Computers playing Jeopardy!

CSE 392, Computers Playing Jeopardy!, Fall 2011

Stony Brook University

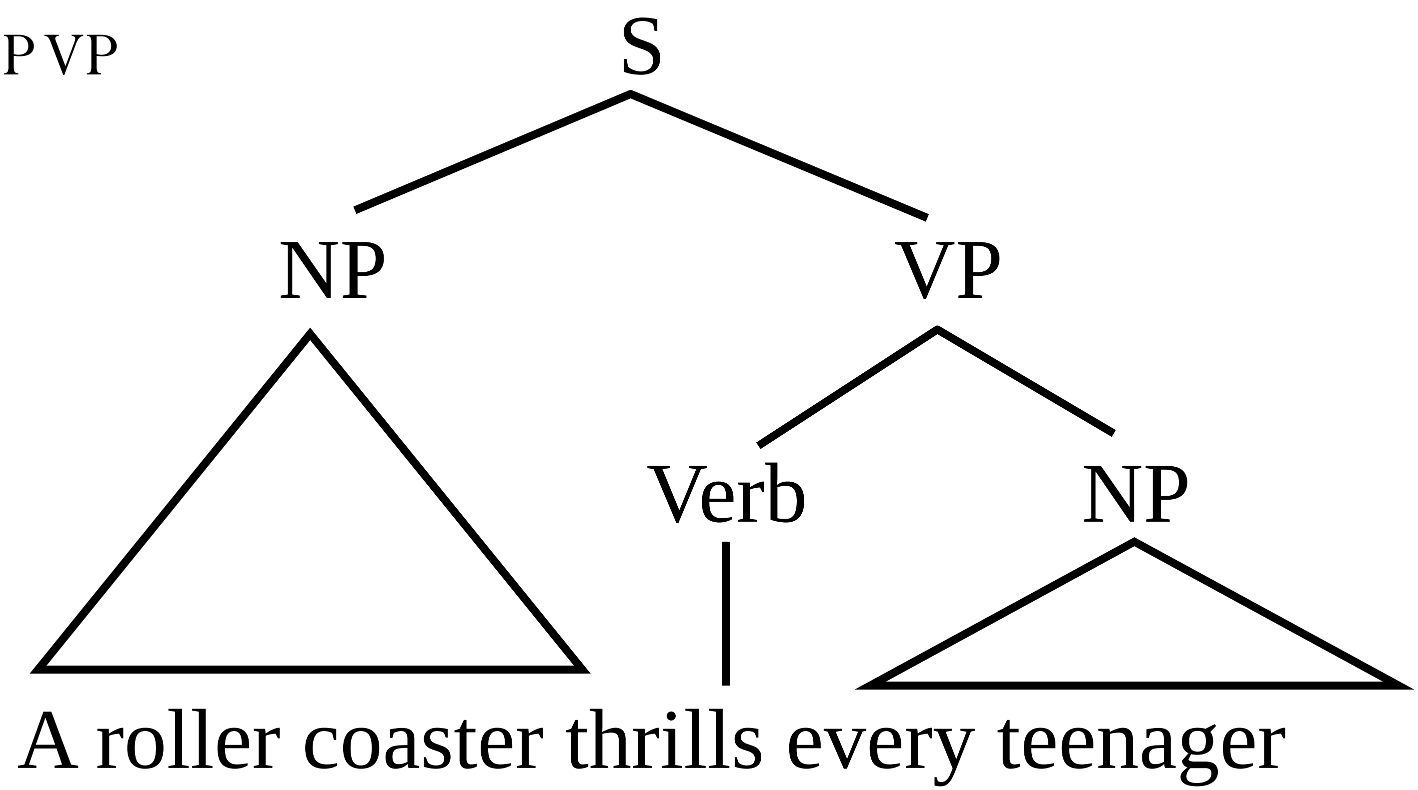
http://www.cs.stonybrook.edu/~cse392

Last class: grammars and parsing in Prolog

Verb $\rightarrow$ thrills

VP $\rightarrow$ Verb NP

S $\rightarrow$ NP VP

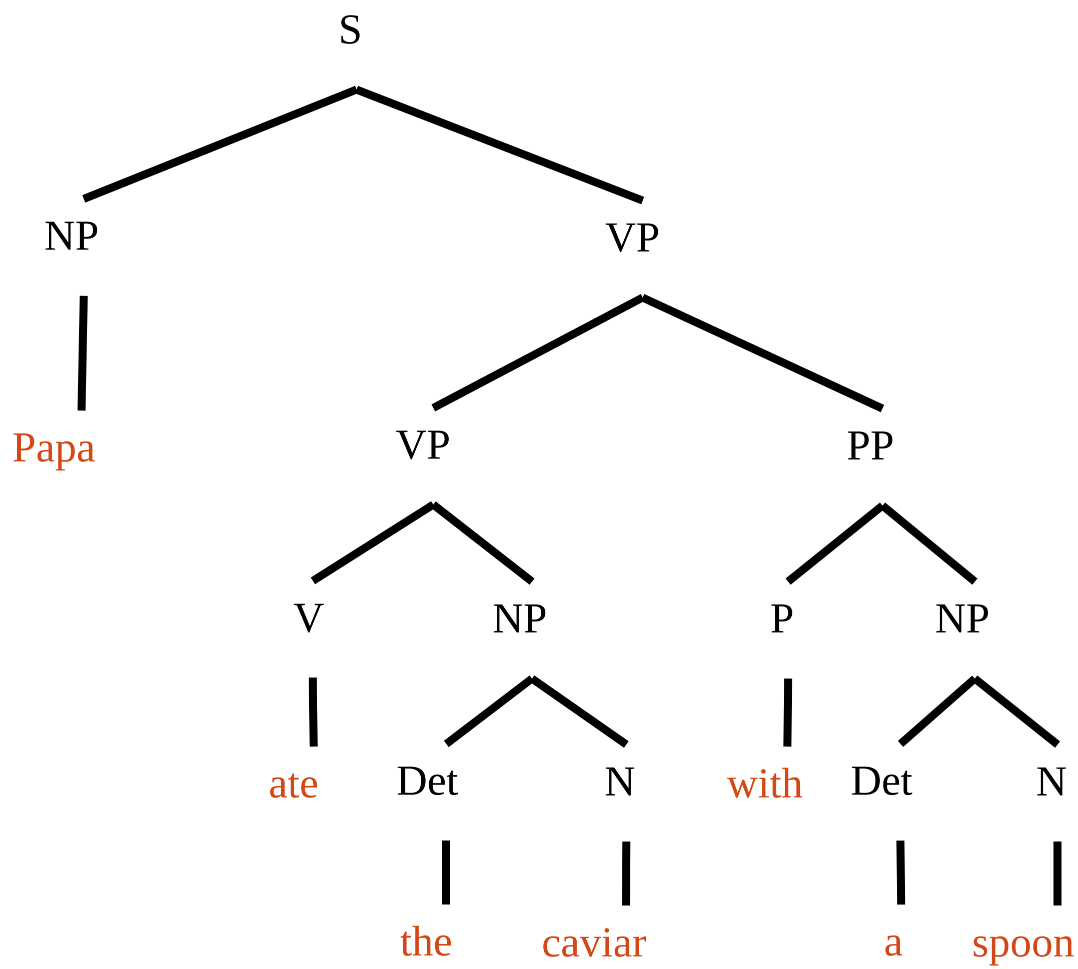


Today: NLP ambiguity

- Example: books: NOUN OR VERB
  - You do not need books for this class.
  - She books her trips early.
- Another example: Thank you for not smoking or playing iPods without earphones.
  - Thank you for not smoking () without earphones ☺
- These cases can be detected as special uses of the same word
- Caveout:
  - If we write too many rules, we may write 'unnatural' grammars – special rules became general rules – it puts a burden too large on the person writing the grammar

Ambiguity in Parsing

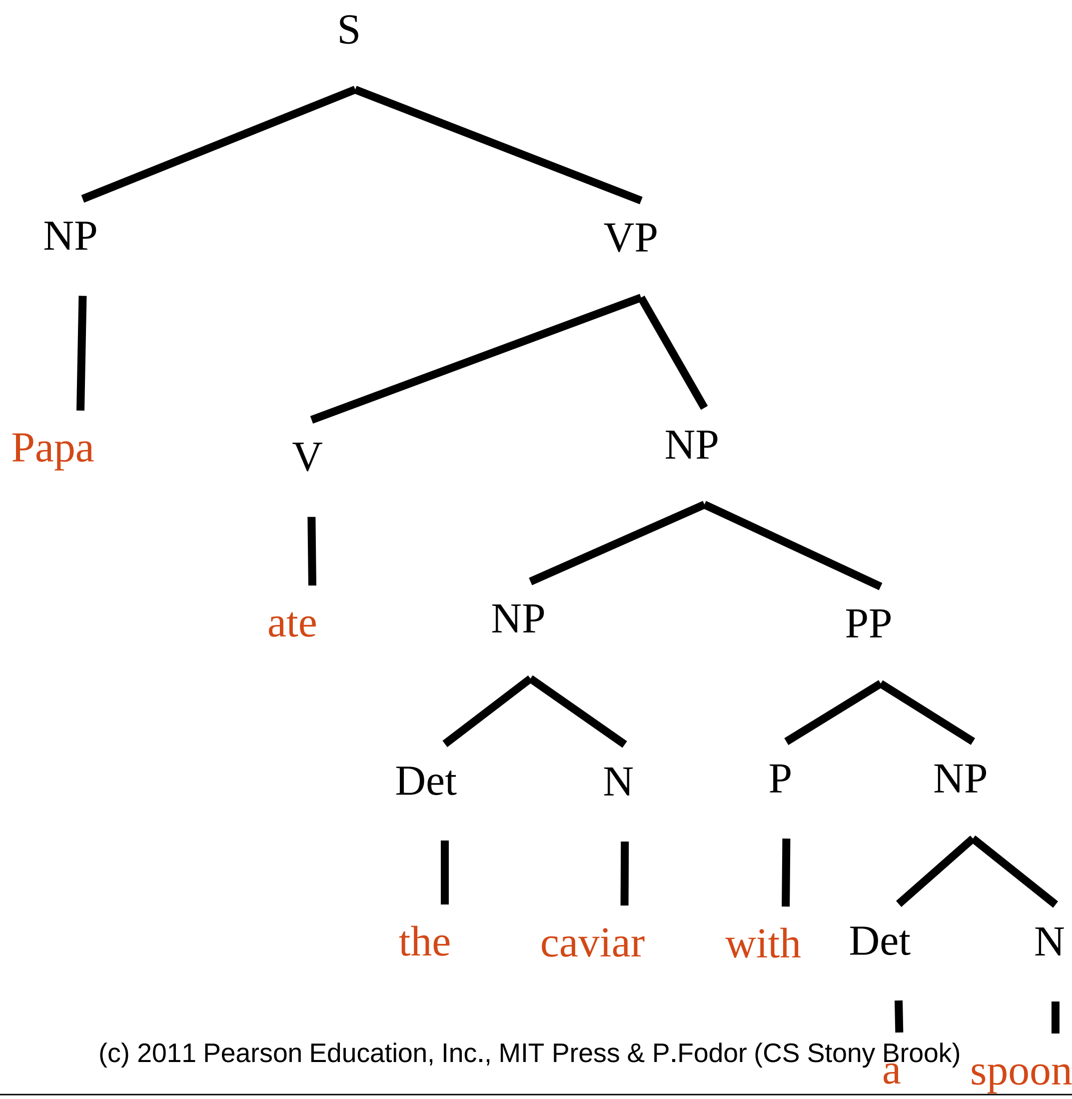
S → NP VP
NP → Det N
NP → NP PP
VP → V NP
VP → VP PP
PP → P NP



NP → Papa
N → caviar
N → spoon
V → spoon
V → ate
P → with
Det → the
Det → a

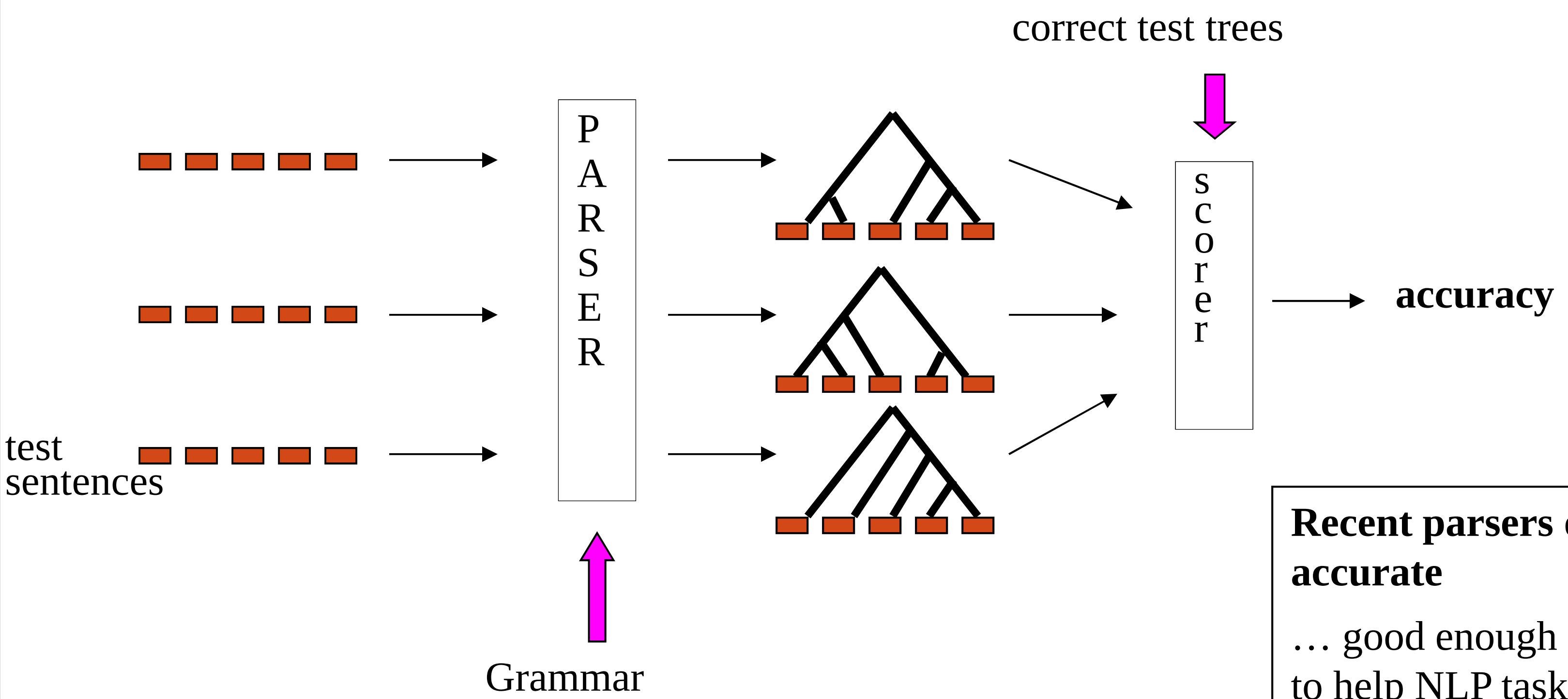
Ambiguity in Parsing

S → NP VP
NP → Det N
NP → NP PP
VP → V NP
VP → VP PP
PP → P NP



NP → Papa
N → caviar
N → spoon
V → spoon
V → ate
P → with
Det → the
Det → a

Scores



Recent parsers quite accurate

… good enough
to help NLP tasks!

Speech processing ambiguity

- Speech processing is a very hard problem (gender, accent, background noise)
- Solution: n-grams
  - Letter or word frequencies: 1-grams: THE, COURSE
    - useful in solving cryptograms
  - If you know the previous letter/word: 2-grams
    - "h" is rare in English (4%; 4 points in Scrabble)
    - but "h" is common after "t" (20%)!!!
  - If you know the previous 2 letters/words: 3-grams
    - "h" is <u>really</u> common after "(space) t?"

N-grams

- What are n-grams good for?
  - Useful for search engines, indexers, etc.
  - Useful for text-to-speech
- How to Build a N-gram?
  - Histogram of letters?
  - Histogram of bigrams?

Probabilities and statistics

- descriptive: mean scores
- confirmatory: statistically significant?
- predictive: what will follow?
- Probability notation p(X | Y):

  p(Paul Revere wins | weather's clear) = 0.9

  - Revere's won 90% of races with clear weather

p(win | clear) = p(win, clear) / p(clear)

syntactic sugar

logical conjunction of predicates

predicate selecting races where weather's clear

Properties of p (axioms)

$p(\emptyset) = 0$ $p(\text{all outcomes}) = 1$

$p(X) \leq p(Y)$ for any $X \subseteq Y$

$p(X \cup Y) = p(X) + p(Y)$ provided $X \cap Y = \emptyset$

- example: p(win & clear) + p(win & ¬clear) = p(win)

Properties and Conjunction

- what happens as we add conjuncts to left of bar ?

  p(Paul Revere wins, Valentine places, Epitaph shows | weather's clear)

  - probability can only decrease

- what happens as we add conjuncts to right of bar ?

  p(Paul Revere wins | weather's clear, ground is dry, jockey getting over sprain)

  - probability can increase or decrease
  - Simplifying Right Side (Backing Off) - reasonable estimate

  p(Paul Revere wins | weather's clear, ~~ground is dry, jockey getting over sprain~~)

Factoring Left Side: The Chain Rule

p(Revere, Valentine, Epitaph | weather's clear)

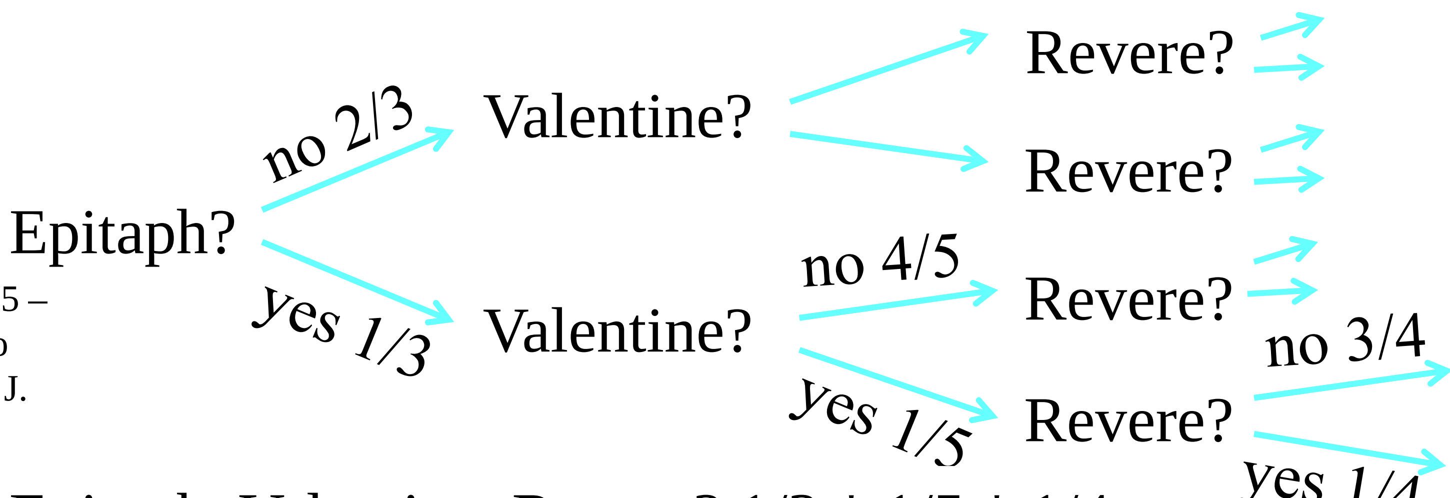
= p(Revere | Valentine, Epitaph, weather's clear)

* p(Valentine | Epitaph, weather's clear)

* p(Epitaph | weather's clear)

True because numerators cancel against denominators

Makes perfect sense when read from bottom to top



600.465 – Intro to NLP – J. Eisner

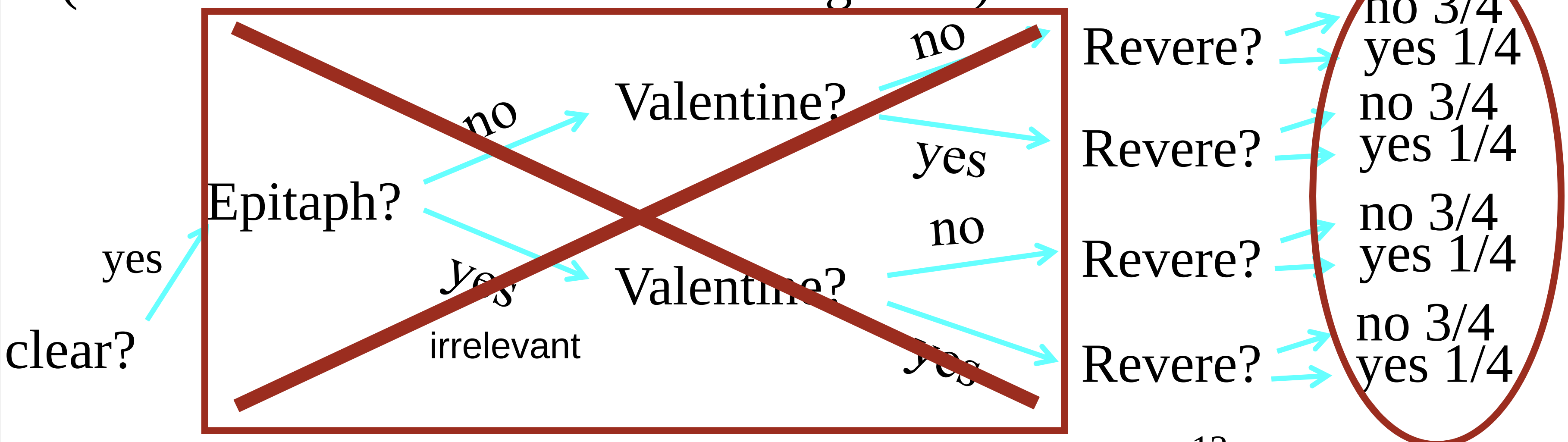
Epitaph, Valentine, Revere? 1/3 * 1/5 * 1/4

Factoring Left Side: The Chain Rule

p(Rev~~ere | Valentine, Epit~~aph, weather's clear)

conditional independence lets us use backed-off data from all four of these cases to estimate their <u>shared</u> probabilities

If this prob is unchanged by backoff, we say Revere was CONDITIONALLY INDEPENDENT of Valentine and Epitaph (conditioned on the weather's being clear).



Bayes' Theorem

- p(A | B) = p(B | A) * p(A) / p(B)

- Easy to check by removing syntactic sugar
- Use 1: Converts p(B | A) to p(A | B)
- Use 2: Updates p(A) to p(A | B)

Probabilistic Algorithms

- Example: The Viterbi algorithm computes the probability of a sequence of observed events and the most likely sequence of hidden states (the Viterbi path) that result in the sequence of observed events.

http://www.cs.stonybrook.edu/~pfodor/old_page/viterbi/viterbi.P

forward_viterbi(+Observations, +States, +Start_probabilities, +Transition_probabilities, +Emission_probabilities, -Prob, -Viterbi_path, -Viterbi_prob)

forward_viterbi(['walk', 'shop', 'clean'], ['Ranny', 'Sunny'], [0.6, 0.4], [[0.7, 0.3],[0.4,0.6]], [[0.1, 0.4, 0.5], [0.6, 0.3, 0.1]], Prob, Viterbi_path, Viterbi_prob) will return:

Prob = 0.03361, Viterbi_path = [Sunny, Rainy, Rainy, Rainy], Viterbi_prob=0.0094

Viterbi Algorithm

- A dynamic programming algorithm
- Input: a first-order hidden Markov model (HMM)
  - states Y
  - initial probabilities π_i of being in state i
  - transition probabilities $a_{i,j}$ of transitioning from state i to state j
  - observations $x_0 , \ldots , x_T$
- Output: The state sequence $y_0 , \ldots , y_T$ most likely to have produced the observations
  - $V_{t,k}$ is the probability of the most probable state sequence responsible for the first $t + 1$ observations
  - $V_{0,k} = P(x_0 \mid k)\ \pi_\mathrm{i}$
  - $V_{T,k} = P(x_t \mid k)\ \max_{y\ \in \mathrm{Y}} (\mathrm{a}_{y,k} V_{t-1,y})$

 Wikipedia

Viterbi Algorithm

- Alice and Bob live far apart from each other
- Bob does three activities: walks in the park, shops, and cleans his apartment
- Alice has no definite information about the weather where Bob lives
- Alice tries to guess what the weather is based on what Bob does
  - The weather operates as a discrete Markov chain
    - There are two (hidden to Alice) states "Rainy" and "Sunny“
    - start_probability = {'Rainy': 0.6, 'Sunny': 0.4}

 Wikipedia

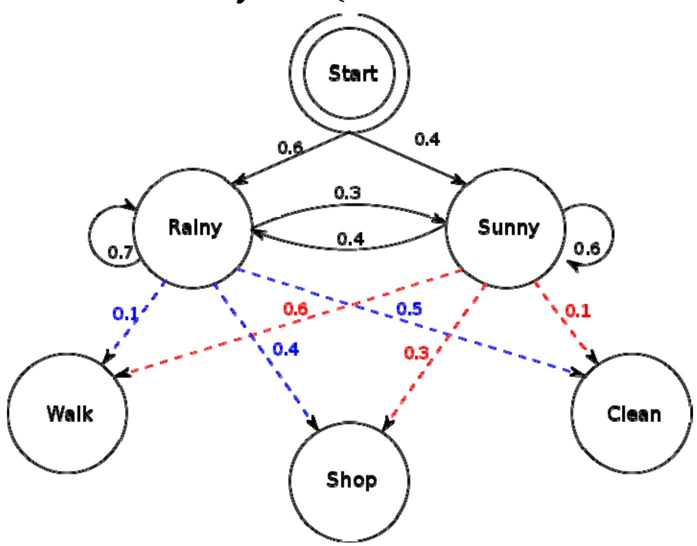
Viterbi Algorithm

- The transition_probability represents the change of the weather

transition_probability = { 'Rainy' : {'Rainy': 0.7, 'Sunny': 0.3},
'Sunny' : {'Rainy': 0.4, 'Sunny': 0.6}}

- The emission_probability represents how likely Bob is to perform a certain activity on each day:

emission_probability = { 'Rainy' : {'walk': 0.1, 'shop': 0.4, 'clean': 0.5},
'Sunny' : {'walk': 0.6, 'shop': 0.3, 'clean': 0.1}, }

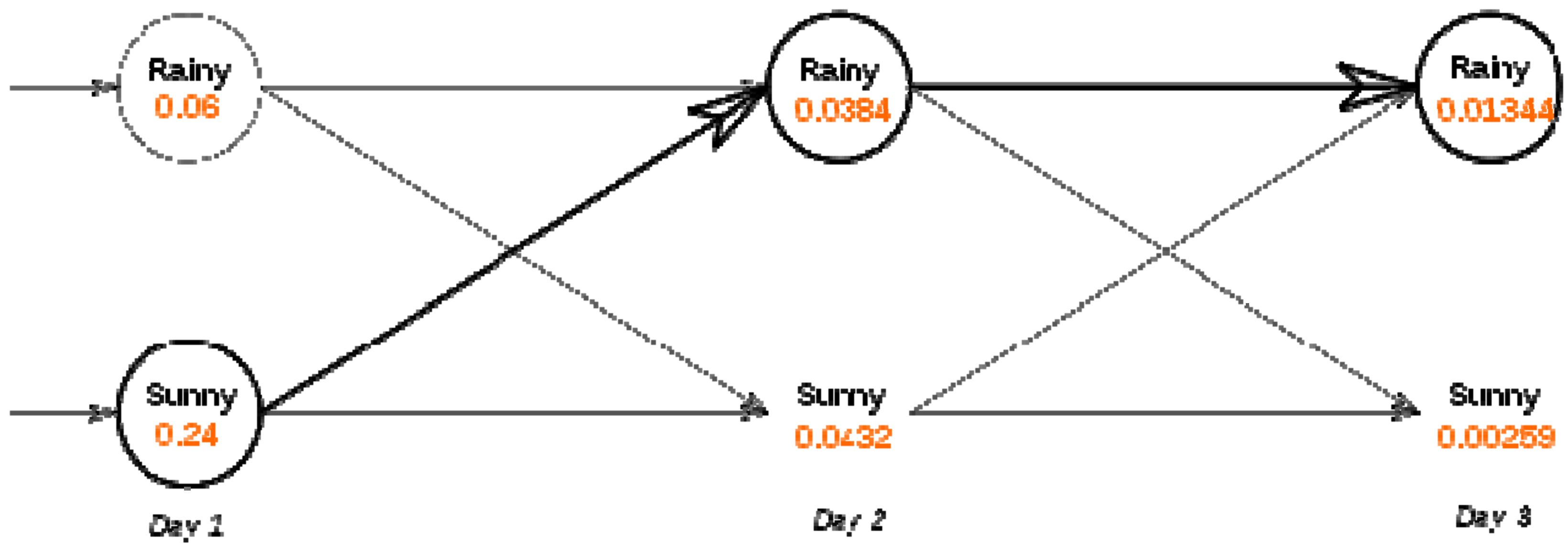


 Wikipedia

Viterbi Algorithm

- Alice talks to Bob and discovers the history of his activities:
  - on the first day he went for a walk
  - on the second day he went shopping
  - on the third day he cleaned his apartment

  ['walk', 'shop', 'clean']
- What is the most likely sequence of rainy/sunny days that would explain these observations?


Rainy
0.06
Sunny
0.24
Day 1
Rainy
0.0384
Sunny
0.0432
Day 2
Rainy
0.01344
Sunny
0.00259
Day 3

 Wikipedia