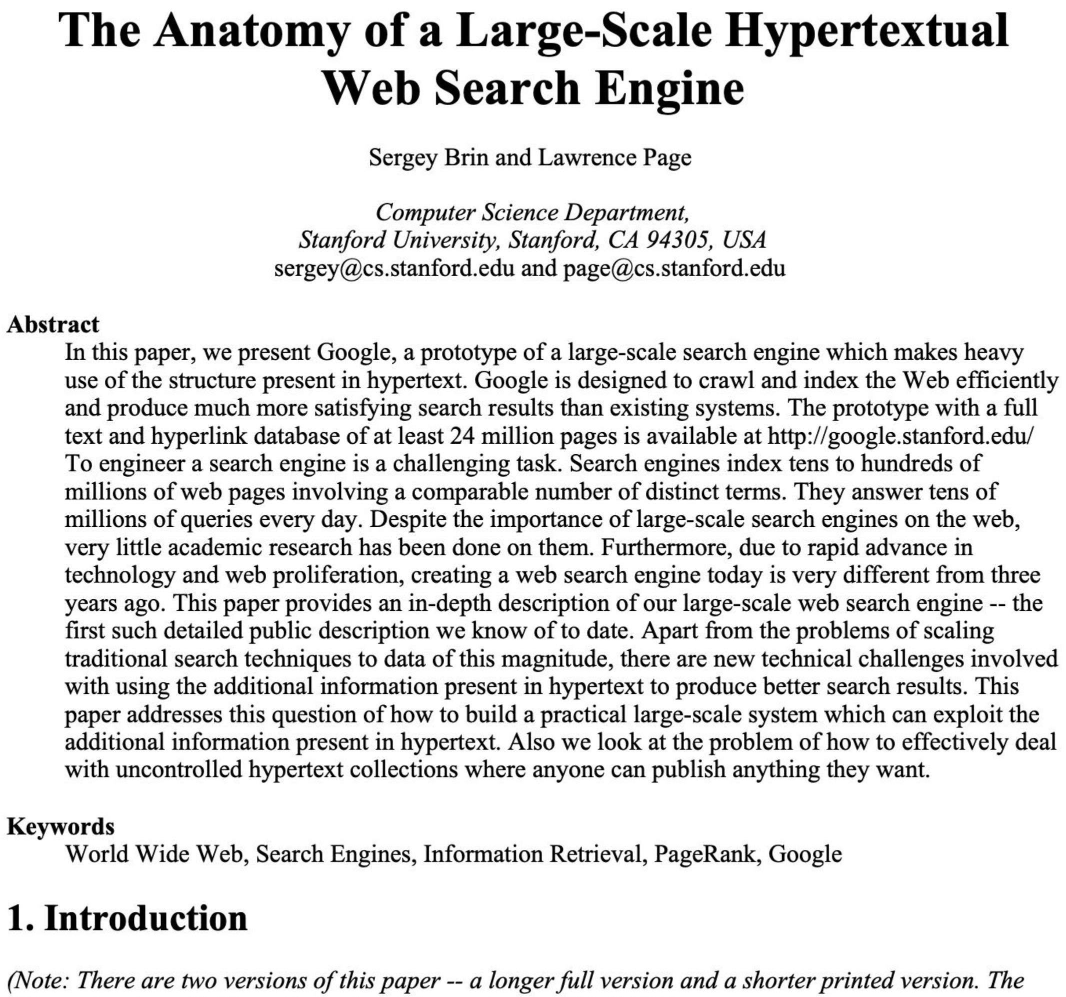
Random Walk

- A random walk (also known as a drunkard's walk), a process for determining the probable location of a point subject to random motions, given the probabilities of moving some distance in some direction
- A random process that describes a path that consists of a succession of random steps on some mathematical space
- Random walks are an example of Markov processes, in which future behaviour is independent of past history
- Examples
  - Path traced by a molecule as it travels in a liquid or a gas (Brownian motion)
  - The search path of a foraging animal
  - The price of a fluctuating stock
  - The financial status of a gambler
- Random walk: outgoing edge is chosen uniformly at random
- Markov chain: outgoing edge is chosen according to an arbitrary fixed distribution



PageRank

PageRank Paper



The Anatomy of a Large-Scale Hypertextual Web Search Engine

Sergey Brin and Lawrence Page

Computer Science Department,
Stanford University, Stanford, CA 94305, USA
sergey@cs.stanford.edu and page@cs.stanford.edu

Abstract

In this paper, we present Google, a prototype of a large-scale search engine which makes heavy use of the structure present in hypertext. Google is designed to crawl and index the Web efficiently and produce much more satisfying search results than existing systems. The prototype with a full text and hyperlink database of at least 24 million pages is available at http://google.stanford.edu/ To engineer a search engine is a challenging task. Search engines index tens to hundreds of millions of web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. Despite the importance of large-scale search engines on the web, very little academic research has been done on them. Furthermore, due to rapid advance in technology and web proliferation, creating a web search engine today is very different from three years ago. This paper provides an in-depth description of our large-scale web search engine -- the first such detailed public description we know of to date. Apart from the problems of scaling traditional search techniques to data of this magnitude, there are new technical challenges involved with using the additional information present in hypertext to produce better search results. This paper addresses this question of how to build a practical large-scale system which can exploit the additional information present in hypertext. Also we look at the problem of how to effectively deal with uncontrolled hypertext collections where anyone can publish anything they want.

Keywords

World Wide Web, Search Engines, Information Retrieval, PageRank, Google

1. Introduction

(Note: There are two versions of this paper -- a longer full version and a shorter printed version. The

PageRank

- A random surfer who reaches their target site after several clicks, then switches to a random page
- The PageRank value of a page: The chance that the random surfer will land on that page by clicking on a link.
- It can be modeled as a Markov chain in which the states are pages, and the transitions are the equally probable links between pages

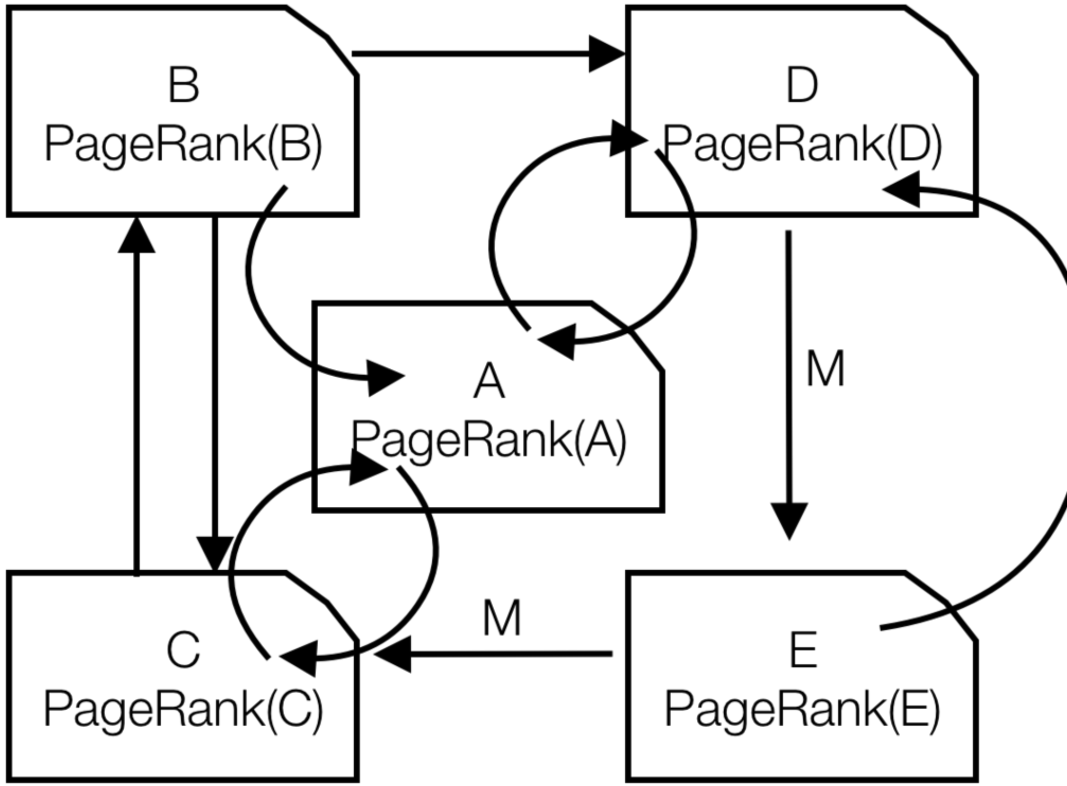
PageRank

- N: number of known webpages
- A page i has k_i links to it
- Transition probability for all pages that are linked to i:

$$\frac{\alpha}{k_i} + \frac{1-\alpha}{N}$$

- Transition probability for all pages that are not linked to i:

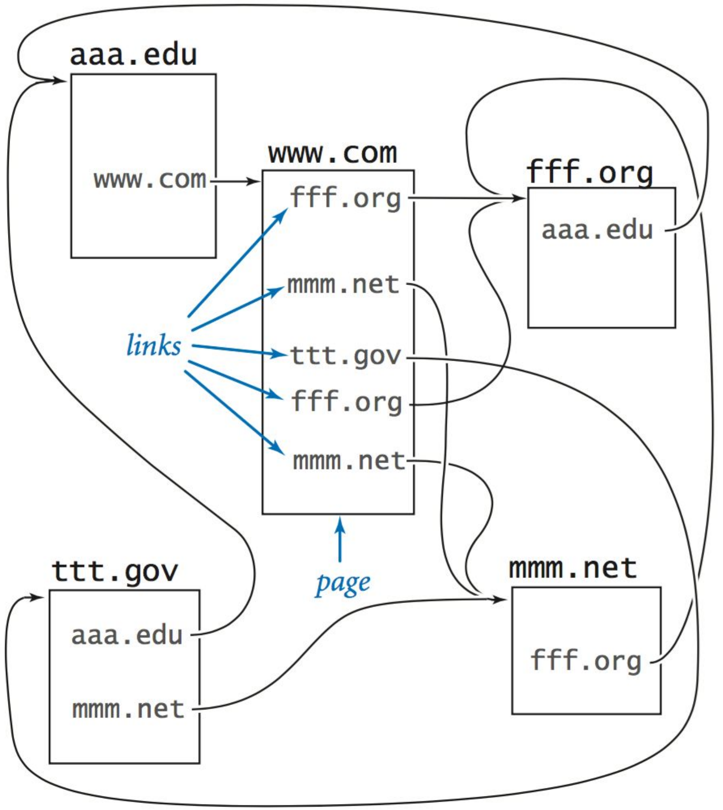
$$\frac{1-\alpha}{N}$$



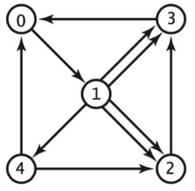
https://en.wikipedia.org/wiki/Markov_chain

PageRank

source:https://introcs.cs.princeton.edu/java/16pagerank/



input graph



```
5
0 1
1 2   1 2
1 3   1 3   1 4
2 3
3 0
4 0   4 2
```

link counts

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

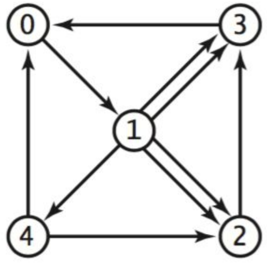
outdegrees

$$\begin{bmatrix} 1 \\ 5 \\ 1 \\ 1 \\ 2 \end{bmatrix}$$

leap probabilities + *link probabilities* = *transition matrix*

$$\begin{bmatrix} .02 & .02 & .02 & .02 & .02 \\ .02 & .02 & .02 & .02 & .02 \\ .02 & .02 & .02 & .02 & .02 \\ .02 & .02 & .02 & .02 & .02 \\ .02 & .02 & .02 & .02 & .02 \end{bmatrix} + \begin{bmatrix} 0 & .90 & 0 & 0 & 0 \\ 0 & 0 & .36 & .36 & .18 \\ 0 & 0 & 0 & .90 & 0 \\ .90 & 0 & 0 & 0 & 0 \\ .45 & 0 & .45 & 0 & 0 \end{bmatrix} = \begin{bmatrix} .02 & .92 & .02 & .02 & .02 \\ .02 & .02 & .38 & .38 & .20 \\ .02 & .02 & .02 & 92 & .02 \\ .92 & .02 & .02 & .02 & .02 \\ .47 & .02 & .47 & .02 & .02 \end{bmatrix}$$

PageRank



source:https://introcs.cs.princeton.edu/java/16pagerank/

p

$$\begin{array}{ccccc}
.02 & .92 & .02 & .02 & .02 \\
.02 & .02 & .38 & .38 & .20 \\
.02 & .02 & .02 & .92 & .02 \\
.92 & .02 & .02 & .02 & .02 \\
.47 & .02 & .47 & .02 & .02
\end{array}$$

probability of surfing from i to 2 in one move

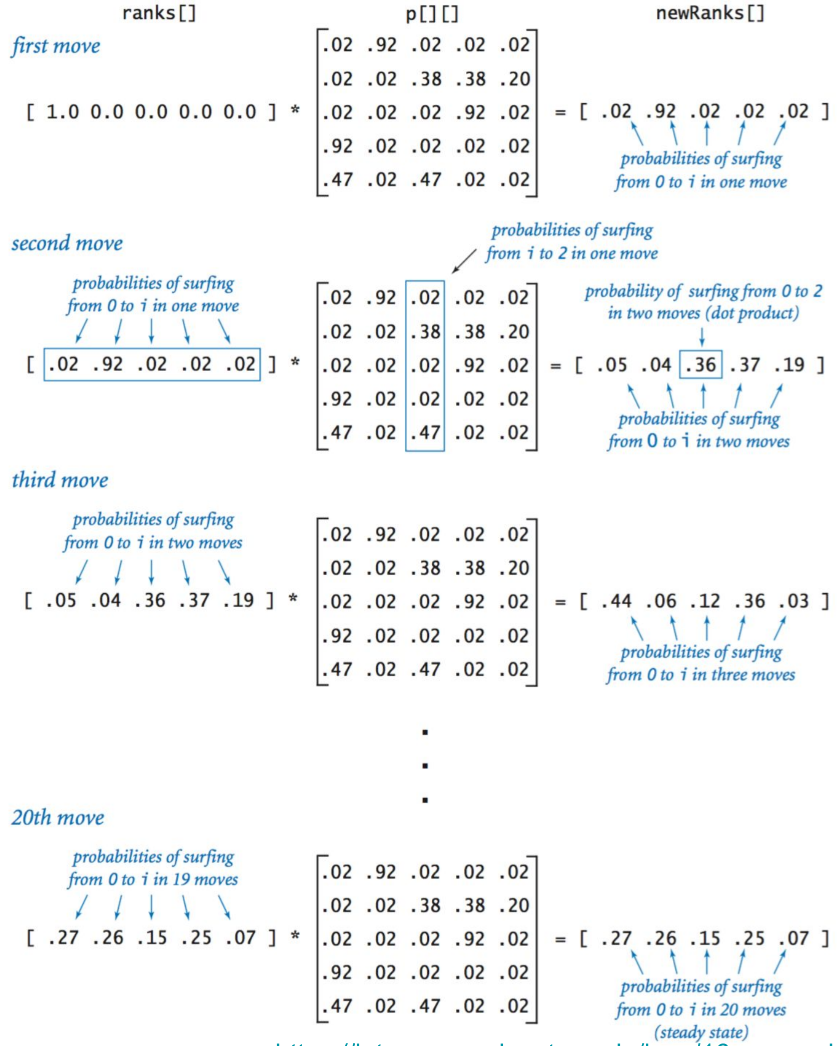
probability of surfing from 1 to i in one move

p^2

$$\begin{array}{ccccc}
.05 & .04 & .36 & .37 & .19 \\
.45 & .04 & \boxed{.12} & .37 & .02 \\
.86 & .04 & .04 & .05 & .02 \\
.05 & .85 & .04 & .05 & .02 \\
.05 & .44 & .04 & .45 & .02
\end{array}$$

probability of surfing from 1 to 2 in two moves (dot product)

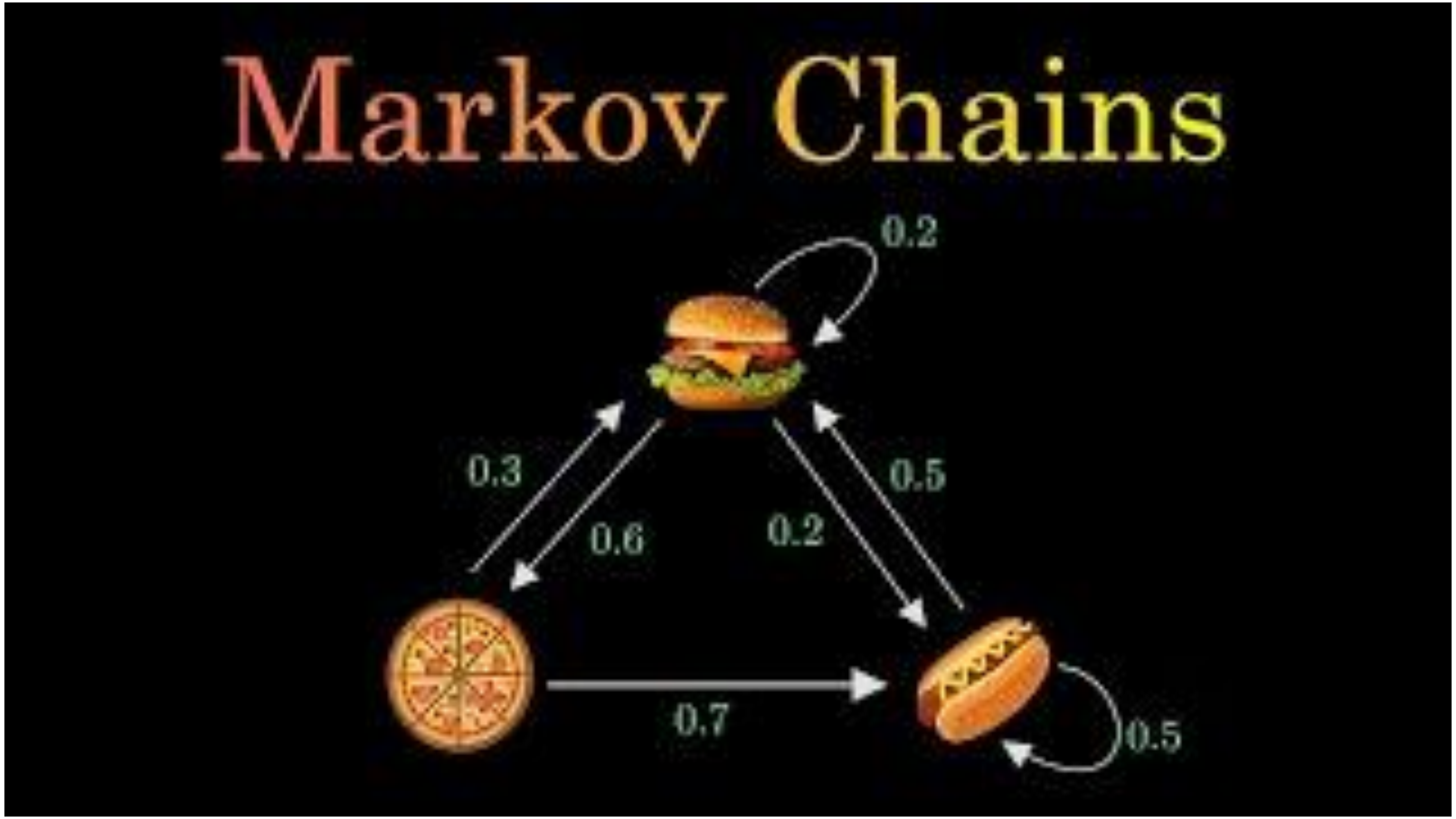
PageRank


ranks[]
p[][]
newRanks[]
first move
[1.0 0.0 0.0 0.0 0.0] *
.02 .92 .02 .02 .02
.02 .02 .38 .38 .20
.02 .02 .02 .92 .02
.92 .02 .02 .02 .02
.47 .02 .47 .02 .02
= [.02 .92 .02 .02 .02]
probabilities of surfing from 0 to i in one move
second move
probabilities of surfing from 0 to i in one move
[.02 .92 .02 .02 .02] *
probabilities of surfing from i to 2 in one move
probability of surfing from 0 to 2 in two moves (dot product)
= [.05 .04 .36 .37 .19]
probabilities of surfing from 0 to i in two moves
third move
probabilities of surfing from 0 to i in two moves
[.05 .04 .36 .37 .19] *
= [.44 .06 .12 .36 .03]
probabilities of surfing from 0 to i in three moves
20th move
probabilities of surfing from 0 to i in 19 moves
[.27 .26 .15 .25 .07] *
= [.27 .26 .15 .25 .07]
probabilities of surfing from 0 to i in 20 moves (steady state)

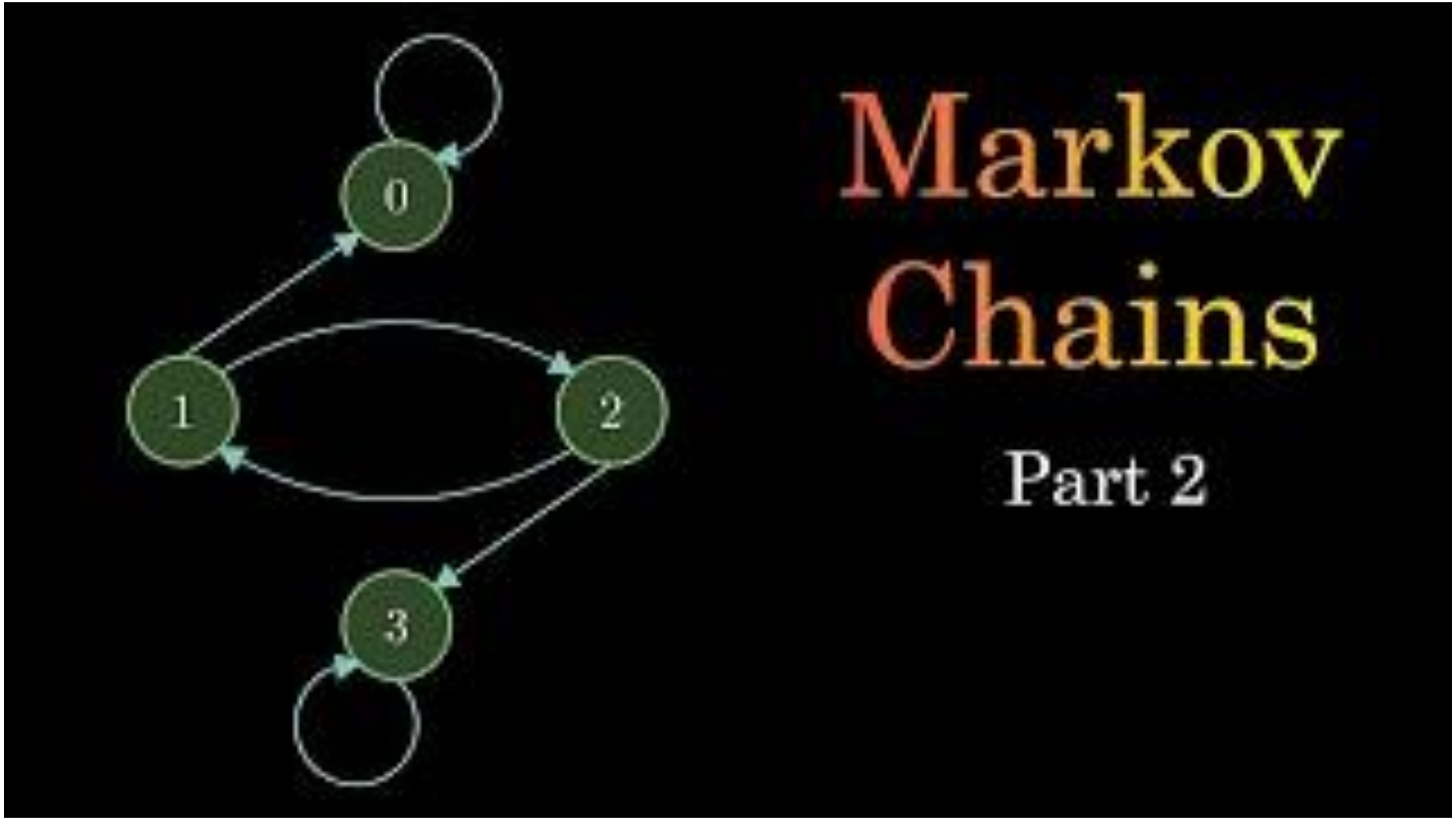
source:https://introcs.cs.princeton.edu/java/16pagerank/

PageRank

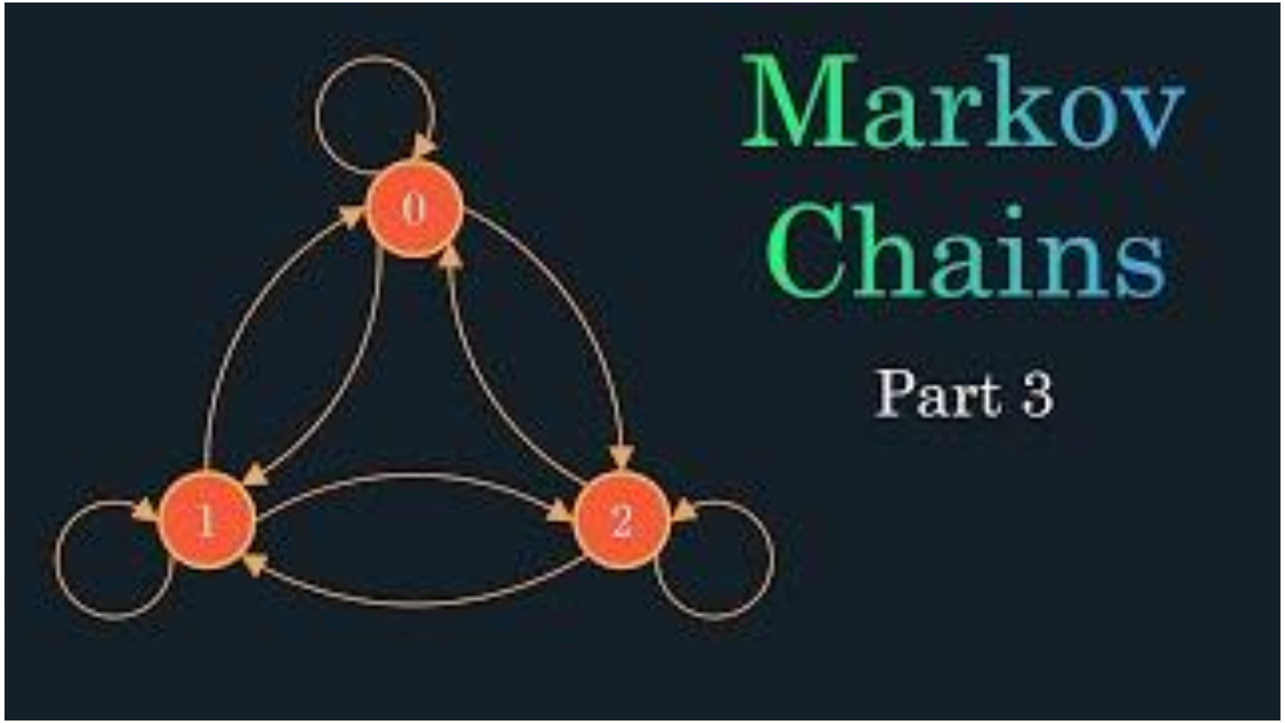
- PageRank
- introduction-to-page-rank notebook
- THE $25,000,000,000∗ EIGENVECTOR THE LINEAR ALGEBRA BEHIND GOOGLE



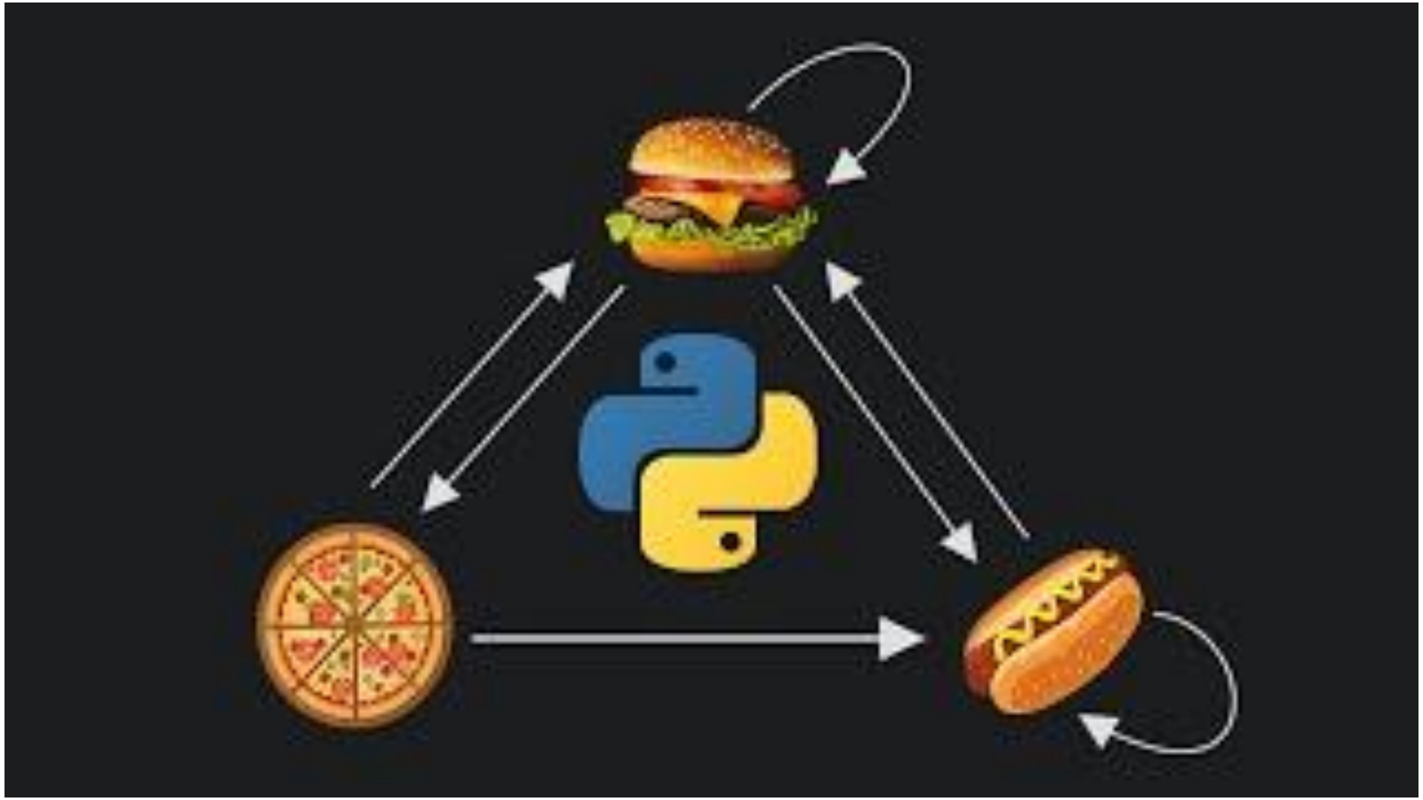
https://youtu.be/i3AkTO9HLXo



https://youtu.be/VNHeFp6zXKU



https://youtu.be/Zo3ieESzr4E

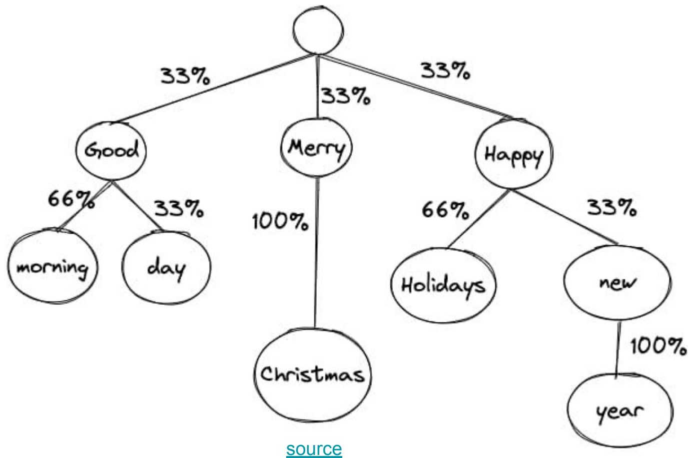


https://youtu.be/G7FIQ9fXl6U

Claude Shannon’s Markov Chain Text Generator

- In this paper, Shannon proposed using a Markov chain to create a statistical model of the sequences of letters in a piece of English text.

  “ To construct the Markov model of order 1, for example, one opens a book at random and selects a letter at random on the page. This letter is recorded. The book is then opened to another page and one reads until this letter is encountered. The succeeding letter is then recorded. Turning to another page this second letter is searched for and the succeeding letter recorded, etc. It would be interesting if further approximations could be constructed, but the labor involved becomes enormous at the next stage.”



source

Markov Chain as a Generative Language Model

https://youtu.be/E4WcBWuQQws

Markov Text Generation

- N-grams and Markov chains
- Markov-Shannon, a simple text generator
- Text generation with Markovify
- Build a Sentence Generator Markov Chain in 20 lines of Python

Hidden Markov Models

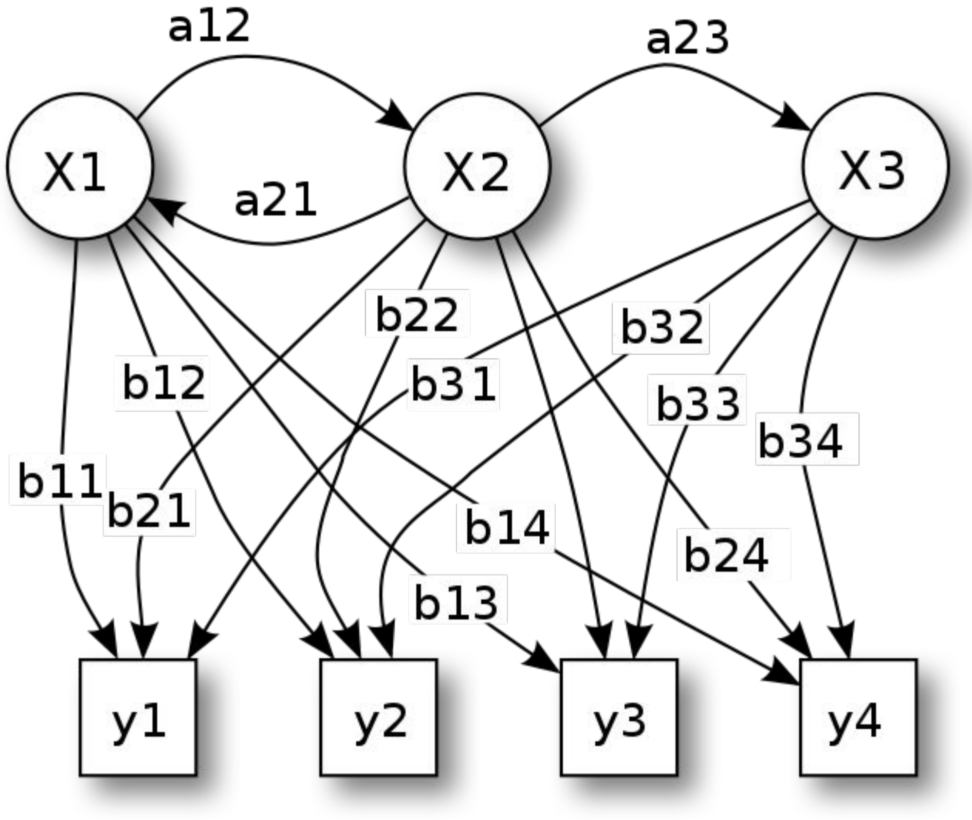
Hidden Markov Model (HMM)

- There is an observable process Y whose outcomes are "influenced" by the outcomes of X in a known way.
- X cannot be observed directly (hidden state)
    - X is a Markov process whose behavior is not directly observable
- The goal is to learn about X by observing Y
- HMM main assumption: the outcome of Y at time $t = t_0$ must be "influenced" exclusively by the outcome of X at $t = t_0$ and that the outcomes of X and Y at $t < t_0$ must be conditionally independent of Y at $t = t_0$ given X at time $t = t_0$

$$\mathbf{P}\left(Y_n \in A \mid X_1 = x_1, \ldots, X_n = x_n\right) = \mathbf{P}\left(Y_n \in A \mid X_n = x_n\right)$$

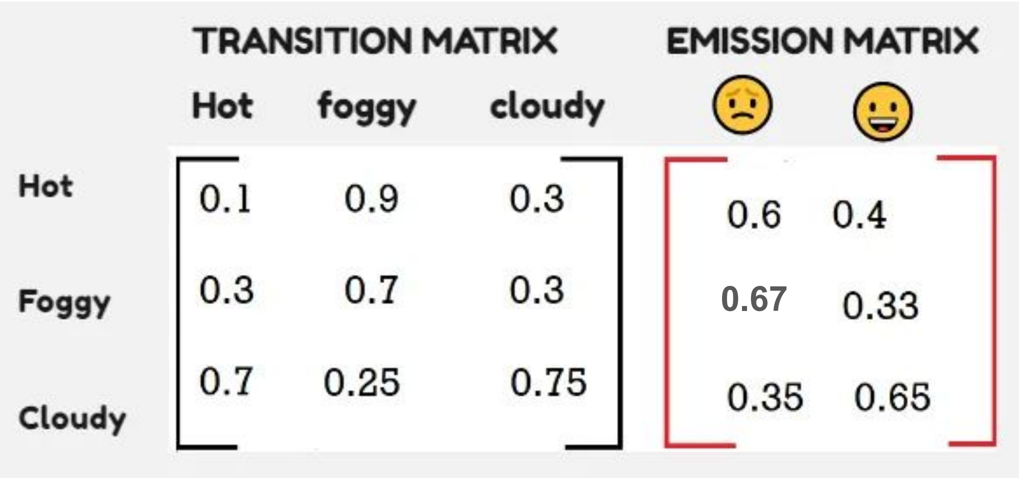
HMM Parameters

- X — states (hidden)
- y — possible observations
- a — state transition probabilities
- b — output probabilities



https://en.wikipedia.org/wiki/Hidden_Markov_model

Emission Matrix

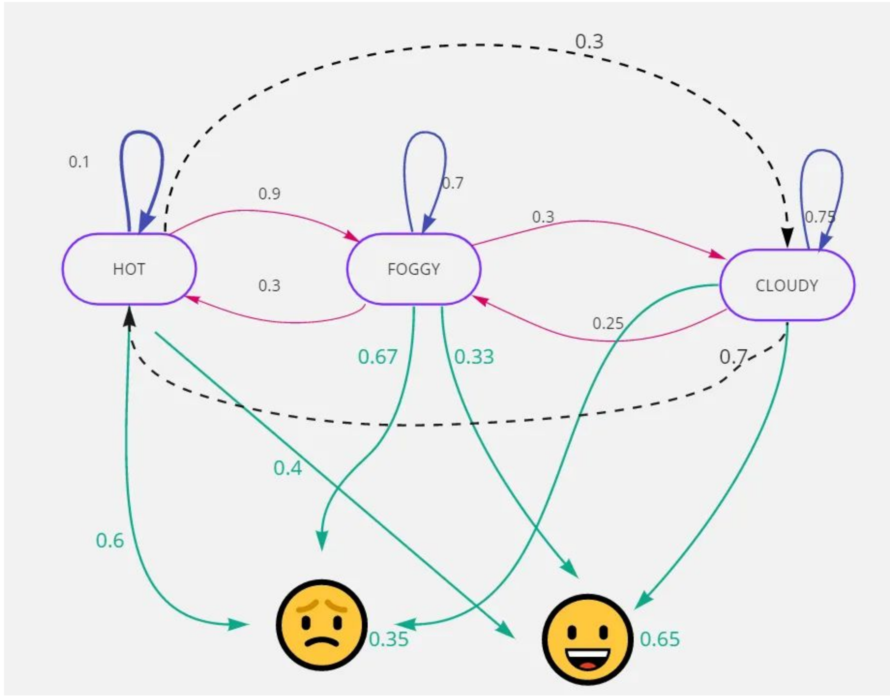


TRANSITION MATRIX

	Hot	foggy	cloudy
Hot	0.1	0.9	0.3
Foggy	0.3	0.7	0.3
Cloudy	0.7	0.25	0.75

EMISSION MATRIX

	😟	😀
Hot	0.6	0.4
Foggy	0.67	0.33
Cloudy	0.35	0.65



source

HMM Example: Weather guessing game

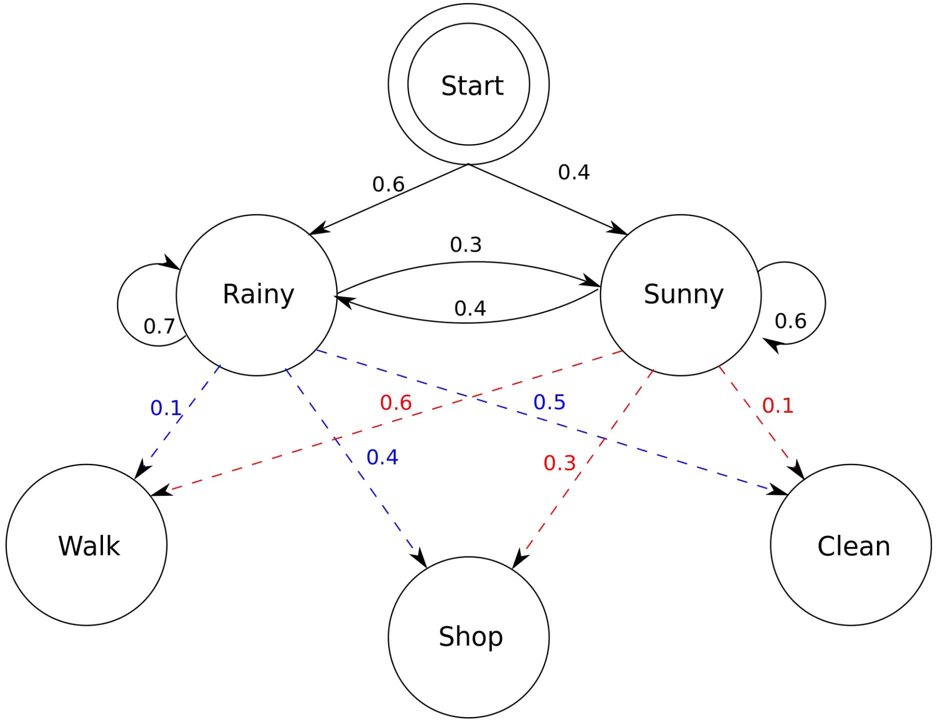
- Consider two friends, Alice and Bob, who live far apart from each other and who talk together daily over the telephone about what they did that day.
- Bob is only interested in three activities: walking in the park, shopping, and cleaning his apartment. The choice of what to do is determined exclusively by the weather on a given day.
- Alice has no definite information about the weather, but she knows general trends. Based on what Bob tells her he did each day, Alice tries to guess what the weather must have been like.

source

source

HMM Example: Weather Guessing Game

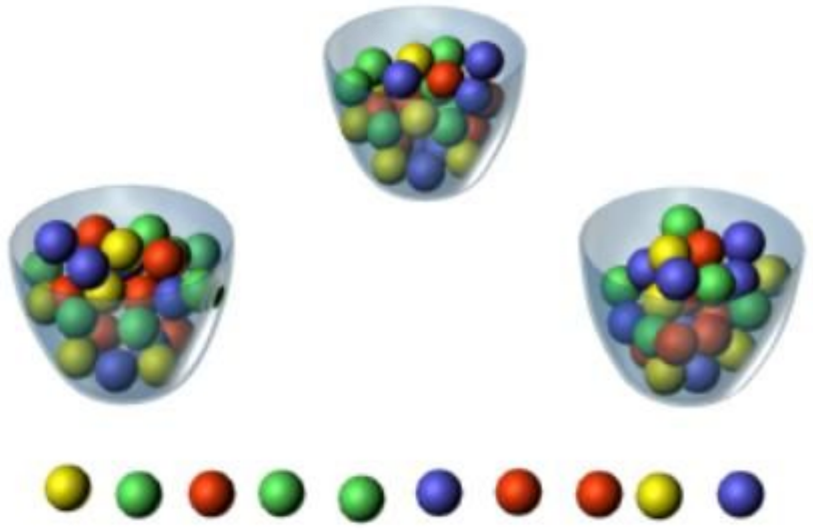
- Alice believes that the weather operates as a discrete Markov chain.
- There are two states, "Rainy" and "Sunny", but she cannot observe them directly, that is, they are hidden from her.
- On each day, there is a certain chance that Bob will perform one of the following activities, depending on the weather: "walk", "shop", or "clean".
- Since Bob tells Alice about his activities, those are the observations.
- The entire system is that of a hidden Markov model (HMM). Alice knows the general weather trends in the area, and what Bob likes to do on average. In other words, the parameters of the HMM are known.



https://en.wikipedia.org/wiki/Hidden_Markov_model

HMM Example: Drawing balls from hidden urns

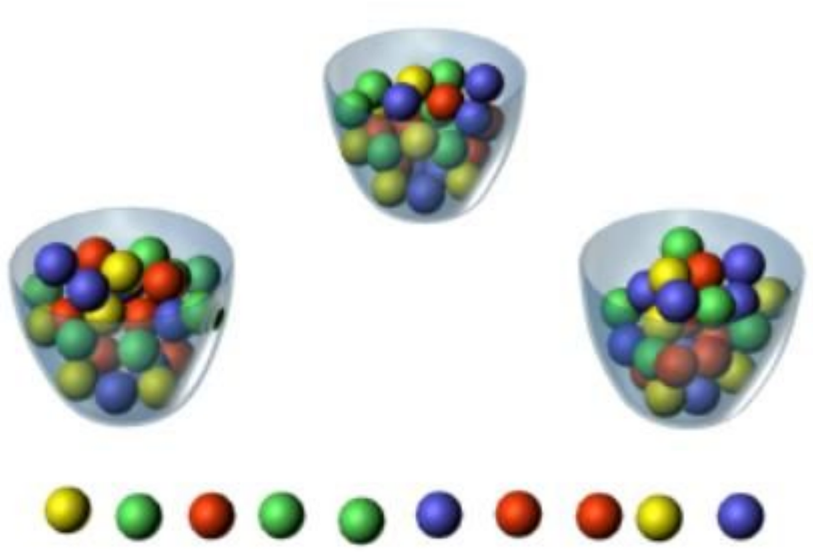
- In a room that is not visible to an observer there is a genie. The room contains urns X1, X2, X3, ... each of which contains a known mix of balls, each ball labeled y1, y2, y3, ...
- The genie chooses an urn in that room and randomly draws a ball from that urn. It then puts the ball onto a conveyor belt, where the observer can observe the sequence of the balls but not the sequence of urns from which they were drawn.



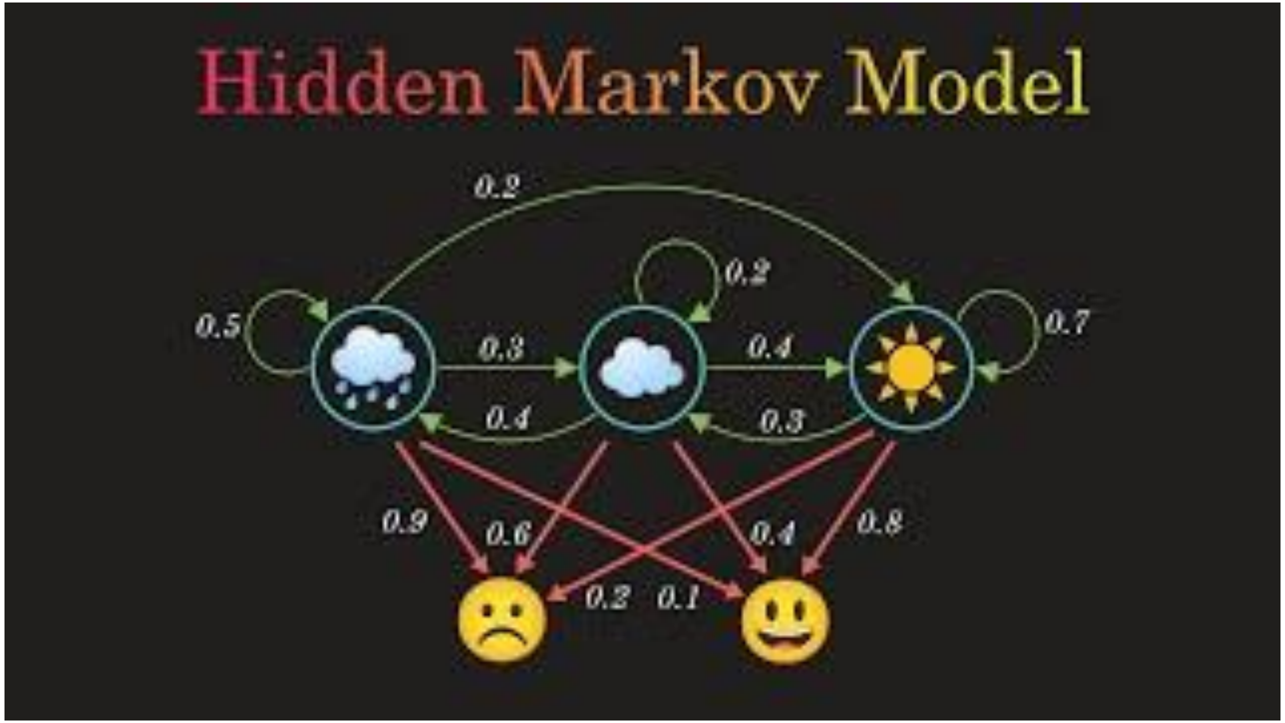
source

HMM Example: Drawing balls from hidden urns

- The genie has some procedure to choose urns; the choice of the urn for the n-th ball depends only upon a random number and the choice of the urn for the (n − 1)-th ball. The choice of urn does not directly depend on the urns chosen before this single previous urn; therefore, this is called a Markov process.
  - Balls on the belt: observations
  - Urns: hidden state



source



https://youtu.be/RWkHJnFj5rY

Three Important HMM Tasks

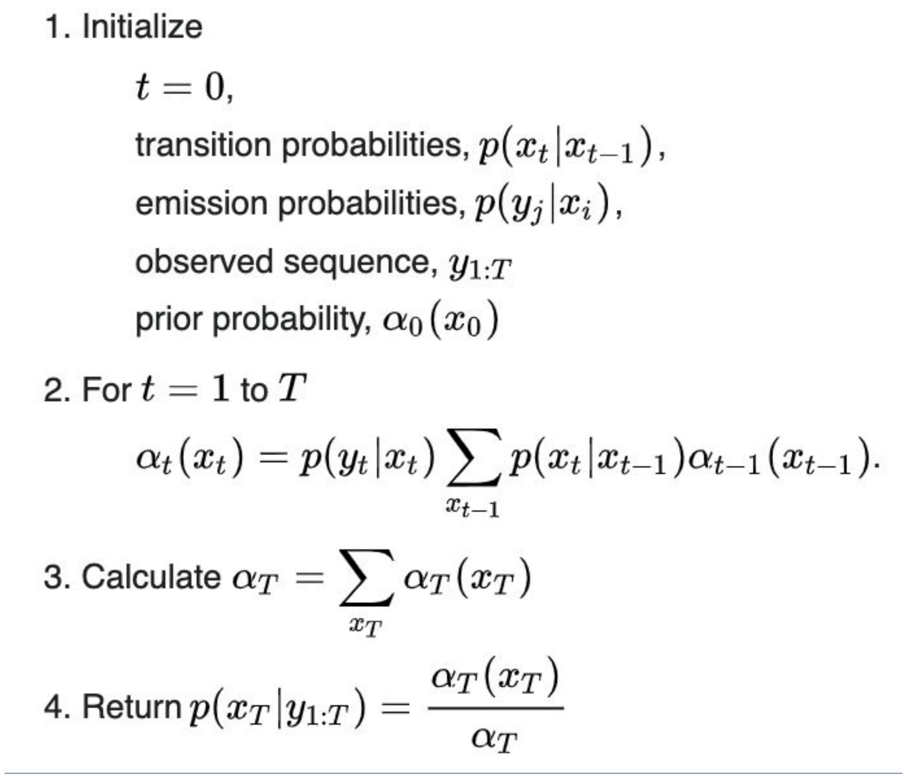
1. **Most likely explanation for a sequence (Decoding)**
   - Estimate the joint probability of the entire sequence of hidden states that generated a particular sequence of observations.
   - An example is part-of-speech tagging, where the hidden states represent the underlying parts of speech corresponding to an observed sequence of words.
   - Viterbi algorithm
2. **Probability of an observed sequence (Evaluation)**
   - The task is to compute in a best way, given the parameters of the model, the probability of a particular output sequence.
   - Forward algorithm
3. **Parameter learning (Training)**
   - Given an output sequence or a set of such sequences, the best set of state transition and emission probabilities.
   - Maximum likelihood estimate of the parameters of the HMM given the set of output sequences
   - Baum–Welch algorithm: a special case of the expectation-maximization

Forward Algorithm

- Is used to calculate a 'belief state'
  - The probability of a state at a certain time, given the history of evidence.
  - $p(x_t|y_{1:t})$: x(t) is the hidden state which is abbreviated as x_t and $y_{1:t}$ are the observations 1 to t.
- Also known as filtering
- The forward algorithm is closely related to, but distinct from, the Viterbi algorithm.

$$\begin{tikzcd}
\text{----} \arrow[r] & x(t-1) \arrow[r] \arrow[d] & x(t) \arrow[r] \arrow[d] & x(t+1) \arrow[r] \arrow[d] & \text{----} \\
 & y(t-1) & y(t) & y(t+1) &
\end{tikzcd}$$

Forward Algorithm



1. Initialize

   $t = 0,$

   transition probabilities, $p(x_t|x_{t-1})$,

   emission probabilities, $p(y_j|x_i)$,

   observed sequence, $y_{1:T}$

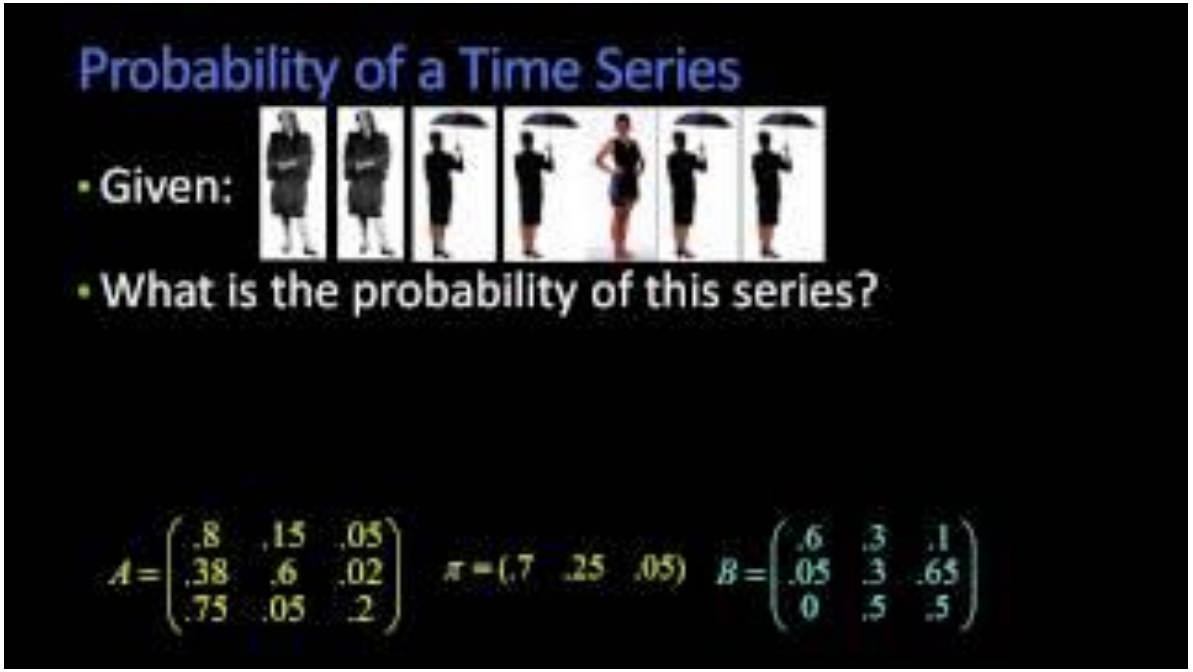
   prior probability, $\alpha_0(x_0)$

2. For $t = 1$ to T

   $$\alpha_t(x_t) = p(y_t|x_t)\sum_{x_{t-1}} p(x_t|x_{t-1})\alpha_{t-1}(x_{t-1}).$$

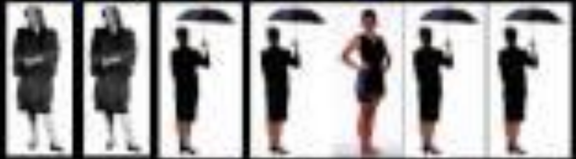
3. Calculate $\alpha_T = \sum_{x_T} \alpha_T(x_T)$

4. Return $p(x_T|y_{1:T}) = \dfrac{\alpha_T(x_T)}{\alpha_T}$



Probability of a Time Series

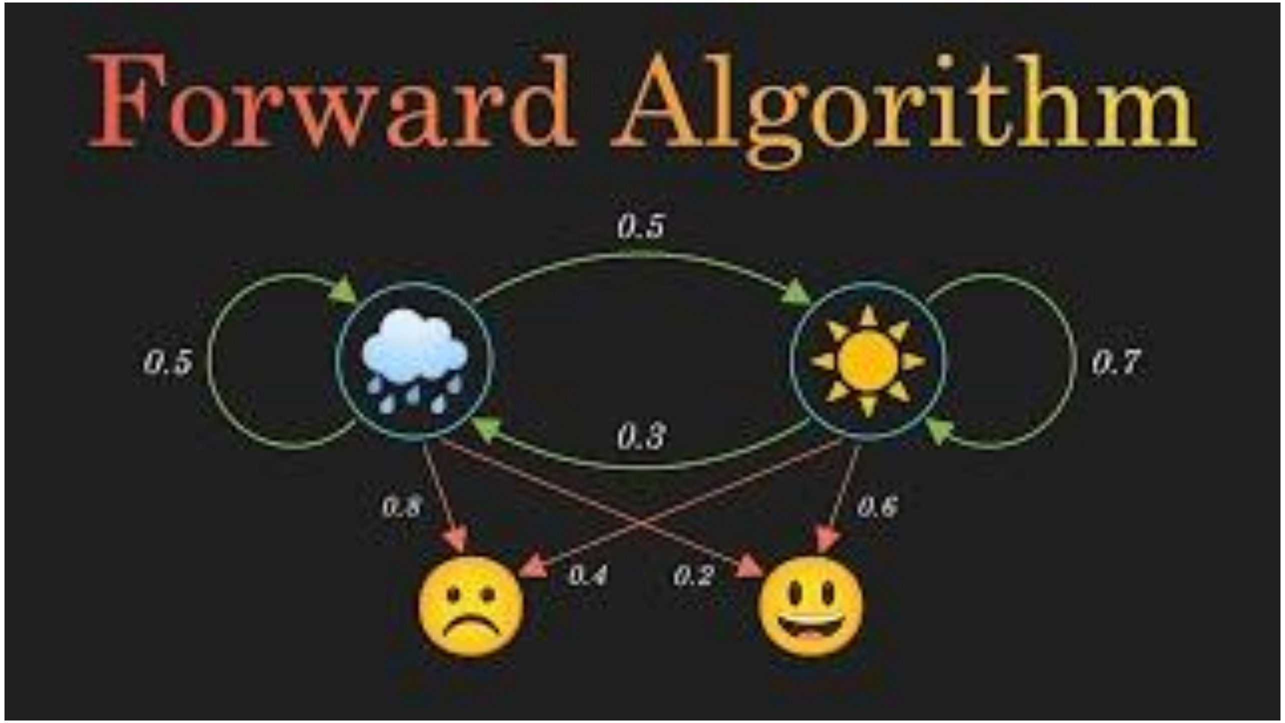
- Given:



- What is the probability of this series?

$$A=\begin{pmatrix} .8 & .15 & .05 \\ .38 & .6 & .02 \\ .75 & .05 & .2 \end{pmatrix} \quad \pi=(.7 \quad .25 \quad .05) \quad B=\begin{pmatrix} .6 & .3 & .1 \\ .05 & .3 & .65 \\ 0 & .5 & .5 \end{pmatrix}$$

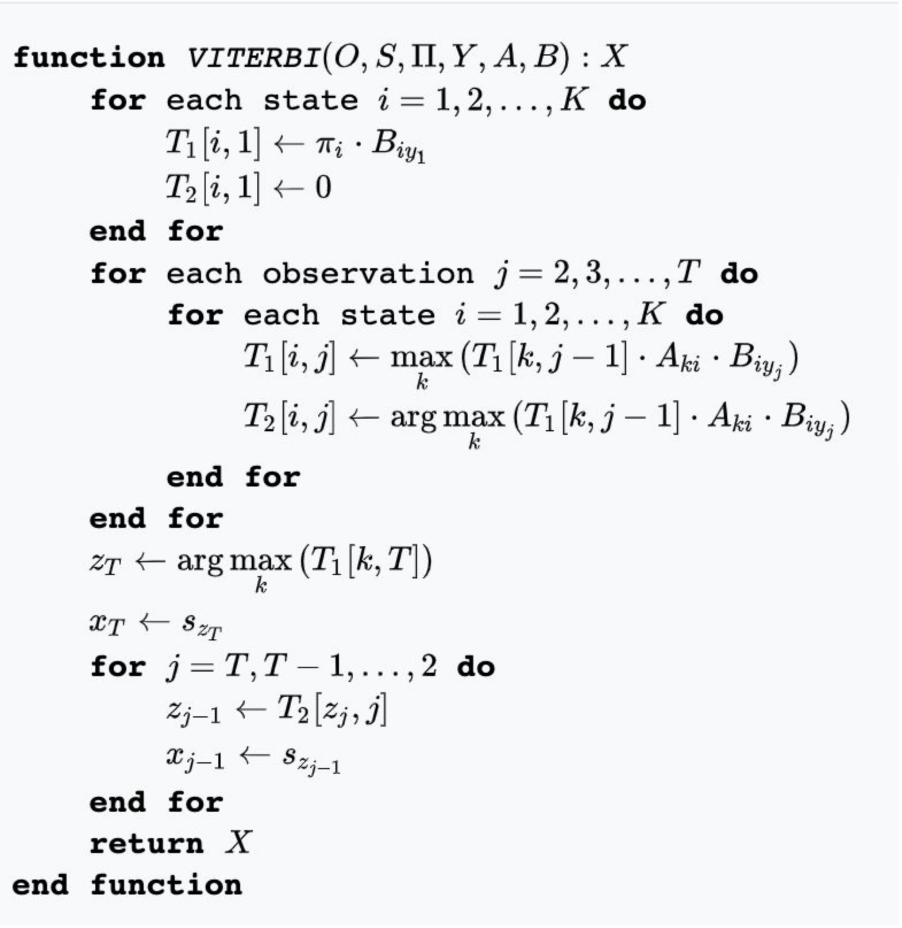
https://youtu.be/5araDjcBHMQ



https://youtu.be/9-sPm4CfcD0

Viterbi Algorithm (VA)

- It computes the most likely state sequence given the history of observations
  - Viterbi path: The maximum a posteriori probability estimate of the most likely sequence of hidden states
  - The state sequence that maximizes $p(x_{0:t}|y_{0:t})$
- Dynamic programming algorithm



```
function $VITERBI(O, S, \Pi, Y, A, B) : X$
    for each state $i = 1, 2, \ldots, K$ do
        $T_1[i, 1] \leftarrow \pi_i \cdot B_{iy_1}$
        $T_2[i, 1] \leftarrow 0$
    end for
    for each observation $j = 2, 3, \ldots, T$ do
        for each state $i = 1, 2, \ldots, K$ do
            $T_1[i, j] \leftarrow \max_k (T_1[k, j-1] \cdot A_{ki} \cdot B_{iy_j})$
            $T_2[i, j] \leftarrow \arg\max_k (T_1[k, j-1] \cdot A_{ki} \cdot B_{iy_j})$
        end for
    end for
    $z_T \leftarrow \arg\max_k (T_1[k, T])$
    $x_T \leftarrow s_{z_T}$
    for $j = T, T-1, \ldots, 2$ do
        $z_{j-1} \leftarrow T_2[z_j, j]$
        $x_{j-1} \leftarrow s_{z_{j-1}}$
    end for
    return $X$
end function
```

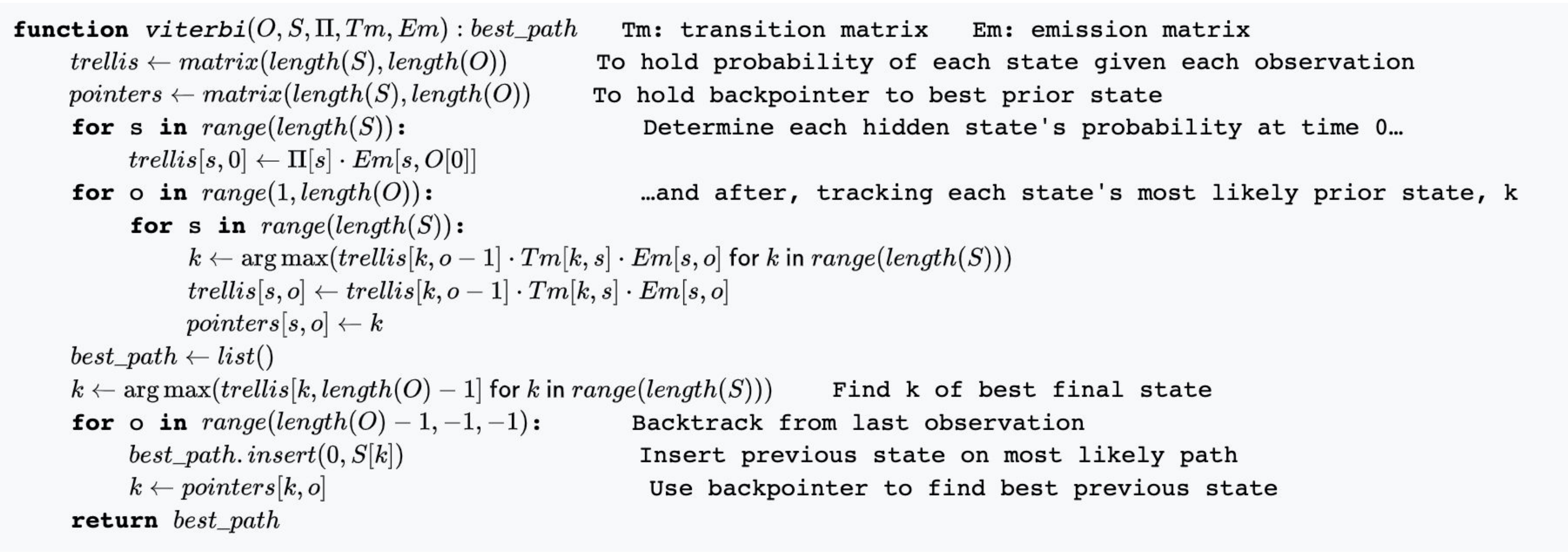
https://en.wikipedia.org/wiki/Viterbi_algorithm

Input

- The observation space $O = \{o_1, o_2, \ldots, o_N\}$,
- the state space $S = \{s_1, s_2, \ldots, s_K\}$,
- an array of initial probabilities $\Pi = (\pi_1, \pi_2, \ldots, \pi_K)$ such that π_i stores the probability that $x_1 = s_i$,
- a sequence of observations $Y = (y_1, y_2, \ldots, y_T)$ such that $y_t = o_i$ if the observation at time t is o_i,
- transition matrix A of size $K \times K$ such that A_{ij} stores the transition probability of transiting from state s_i to state s_j,
- emission matrix B of size $K \times N$ such that B_{ij} stores the probability of observing o_j from state s_i.

Output

- The most likely hidden state sequence $X = (x_1, x_2, \ldots, x_T)$

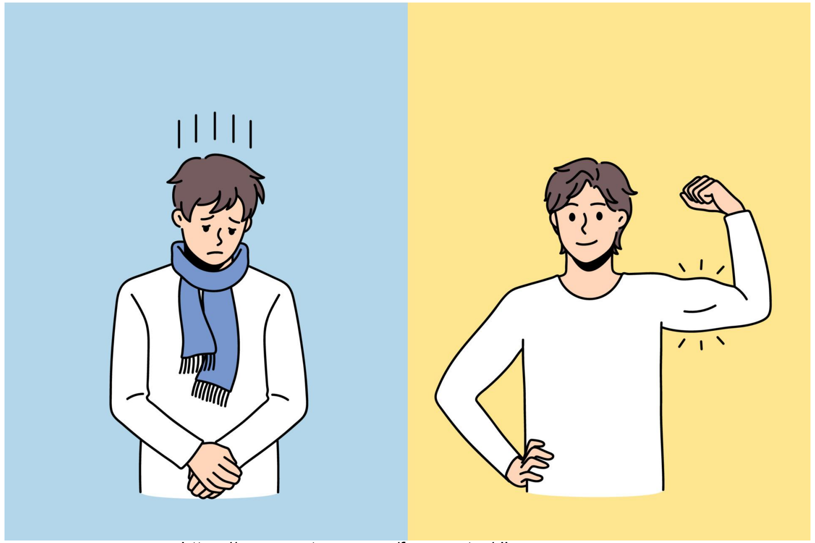


```
function $viterbi(O, S, \Pi, Tm, Em) : best\_path$    Tm: transition matrix   Em: emission matrix
    $trellis \leftarrow matrix(length(S), length(O))$      To hold probability of each state given each observation
    $pointers \leftarrow matrix(length(S), length(O))$     To hold backpointer to best prior state
    for s in $range(length(S))$:                          Determine each hidden state's probability at time 0…
        $trellis[s, 0] \leftarrow \Pi[s] \cdot Em[s, O[0]]$
    for o in $range(1, length(O))$:                       …and after, tracking each state's most likely prior state, k
        for s in $range(length(S))$:
            $k \leftarrow \arg\max(trellis[k, o-1] \cdot Tm[k, s] \cdot Em[s, o]$ for $k$ in $range(length(S)))$
            $trellis[s, o] \leftarrow trellis[k, o-1] \cdot Tm[k, s] \cdot Em[s, o]$
            $pointers[s, o] \leftarrow k$
    $best\_path \leftarrow list()$
    $k \leftarrow \arg\max(trellis[k, length(O)-1]$ for $k$ in $range(length(S)))$      Find k of best final state
    for o in $range(length(O)-1, -1, -1)$:                Backtrack from last observation
        $best\_path.insert(0, S[k])$                      Insert previous state on most likely path
        $k \leftarrow pointers[k, o]$                     Use backpointer to find best previous state
    return $best\_path$
```

https://en.wikipedia.org/wiki/Viterbi_algorithm

Viterbi: Example

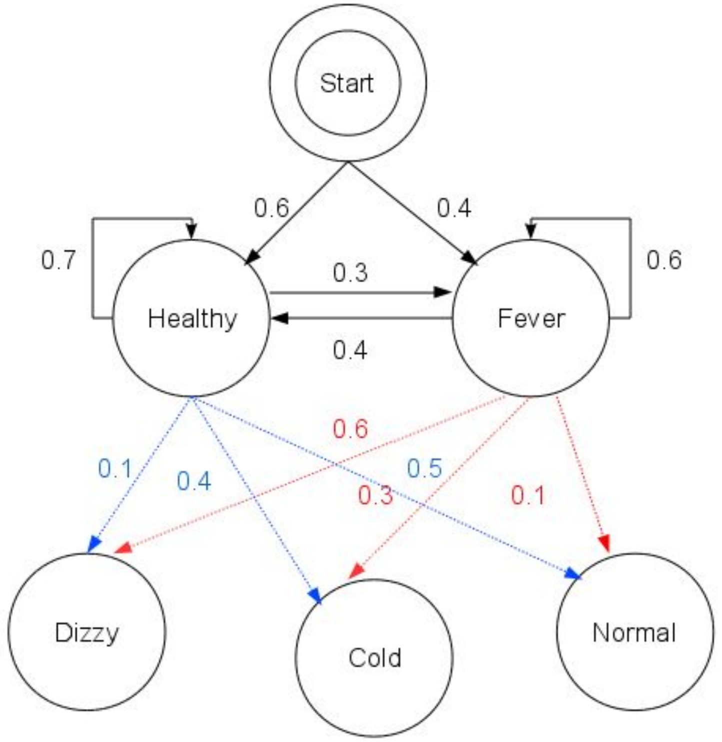
- Consider a village where all villagers are either healthy or have a fever
- Only the village doctor can determine whether each has a fever. The doctor diagnoses fever by asking patients how they feel.The villagers may only answer that they feel normal, dizzy, or cold.



https://www.vecteezy.com/free-vector/disease

Viterbi: Example

- The doctor believes that the health condition of the patients operates as a discrete Markov chain. There are two states, "Healthy" and "Fever", but the doctor cannot observe them directly; they are hidden from the doctor. On each day, there is a certain chance that a patient will tell the doctor "I feel normal", "I feel cold", or "I feel dizzy", depending on the patient's health condition.



https://en.wikipedia.org/wiki/Viterbi_algorithm

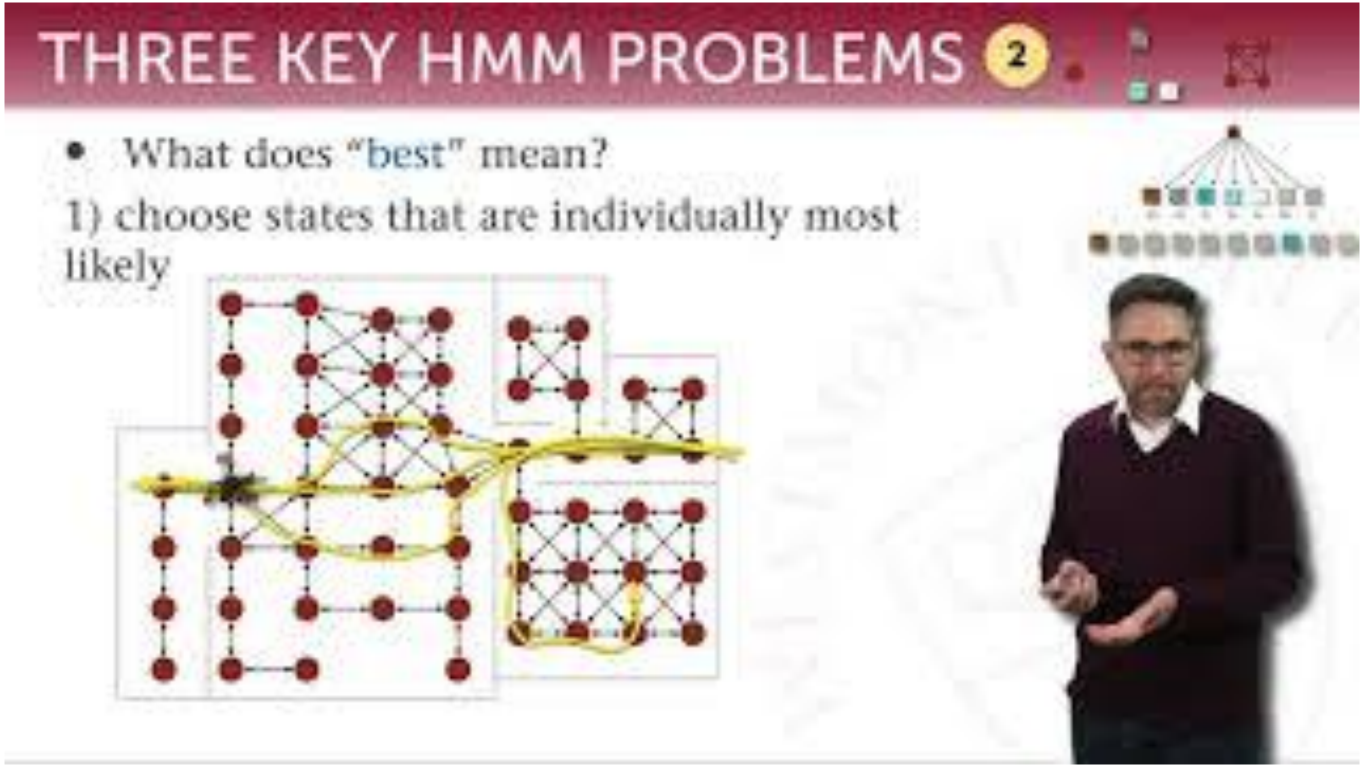
Viterbi: Example

- A patient visits three days in a row, and the doctor discovers that the patient feels normal on the first day, cold on the second day, and dizzy on the third day. The doctor has a question: what is the most likely sequence of health conditions of the patient that would explain these observations?
- Check village fever Jupyter notebook

Baum-Welch Algorithm

- A special case of the expectation–maximization algorithm used to find the unknown parameters of an HMM
- Uses EM to find the maximum likelihood estimate of the parameters of a hidden Markov model given a set of observed feature vectors
- In the the expectation step, it uses the forward-backward algorithm

Viterbi



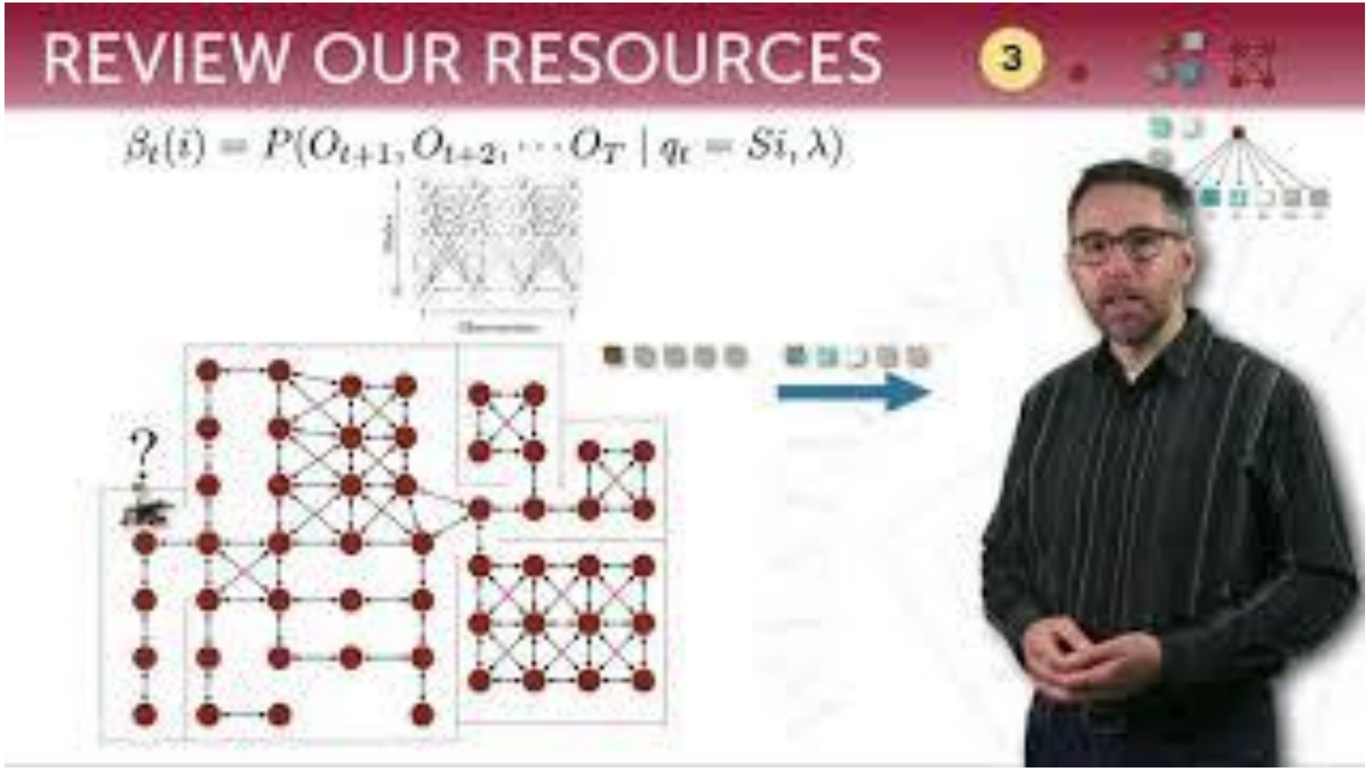
THREE KEY HMM PROBLEMS 2

- What does "best" mean?

1) choose states that are individually most likely

https://youtu.be/s9dU3sFeE40

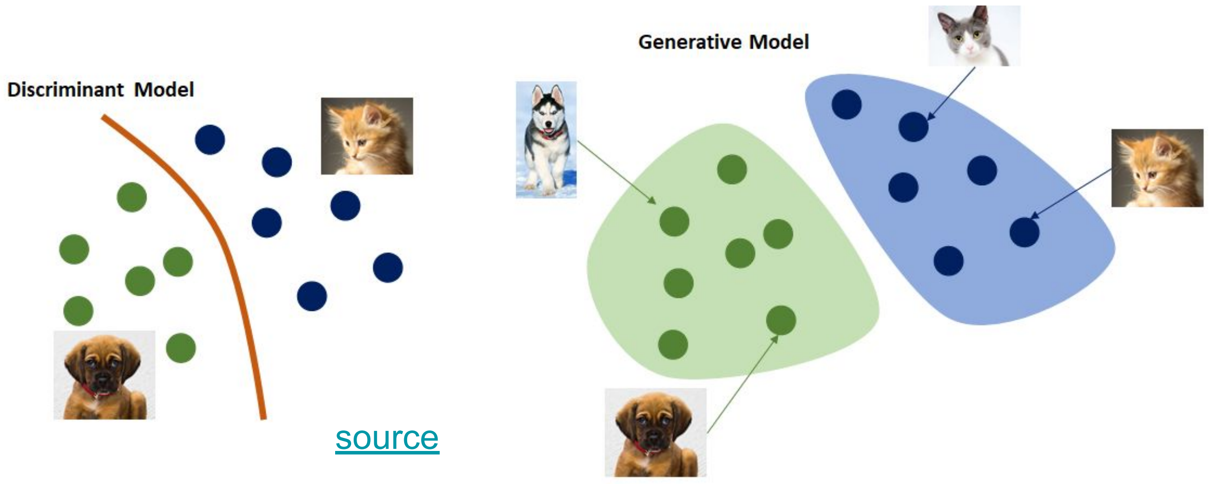
Baum-Welch


REVIEW OUR RESOURCES
3
$\beta_t(i) = P(O_{t+1}, O_{t+2}, \cdots O_T \mid q_t = Si, \lambda)$
?

https://youtu.be/JRsdt05pMoI

HMMs as a Generative Model

- HMMs are generative models which try to learn the joint distribution p(x,y)
    - A sequence of observable X variable is generated by a sequence of internal hidden state y
- It performs classification by finding the model which most likely has produced a given sequence
- A generative model can be used to generate new data points/sequences



source

HMM Python Libraries

- https://pypi.org/project/hmms/
- https://github.com/hmmlearn/hmmlearn
- https://pyhhmm.readthedocs.io/en/latest/

HMM: Example Analyses

Speech Recognition

- Hidden states: phonemes
- Visible observations: Mel Frequency Cepstral Coefficients (MFCC) of the input
- Sequence of phonemes → word
- We can train a separate GaussianHMM for each word
  - We will apply forward algorithm for a given input on all HMM models(one per word) and the maximum score HMM will be the prediction

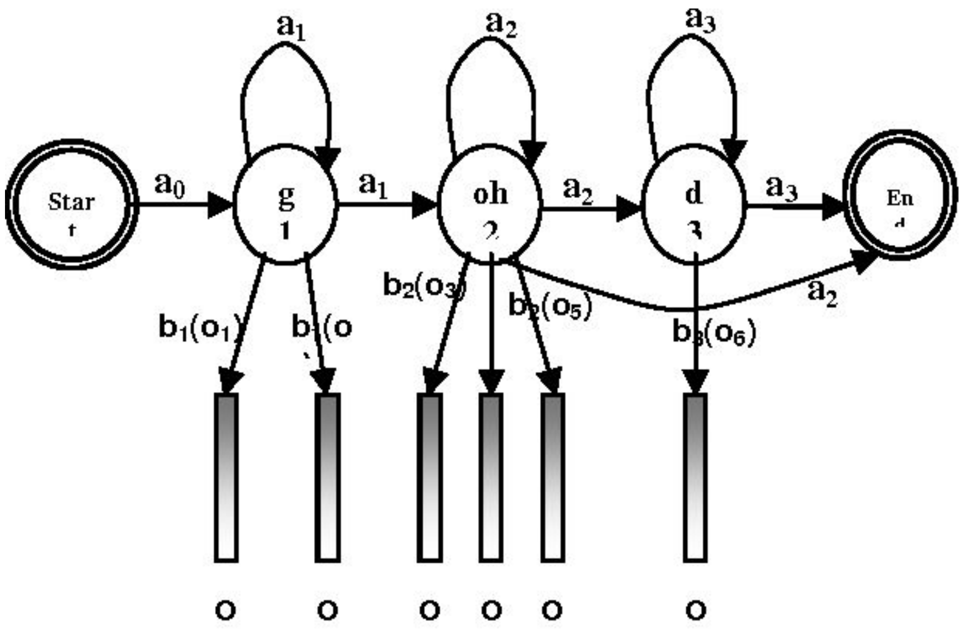


Figure 1: Word generation HMM automata for the word “god”

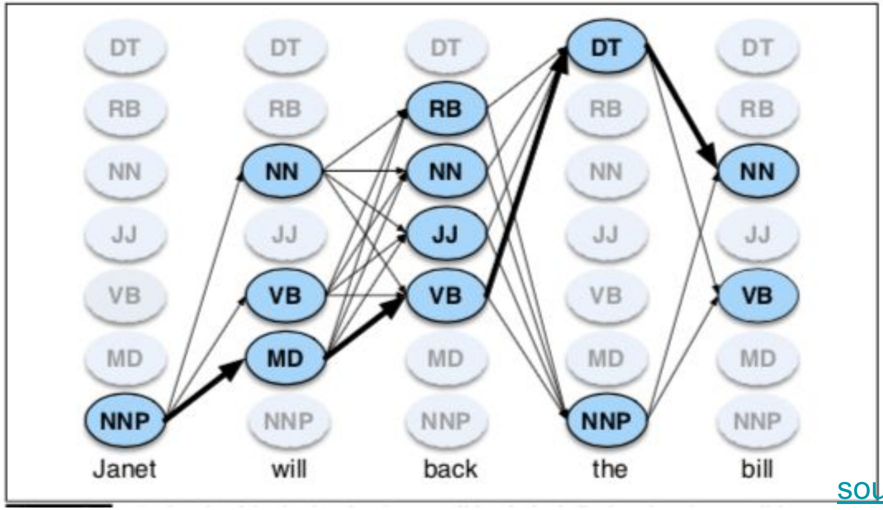
source

Speech Recognition

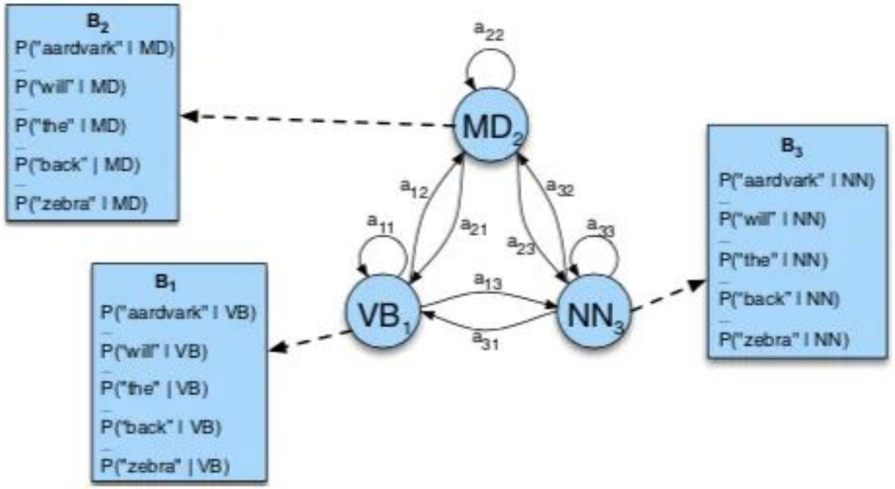
- Speech recognition using hmmlearn library: 7 fruits
- Speech recognition from scratch code in Python: 7 fruits

Part-of-Speech (POS) Tagging

- Visible Observations: The words in a sentence as Observable States
- Hidden states: POS Tags
- We can use Viterbi algorithm: Given a sequence of observations (Words in a sentence), find the most probable sequence of states (POS Tags)



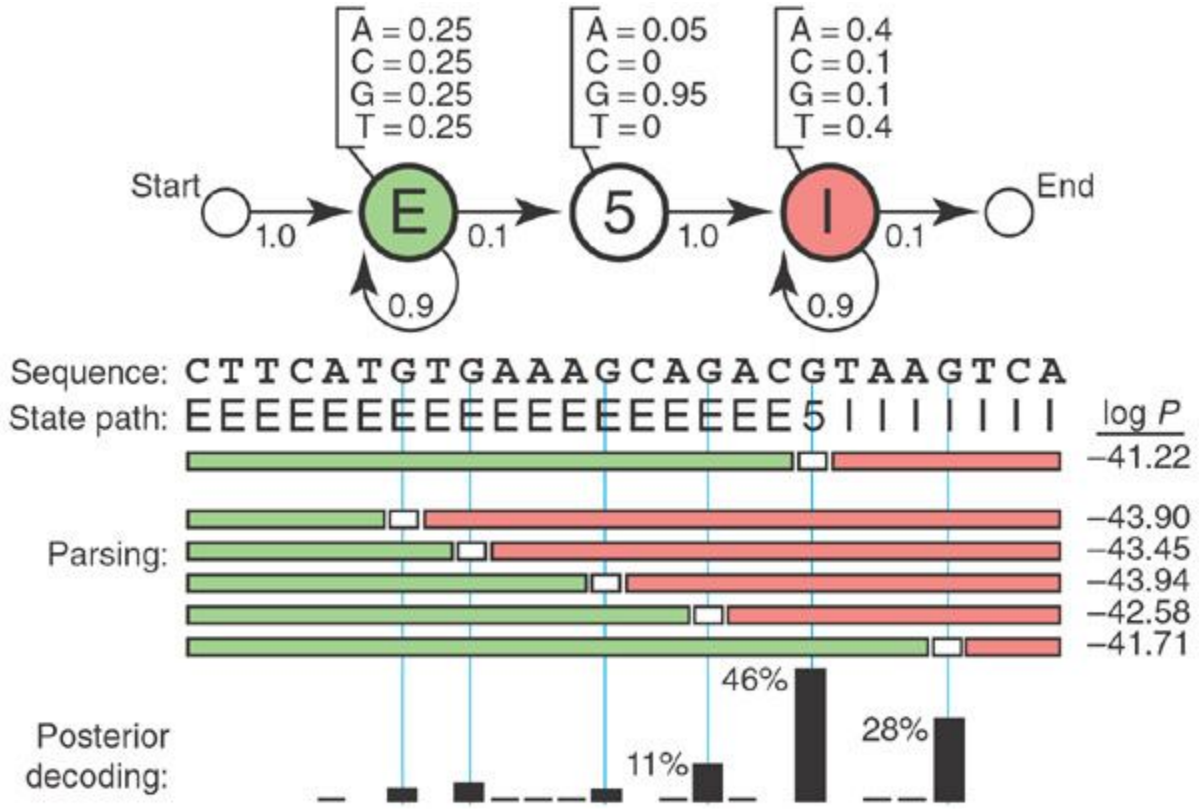
source



POS Tagging

- POS Tagging using Hidden Markov Models (HMM) & Viterbi algorithm in NLP mathematics explained
- https://wisdomml.in/hidden-markov-model-hmm-in-nlp-python/
- https://spotintelligence.com/2023/01/05/hidden-markov-model-hmm-nlp/

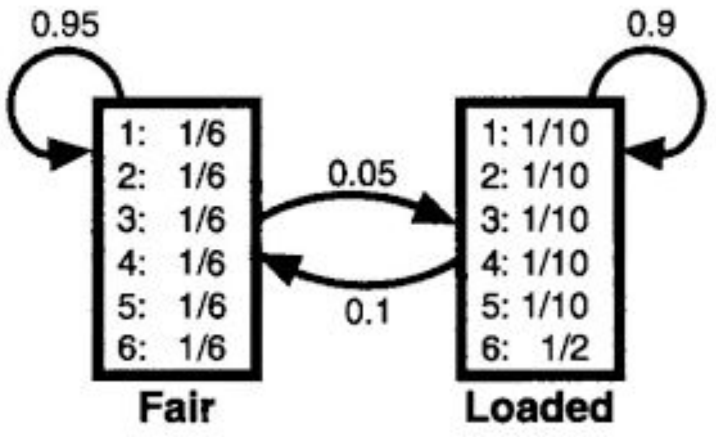
Gene Prediction



Source

Dishonest Casino Example

- Dishonest Casino Example
  - Hidden States: What the casino did (Fair,Loaded,Fair,Fair….)
  - Visible Observation : The series of die tosses
  - Generative model first to generate the sequence in the code



HMM Text Generator

- To generate text from using HMM, we will use the part-of-speech tags as the hidden state and the words will be the outputs emitted at each state.
- We need to calculate:
  - The transition matrix: the probability of jumping from one tag to another
  - The emission matrix: the probability of generating a word given the POS tag

Demo: Text Generation Using HMM

- http://localhost:8888/notebooks/jupyter-notebooks/chapman-generative-AI/HMM-text-generation.ipynb