Grammars and introduction to machine learning

Computers Playing Jeopardy! Course

Stony Brook University

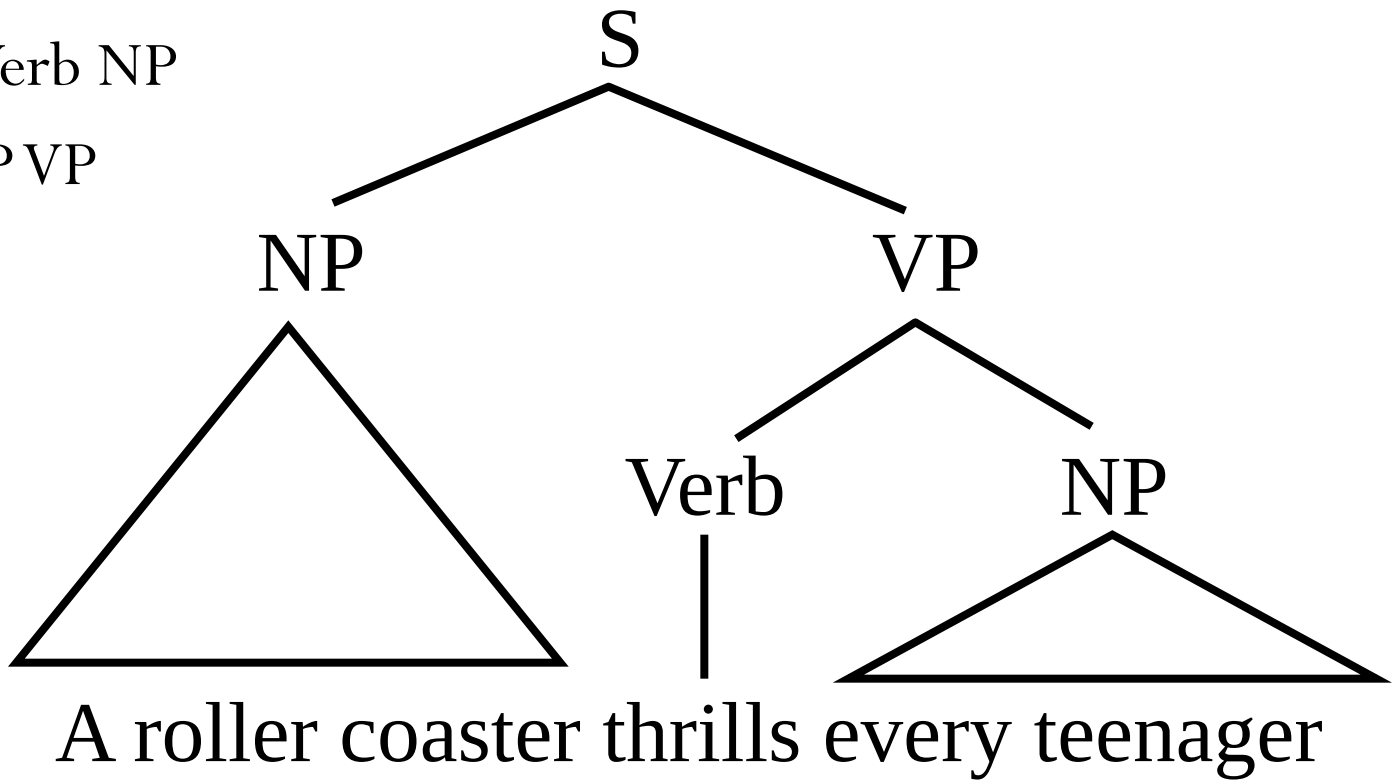
Last class: grammars and parsing in Prolog

Noun -> roller

Verb → thrills

VP→ Verb NP

S → NP VP



(c) Paul Fodor (CS Stony Brook)

Today: NLP ambiguity

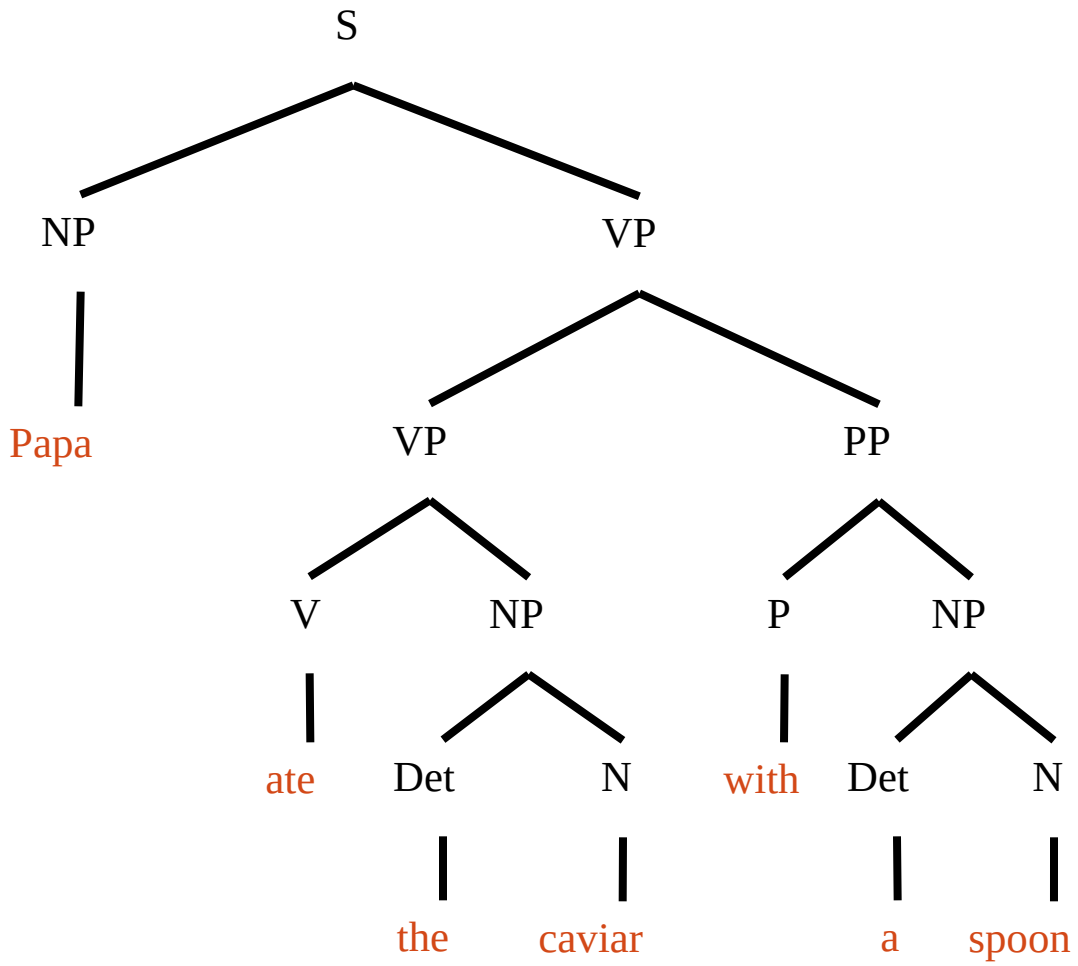
- Example: books: NOUN OR VERB
  - You do not need books for this class.
  - She books her trips early.
- Another example: Thank you for not smoking or playing iPods without earphones.
  - Thank you for not smoking () without earphones ☺
- These cases can be detected as special uses of the same word
  - 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

(c) Paul Fodor (CS Stony Brook)

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

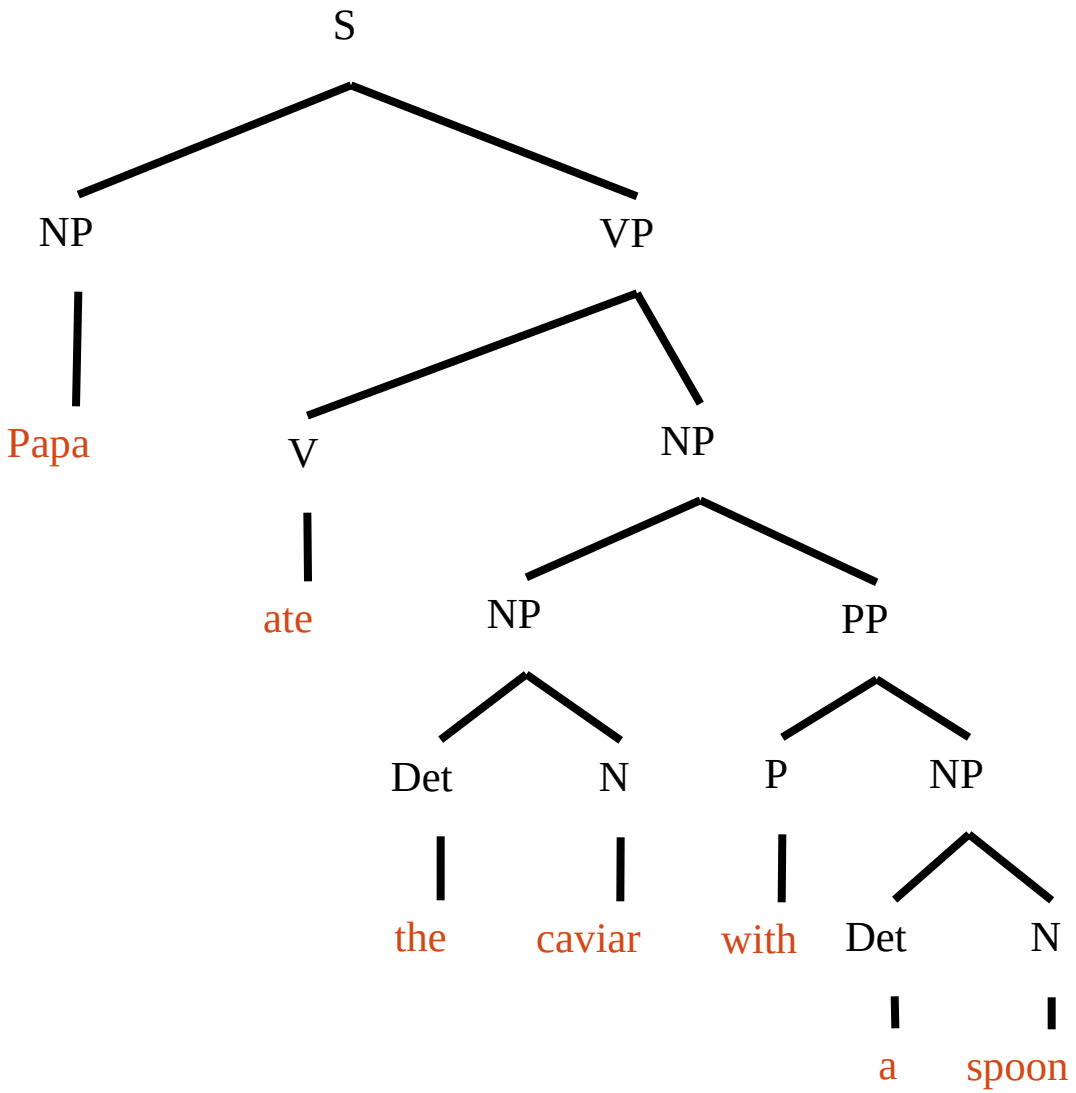


(c) Paul Fodor (CS Stony Brook)

Ambiguity in Parsing

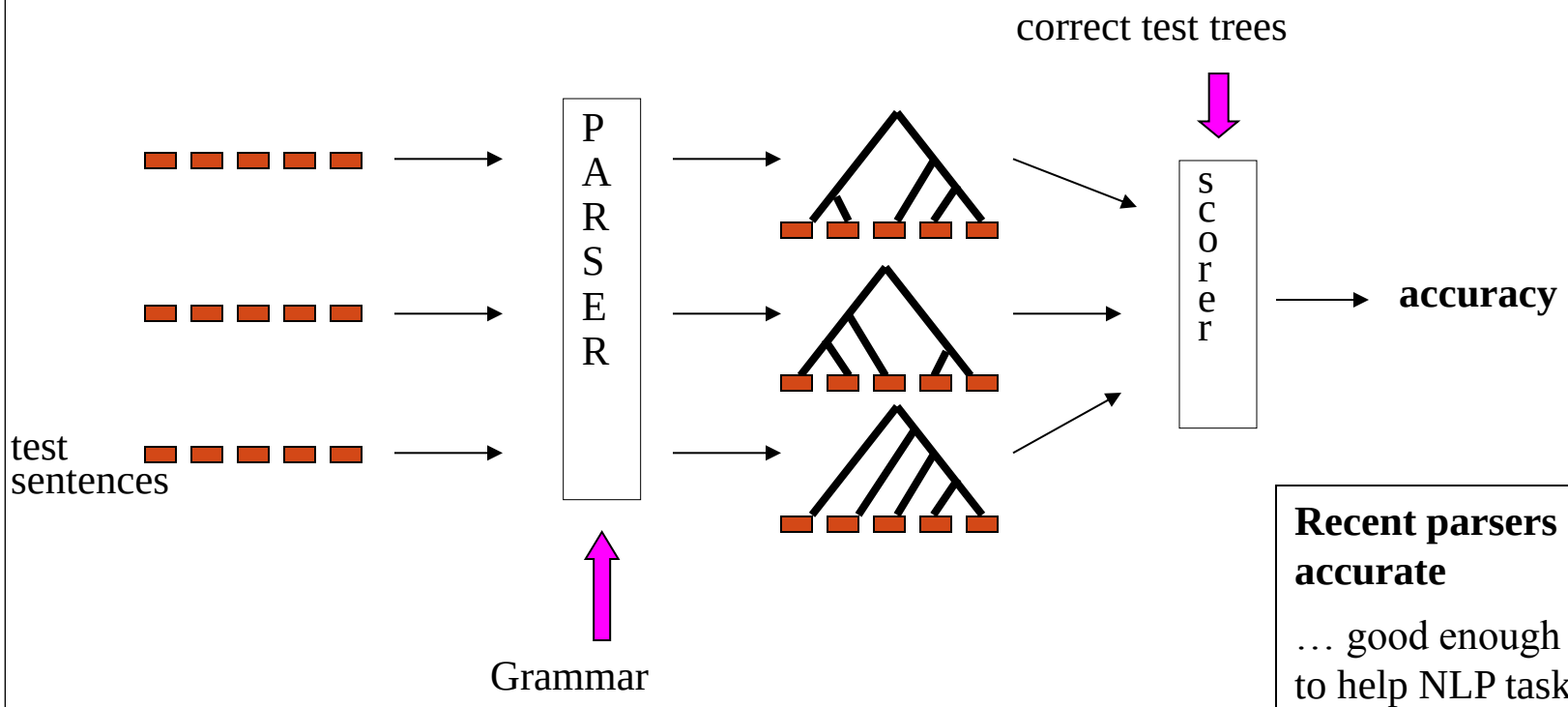
S → NP VP
NP → Det N
NP → NP PP
VP → V NP
VP → VP PP
PP → P NP

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



(c) Paul Fodor (CS Stony Brook)

Scores



Recent parsers quite accurate

… good enough
to help NLP tasks!

(c) Paul Fodor (CS Stony Brook)

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 really common after "(space) t"

(c) Paul Fodor (CS Stony Brook)

N-grams

- An n-gram is a contiguous sequence of *n* items from a given sequence
  - the items can be: letters, words, phonemes, syllables, etc.
  - Examples:
    - Consider the paragraph: "Suppose you should be walking down Broadway after dinner, with ten minutes allotted to the consummation of your cigar while you are choosing between a diverting tragedy and something serious in the way of vaudeville. Suddenly a hand is laid upon your arm." (from the The green door, The Four Million novel, by O. Henry)
    - unigrams (n-grams of size 1): "Suppose", "you", "should", "be", "walking",…
    - bigrams (n-grams of size 2): "Suppose you", "you should", "should be", "be walking",…
    - trigrams (*n*-grams of size 3): "Suppose you should", "you should be", "should be walking",…
    - four-grams: "Suppose you should be", "you should be walking",…

(c) Paul Fodor (CS Stony Brook)

N-grams

- An n-gram model is a type of *probabilistic language model* for predicting the next item in such a sequence
  - Example: from a novel, we can extract all bigrams (sequences of 2- words) with their probabilities of showing up:
    - probability("you can")=

      number of "you can"s in text/total number of bigrams = 0.002
    - probability("you should")=

      number of "you should"s in text/total number of bigrams = 0.001
    - etc.
  - Consider that the *current* word is "you" and we want to predict what is the next word:
    - with probability 0.002%, the next word is "can"
    - with probability 0.001%, the next word is "should"
- Advantages: easy training, simple/intuitive probabilistic model.

(c) Paul Fodor (CS Stony Brook)

N-grams in NLP

- N-gram models are widely used in statistical natural language processing (NLP):
  - in speech recognition, phonemes and sequences of phonemes are modeled using a n-gram distribution:
    - due to multitude of accents, various voice pitches, gender, etc., speech recognition is very error prone, so information about the predicted next phonemes is very useful to detect the exact transcription
  - in parsing text, words are modeled such that each n-gram is composed of n words
    - there is an infinite number of ways to express ideas in natural language. However, there are common and standardized ways used by humans to convert information. A probabilistic model is useful to the parser to construct a data structure that best fits the grammatical rules.
  - in language identification, sequences of characters are modeled for different languages (e.g., in English, "t" and "w" are followed by an "h" 90% of the time).

(c) Paul Fodor (CS Stony Brook)

N-grams in NLP

- Other applications:
  - plagiarism detection,
  - find candidates for the correct spelling of a misspelled word,
  - optical character recognition (OCR),
  - machine translation,
  - correction codes (correct words that were garbled during transmission)
- Modern statistical models are typically made up of two parts:
  - a prior distribution describing **the inherent likelihood of a possible result** and
  - a **likelihood function** used to assess **the compatibility of a possible result** with observed data.

(c) Paul Fodor (CS Stony Brook)

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

(c) Paul Fodor (CS Stony Brook)

Properties of p (axioms)

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

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

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

- example: $p(\text{win} \;\&\; \text{clear}) + p(\text{win} \;\&\; \neg\text{clear}) = p(\text{win})$

(c) Paul Fodor (CS Stony Brook)

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~~)

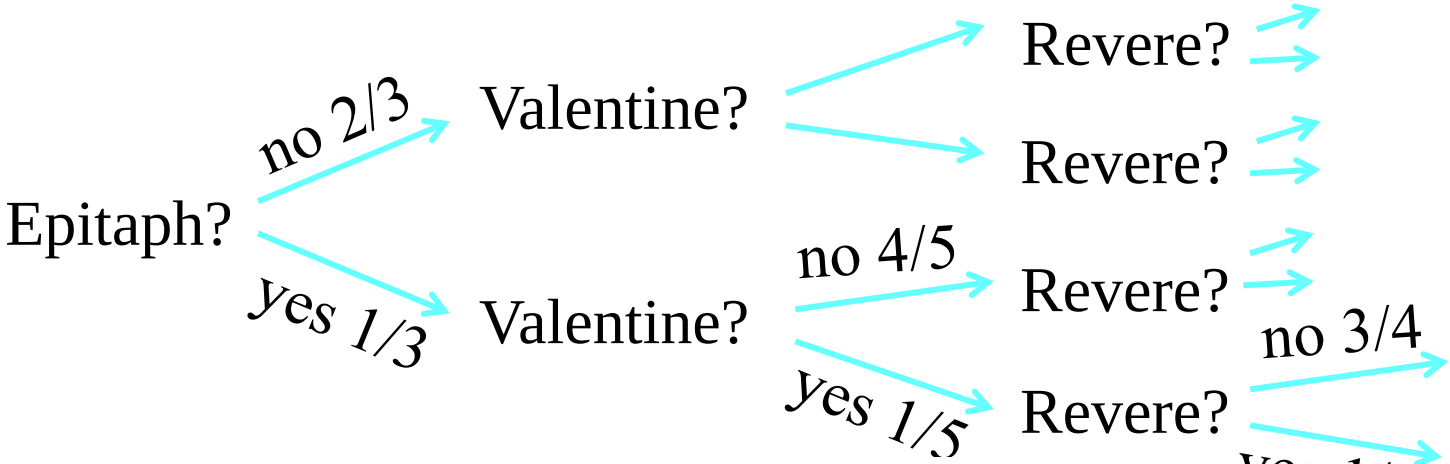
(c) Paul Fodor (CS Stony Brook)

Factoring Left Side: The Chain Rule

$$p(\text{Revere}, \text{Valentine}, \text{Epitaph} \mid \text{weather's clear})$$
$$= p(\text{Revere} \mid \text{Valentine}, \text{Epitaph}, \text{weather's clear})$$
$$* p(\text{Valentine} \mid \text{Epitaph}, \text{weather's clear})$$
$$* p(\text{Epitaph} \mid \text{weather's clear})$$

True because numerators cancel against denominators

Makes perfect sense when read from bottom to top



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

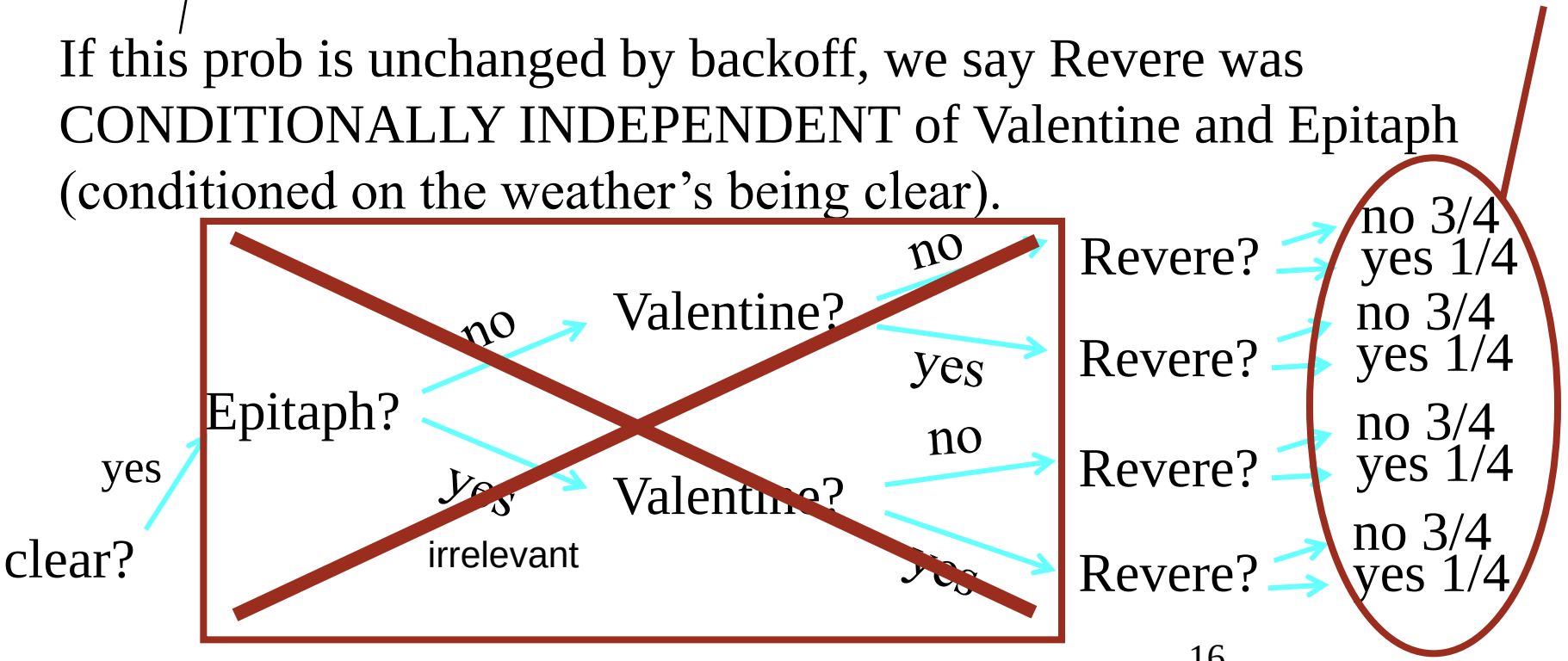
(c) Paul Fodor (CS Stony Brook)

Factoring Left Side: The Chain Rule

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

conditional independence lets us use backed-off data from all four of these cases to estimate their shared 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).



(c) Paul Fodor (CS Stony Brook)

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)

(c) Paul Fodor (CS Stony Brook)

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

(c) Paul Fodor (CS Stony Brook)

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}

(c) Paul Fodor (CS Stony Brook)

Wikipedia

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
  - $V_{0,k} = P(x_0 \mid k)\ \pi_i$
  - $V_{T,k} = P(x_t \mid k)\ \max_{y \in Y} (a_{y,k} V_{t-1,y})$

(c) Paul Fodor (CS Stony Brook)

Wikipedia

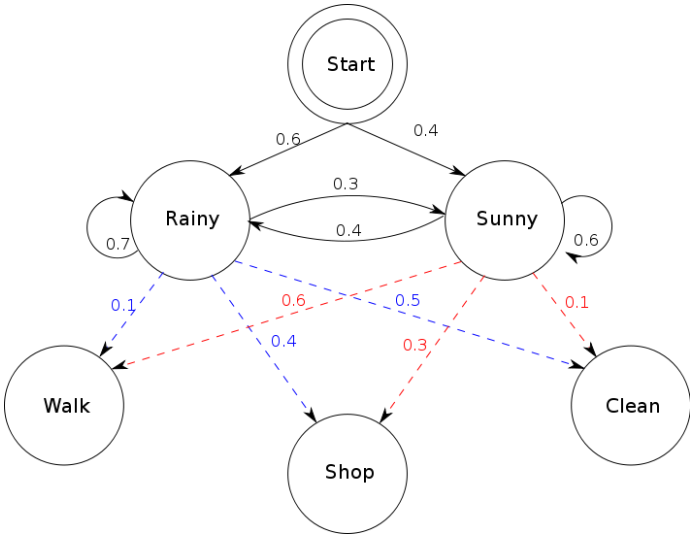
Viterbi Algorithm

- 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}, }



(c) Paul Fodor (CS Stony Brook)

Wikipedia

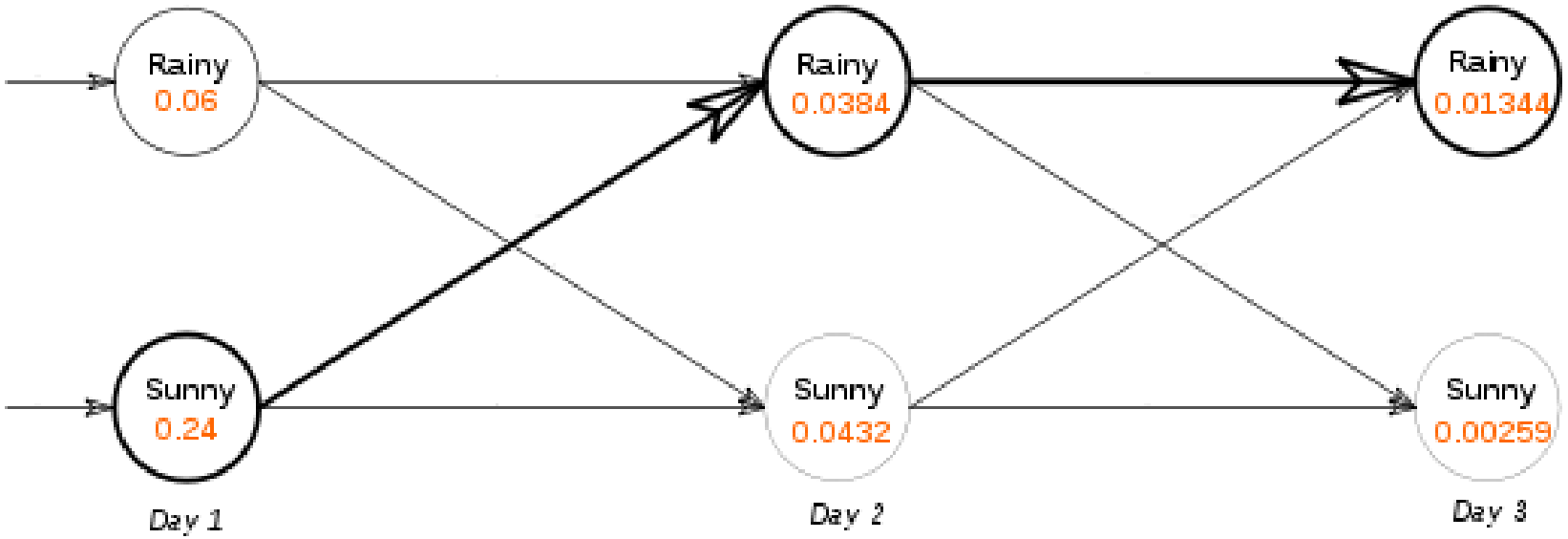
Viterbi Algorithm

- 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?

(c) Paul Fodor (CS Stony Brook)

Wikipedia



(c) Paul Fodor (CS Stony Brook)

Wikipedia