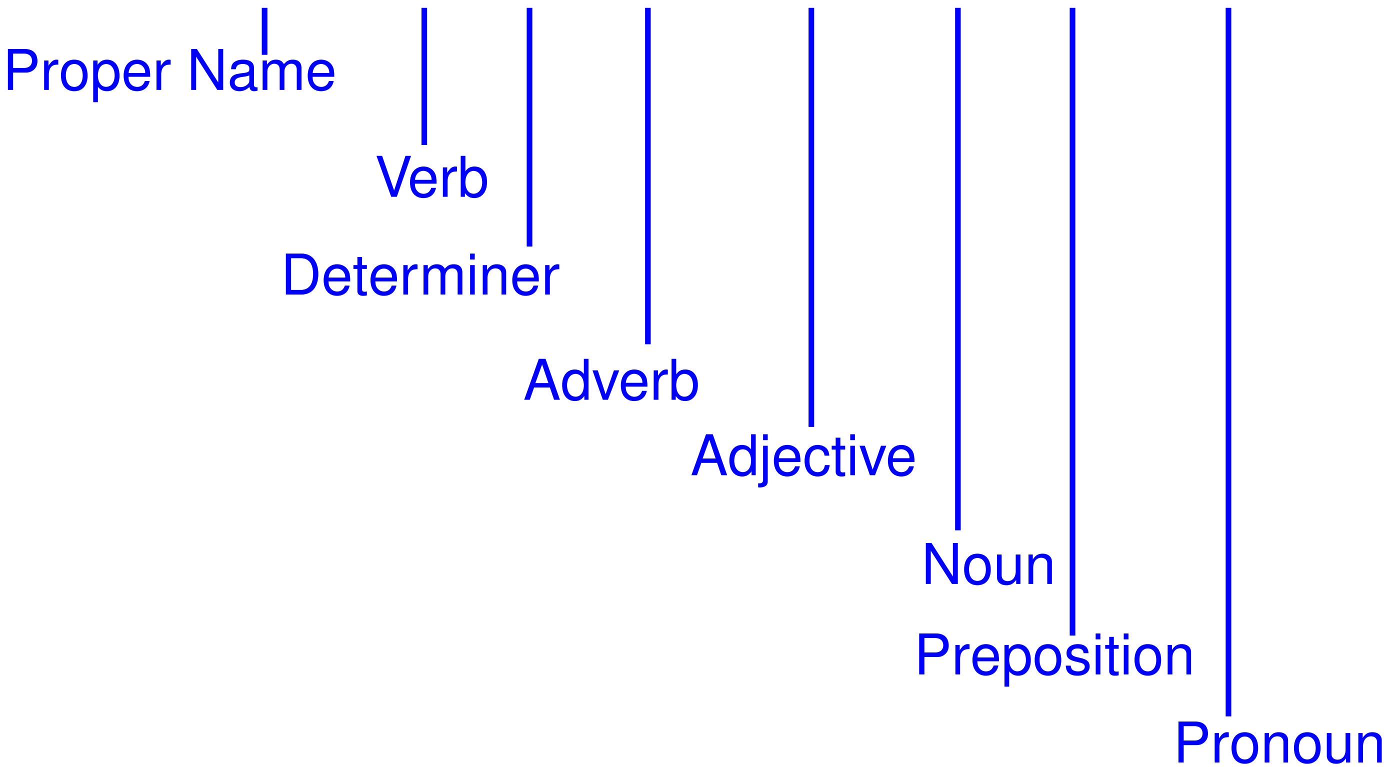
# POS-Tagging

Fabian M. Suchanek

# Def: POS

A Part-of-Speech (also: POS, POS-tag, word class, lexical class, lexical category) is a set of words with the same grammatical role.

"Elvis wrote a really great song by himself"



# Part of Speech

A Part-of-Speech (also: POS, POS-tag, word class, lexical class, lexical category) is a set of words with the same grammatical role.

- Proper nouns (PN): Wile, Elvis, Obama...
- Nouns (N): desert, sand, ...
- Adjectives (ADJ): fast, funny, ...
- Adverbs (ADV): fast, well, ...
- Verbs (V): run, jump, dance, ...
- Pronouns (PRON): he, she, it, this, ...
  (what can replace a noun)
- Determiners (DET): the, a, these, your,...
  (what goes before a noun)
- Prepositions (PREP): in, with, on, ...
  (what goes before determiner + noun)
- Subordinators (SUB): who, whose, that, which, because...
  (what introduces a sub-sentence)

# Def: POS-Tagging

POS-Tagging is the process of assigning to each word in a sentence its POS.

| Wile | tries | a | new | machine. |
|---|---|---|---|---|
| PN | V | DET | ADJ | N |

postagging>65

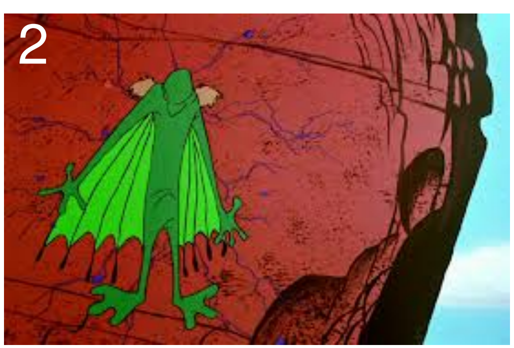
task>31

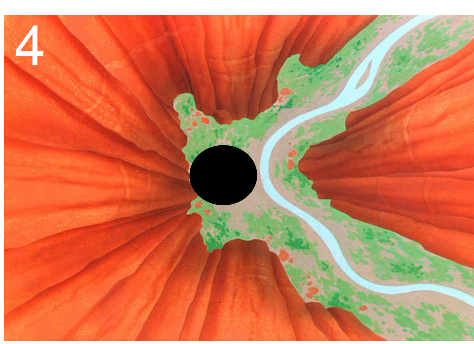
# Task: POS-Tagging

POS-Tag the following sentence:

The coyote falls into a canyon.

- N, V, PN, ADJ, ADV, PRON
- Determiners (DET): the, a, these, your,...
  (what goes before a noun)
- Prepositions (PREP): in, with, on, ...
  (what goes before determiner + noun)
- Subordinators (SUB): who, whose, that,...
  (what introduces a sub-sentence)





# Probabilistic POS-Tagging

Probabilistic POS tagging is an algorithm for automatic POS tagging that works by introducing random variables for words and for POS-tags:

| | (visible) | | (hidden) | | |
|---|---|---|---|---|---|
| World | W1 | W2 | T1 | T2 | Probability |
| $\omega_1$ | Elvis | sings | PN | Verb | $P(\omega_1) = 0.2$ |
| $\omega_2$ | Elvis | sings | Adj | Verb | $P(\omega_2) = 0.1$ |
| $\omega_3$ | Elvis | runs | Prep | PN | $P(\omega_3) = 0.1$ |
| | ... | | ... | | |

# Probabilistic POS-Tagging

Given a sentence $w_1, ..., w_n$

we want to find $argmax_{t_1,...,t_n} P(w_1, ..., w_n, t_1, ..., t_n)$.

| | (visible) | | (hidden) | | |
|---|---|---|---|---|---|
| World | W1 | W2 | T1 | T2 | Probability |
| $\omega_1$ | Elvis | sings | PN | Verb | $P(\omega_1) = 0.2$ |
| $\omega_2$ | Elvis | sings | Adj | Verb | $P(\omega_2) = 0.1$ |
| $\omega_3$ | Elvis | runs | Prep | PN | $P(\omega_3) = 0.1$ |
| | ... | | ... | | |

# Markov Assumption 1

Every tag depends just on its predecessor

$$P(T_i|T_1, ..., T_{i-1}) = P(T_i|T_{i-1})$$

# Markov Assumption 1

Every tag depends just on its predecessor

$$P(T_i|T_1, ..., T_{i-1}) = P(T_i|T_{i-1})$$

The probability that PN, V, D is followed by a noun is the same as the probability that D is followed by a noun:

$$P(N|PN, V, D) = P(N|D)$$

# Markov Assumption 1

Every tag depends just on its predecessor

$$P(T_i|T_1, ..., T_{i-1}) = P(T_i|T_{i-1})$$

The probability that PN, V, D is followed by a noun is the same as the probability that D is followed by a noun:

$$P(N|PN, V, D) = P(N|D)$$

| | | | |
|---|---|---|---|
| Elvis | sings | a | song |
| PN | Verb | Det | ? |

# Markov Assumption 1

Every tag depends just on its predecessor

$$P(T_i|T_1, ..., T_{i-1}) = P(T_i|T_{i-1})$$

The probability that PN, V, D is followed by a noun is the same as the probability that D is followed by a noun:

$$P(N|PN, V, D) = P(N|D)$$

| | | | |
|---|---|---|---|
| Elvis | sings | a | song |
| PN | Verb | Det | ? |

# Markov Assumption 2

Every word depends just on its tag:

$$P(W_i | W_1, ..., W_{i-1}, T_1, ..., T_i) = P(W_i | T_i)$$

# Markov Assumption 2

Every word depends just on its tag:

$$P(W_i|W_1, ..., W_{i-1}, T_1, ..., T_i) = P(W_i|T_i)$$

The probability that the 4th word is “song” depends just on the tag of that word:

$$P(song|Elvis, sings, a, PN, V, D, N) = P(song|N)$$

# Markov Assumption 2

Every word depends just on its tag:

$$P(W_i|W_1, ..., W_{i-1}, T_1, ..., T_i) = P(W_i|T_i)$$

The probability that the 4th word is "song" depends just on the tag of that word:

$$P(song|Elvis, sings, a, PN, V, D, N) = P(song|N)$$

| | | | |
|---|---|---|---|
| Elvis | sings | a | ? |
| PN | Verb | Det | Noun |

# Markov Assumption 2

Every word depends just on its tag:

$$P(W_i|W_1, ..., W_{i-1}, T_1, ..., T_i) = P(W_i|T_i)$$

The probability that the 4th word is “song” depends just on the tag of that word:

$$P(song|Elvis, sings, a, PN, V, D, N) = P(song|N)$$

| | | | |
|---|---|---|---|
| Elvis | sings | a | ? |
| PN | Verb | Det | Noun |

# Homogeneity Assumption 1

The tag probabilities are the same at all positions

$$P(T_i|T_{i-1}) = P(T_k|T_{k-1}) \forall i, k$$

# Homogeneity Assumption 1

The tag probabilities are the same at all positions

$$P(T_i|T_{i-1}) = P(T_k|T_{k-1}) \forall i, k$$

The probability that a Det is followed by a Noun
is the same at position 7 and 2:

$$P(T_7 = Noun|T_6 = Det) = P(T_2 = Noun|T_1 = Det)$$

# Homogeneity Assumption 1

The tag probabilities are the same at all positions

$$P(T_i|T_{i-1}) = P(T_k|T_{k-1}) \forall i, k$$

The probability that a Det is followed by a Noun
is the same at position 7 and 2:

$$P(T_7 = Noun|T_6 = Det) = P(T_2 = Noun|T_1 = Det)$$

Let's denote this probability by

$$P(Noun|Det)$$

"Transition probability"

$$P(s|t) := P(T_i = s|T_{i-1} = t)(for any i)$$

# Homogeneity Assumption 2

The word probabilities are the same at all positions

$$P(W_i|T_i) = P(W_k|T_k) \forall i, k$$

# Homogeneity Assumption 2

The word probabilities are the same at all positions

$$P(W_i|T_i) = P(W_k|T_k) \forall i, k$$

The probability that a PN is "Elvis"
is the same at position 7 and 2:

$$P(W_7 = Elvis|T_7 = PN) = P(W_2 = Elvis|T_2 = PN) = 80\%$$

# Homogeneity Assumption 2

The word probabilities are the same at all positions

$$P(W_i|T_i) = P(W_k|T_k)\forall i,k$$

The probability that a PN is "Elvis"

is the same at position 7 and 2:

$$P(W_7 = Elvis|T_7 = PN) = P(W_2 = Elvis|T_2 = PN) = 80\%$$

Let's denote this probability by

$$P(Elvis|PN)$$

"Emission probability"

$$P(w|t) := P(W_i = w|T_i = t)(for any i)$$

# Def: HMM

A (homogeneous) Hidden Markov Model (also: HMM)

is a sequence of random variables, such that

$$P(w_1, ..., w_n, t_1, ..., t_n) = \prod_i P(w_i|t_i) \times P(t_i|t_{i-1})$$

Words of the sentence

POS-tags

Emission probabilities

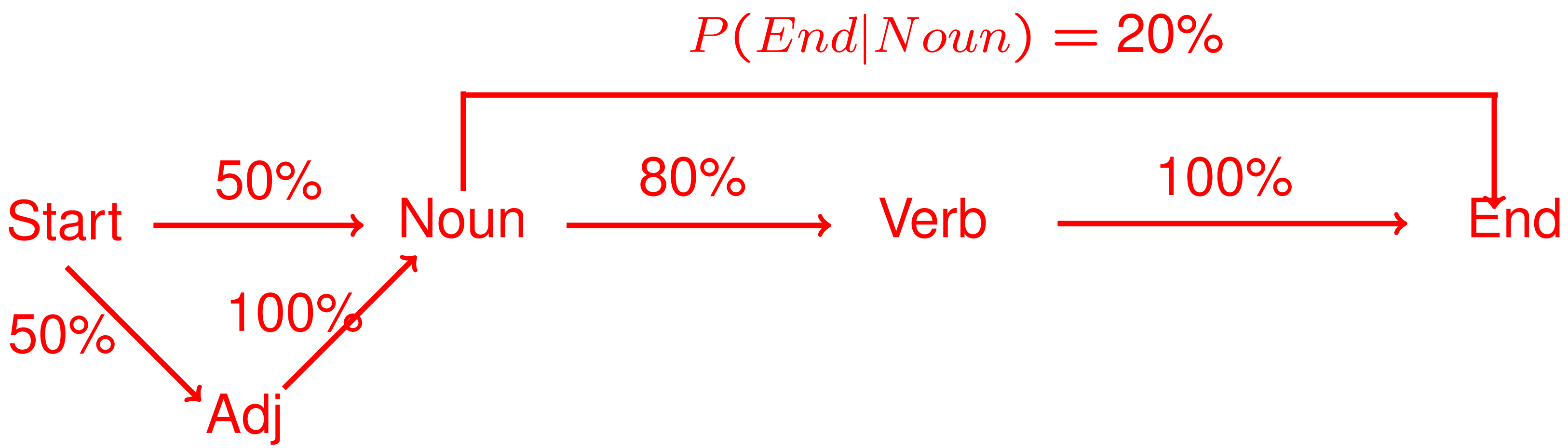
Transition probabilities

... with $t_0 = Start$

# HMMs as graphs

$$P(w_1, ..., w_n, t_1, ..., t_n) = \prod_i P(w_i|t_i) \times P(t_i|t_{i-1})$$

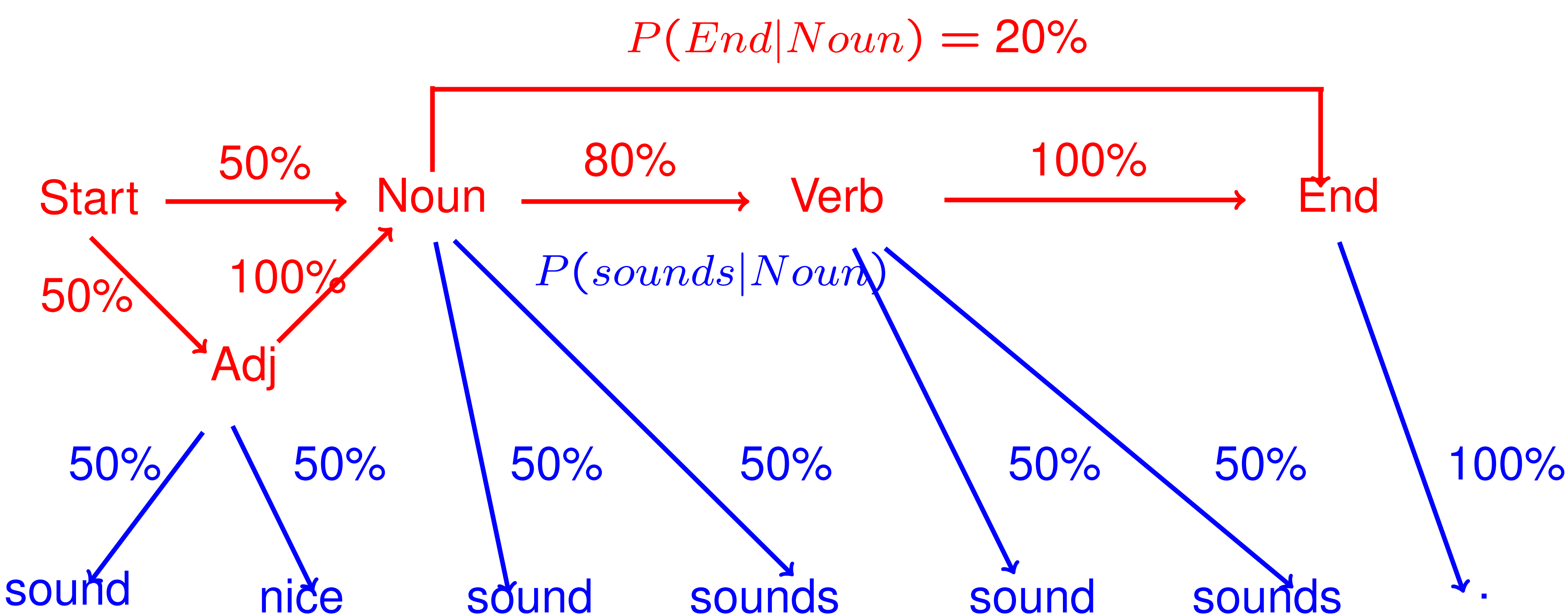
Transition probabilities



# HMMs as graphs

$$P(w_1, ..., w_n, t_1, ..., t_n) = \prod_i P(w_i|t_i) \times P(t_i|t_{i-1})$$

Emission probabilities

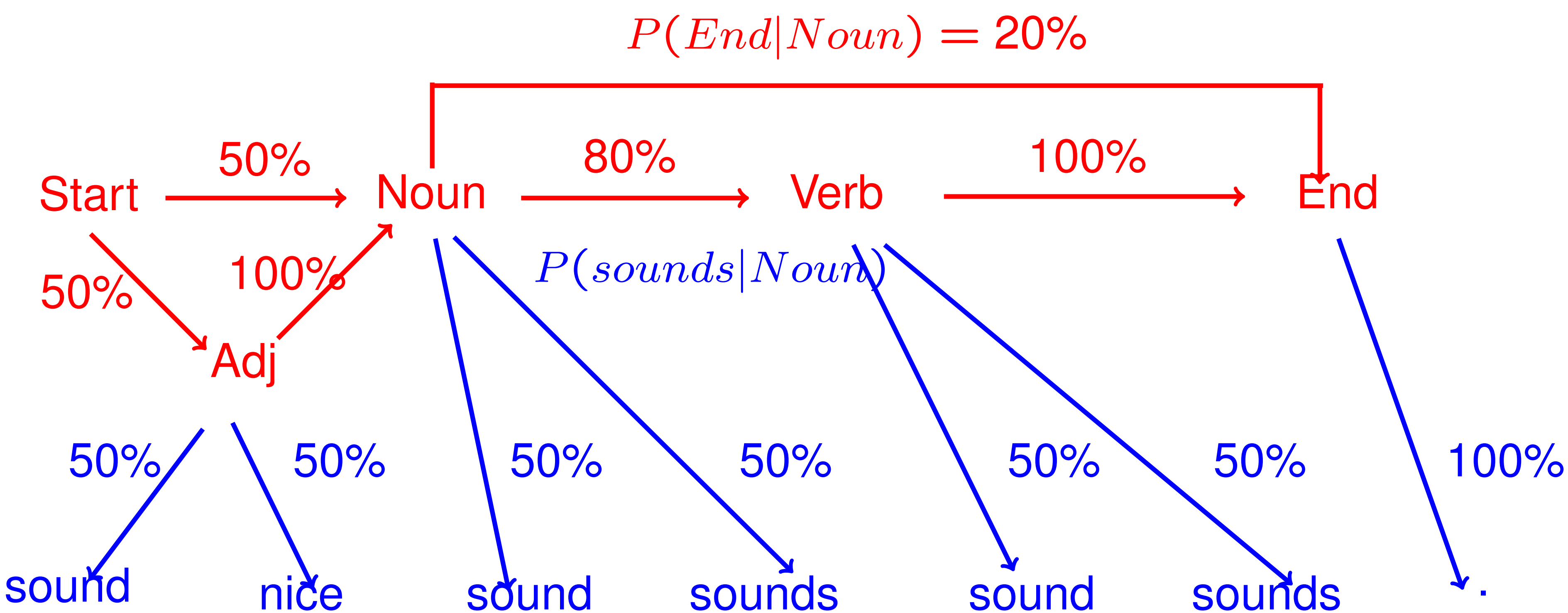


# HMMs as graphs

$$P(w_1, ..., w_n, t_1, ..., t_n) = \prod_i P(w_i|t_i) \times P(t_i|t_{i-1})$$

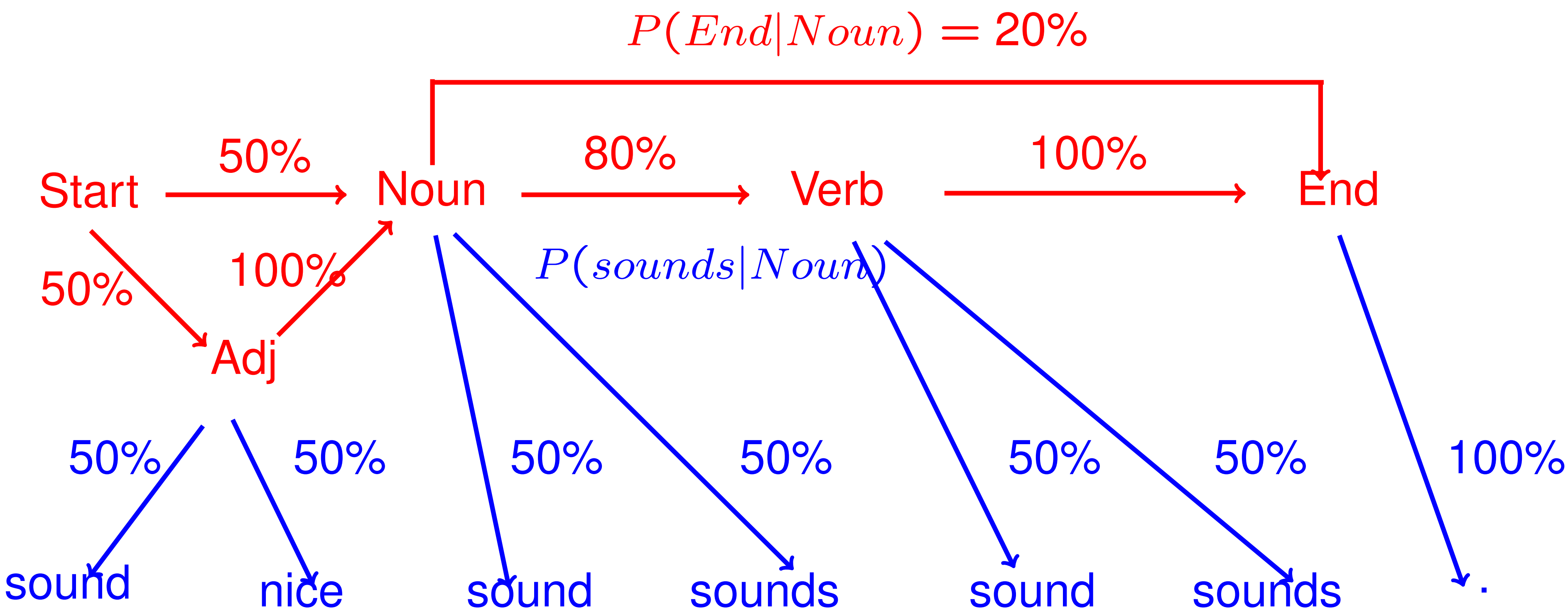
P(nice, sounds, ., Adj, Noun, End) =

50% * 50% * 100% * 50% * 20% * 100% = 2.5%



# HMM Question

What is the most likely tagging for “sound sounds .” ?



# POS tagging with HMMs

What is the most likely tagging for “sound sounds .” ?

Adj + Noun: 50%*50%*100%*50%*20%*100% = 2.5%

Noun + Verb: 50%*50%*80%*50%*100% = 10%

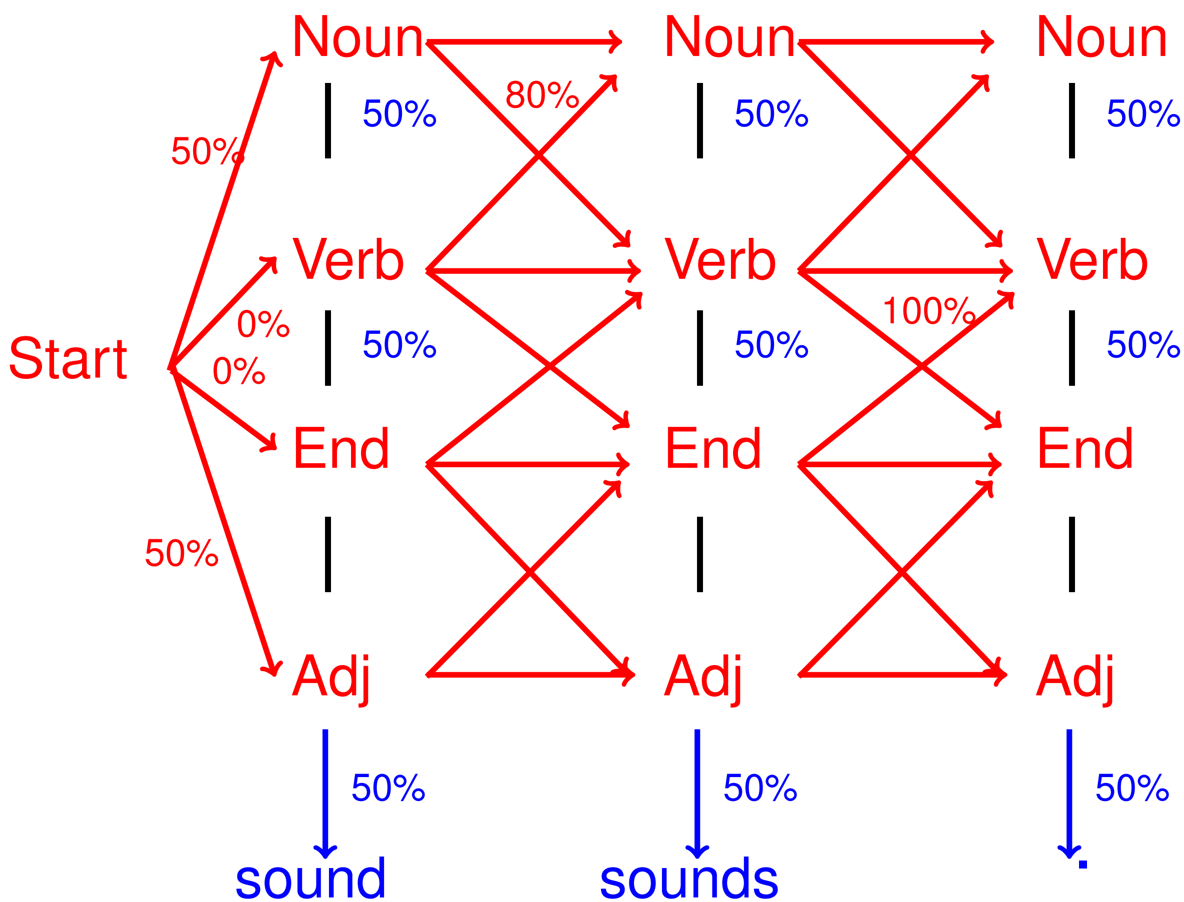
Finding the most likely sequence of tags

that generated a sentence is POS tagging (hooray!).

Viterbi>63

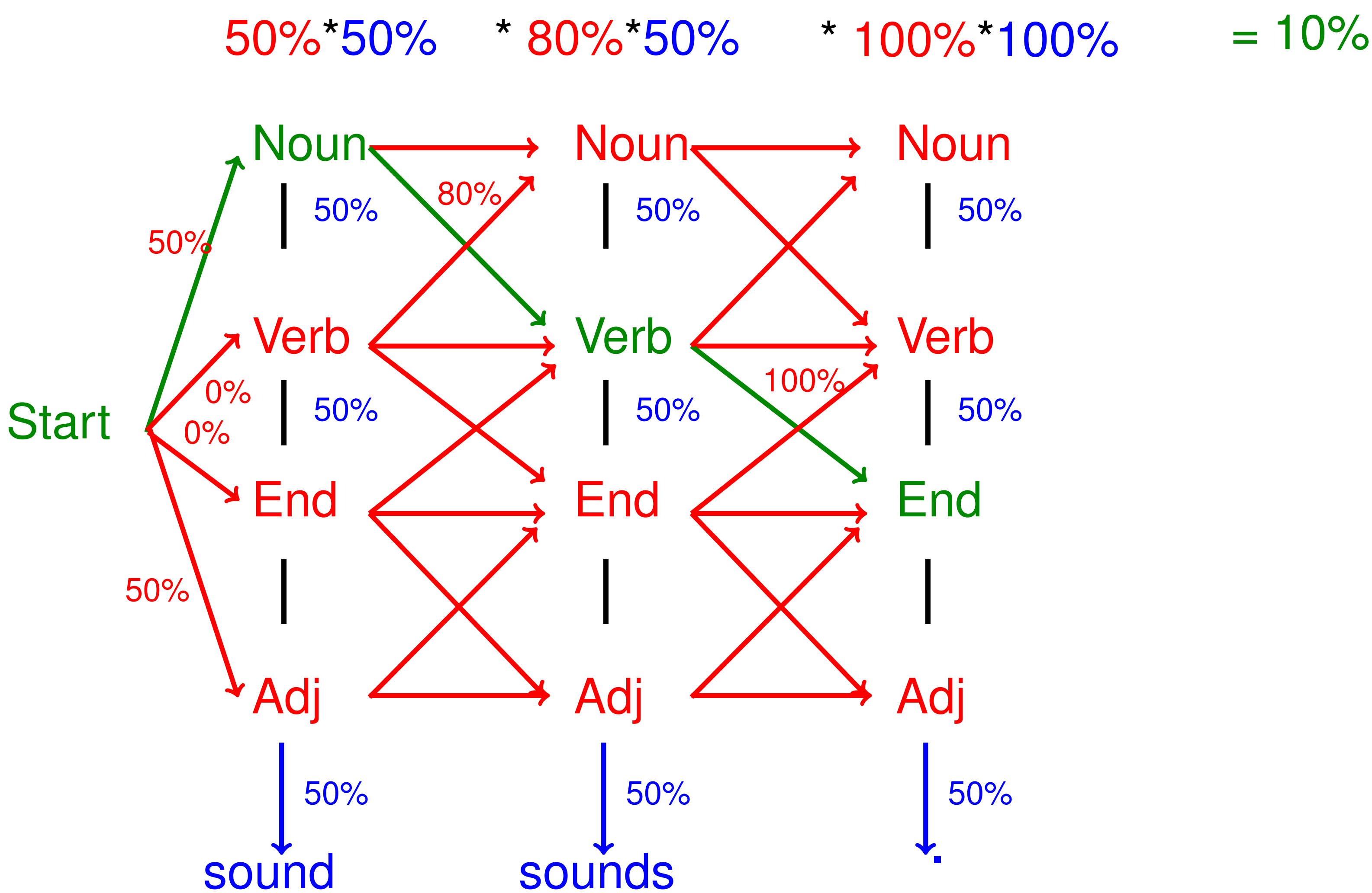
# The HMM as a Trellis

Task: compute the probability of every possible path from left to right, find maximum.

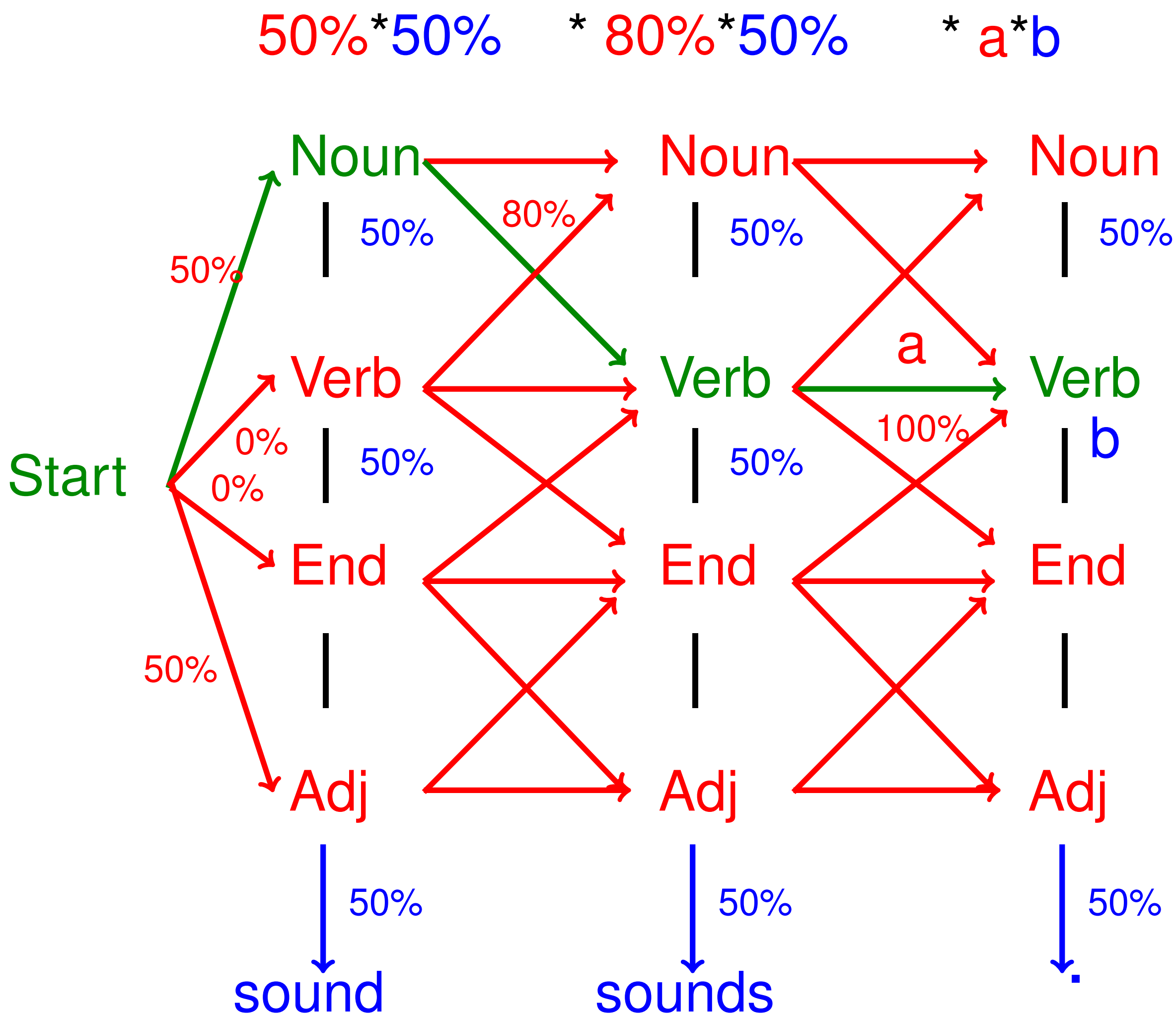


# The HMM as a Trellis

Task: compute the probability of every possible path from left to right, find maximum.

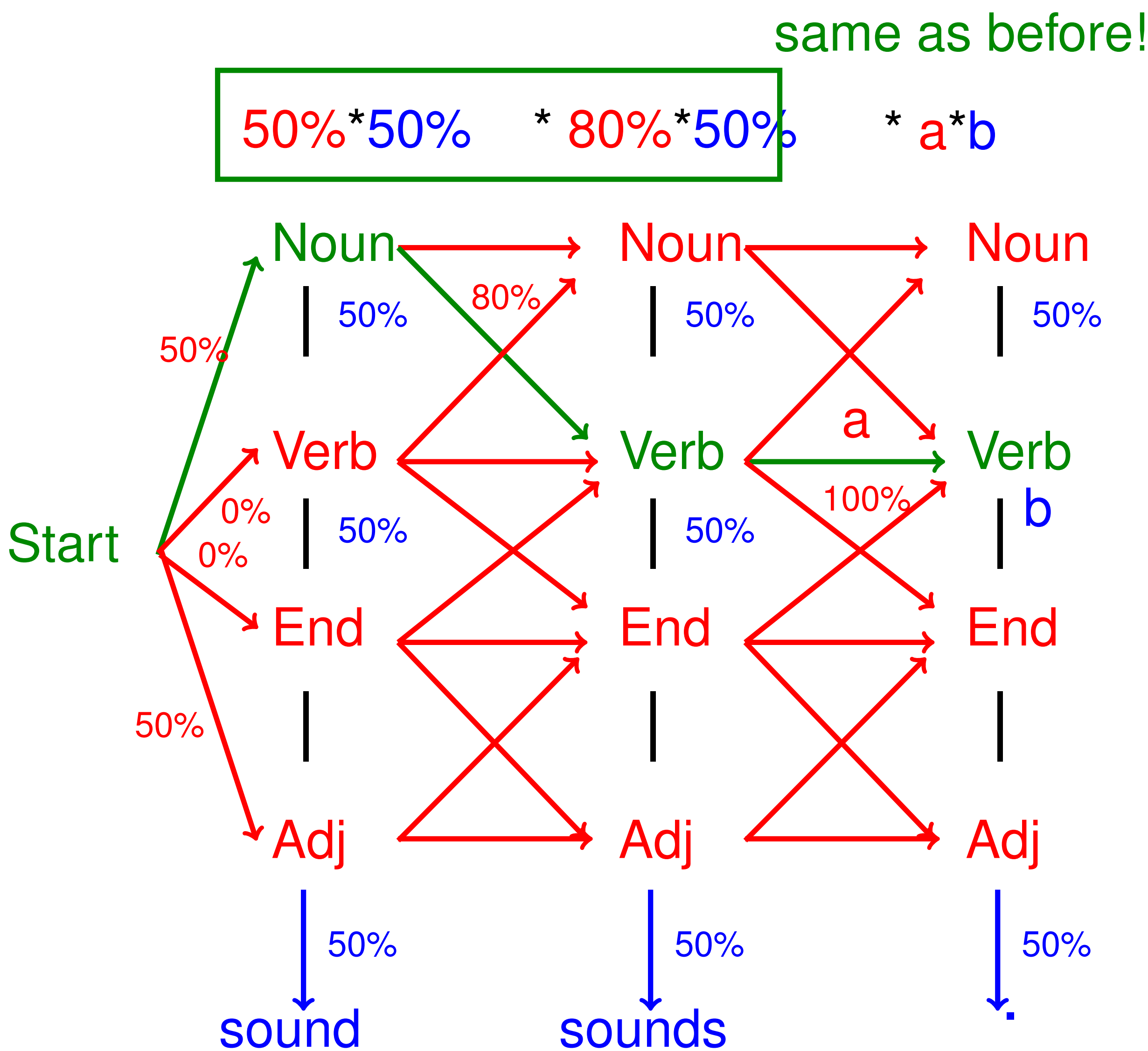


# The HMM as a Trellis



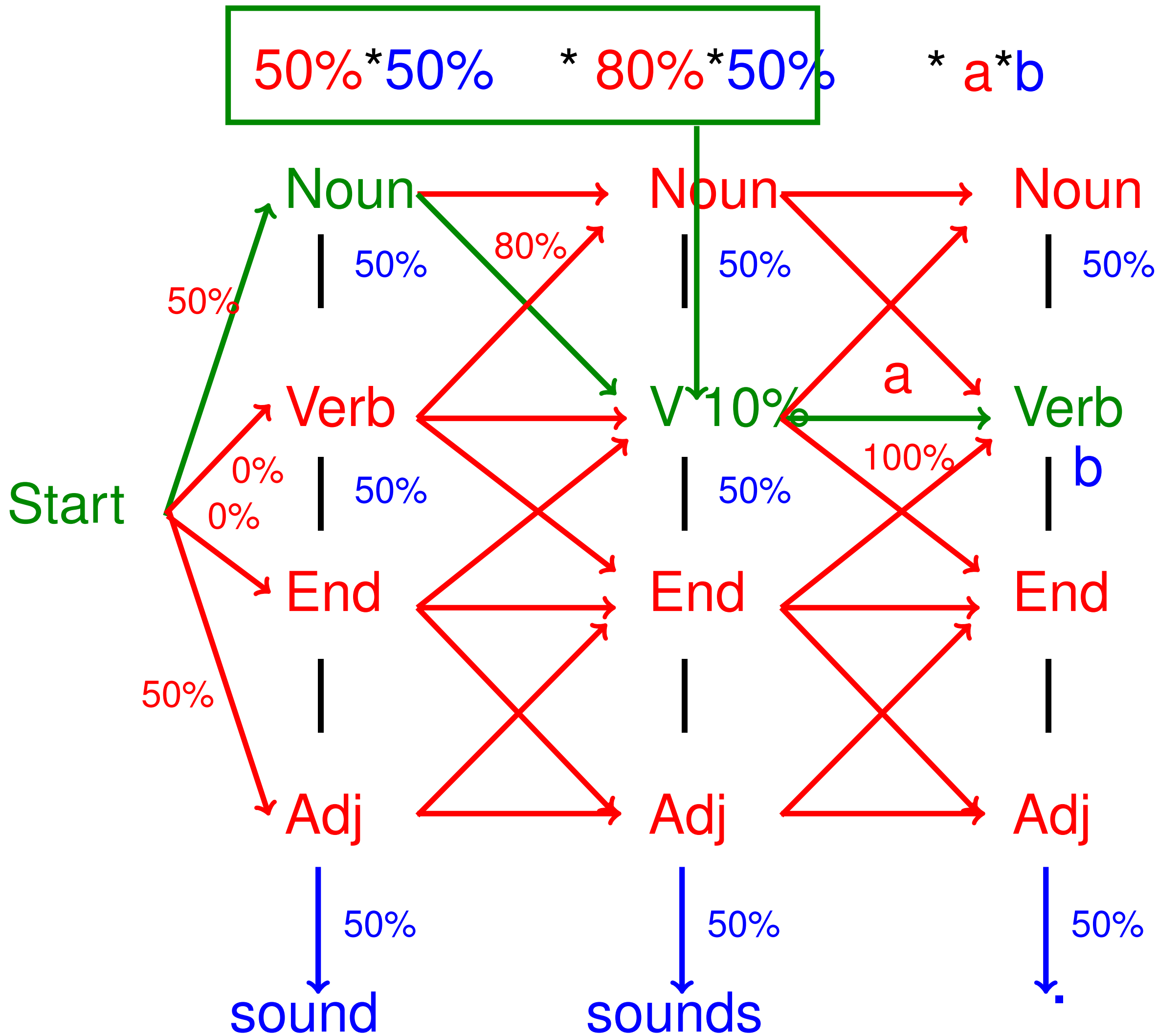

Complexity: $O(T^S)$

# The HMM as a Trellis



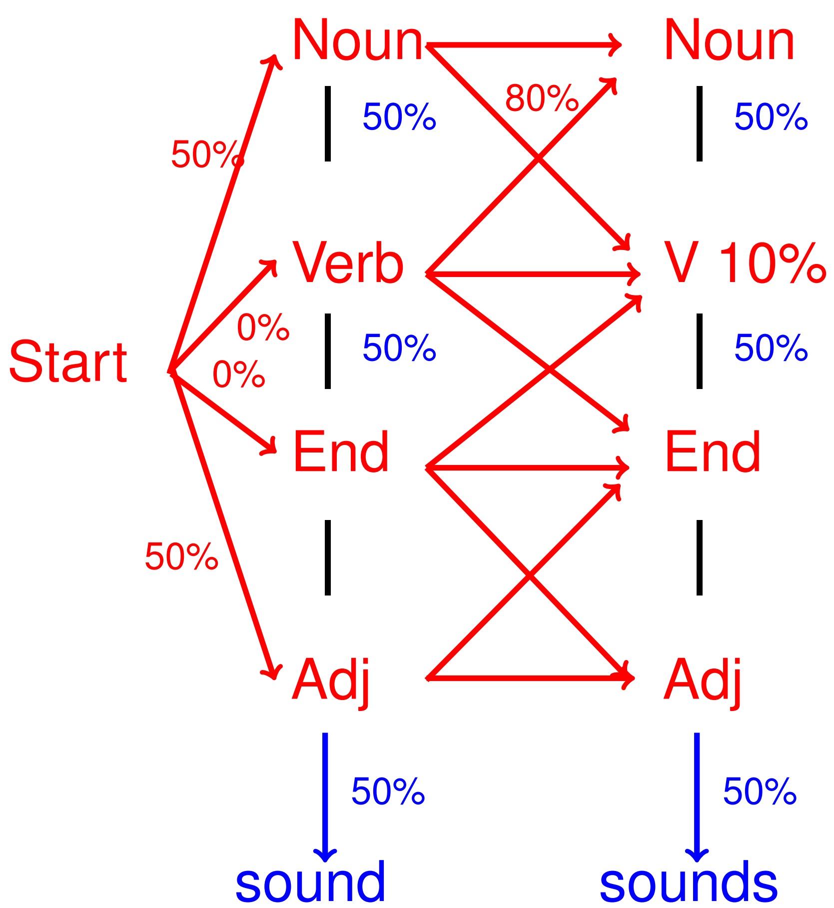

Complexity: $O(T^S)$

# The HMM as a Trellis



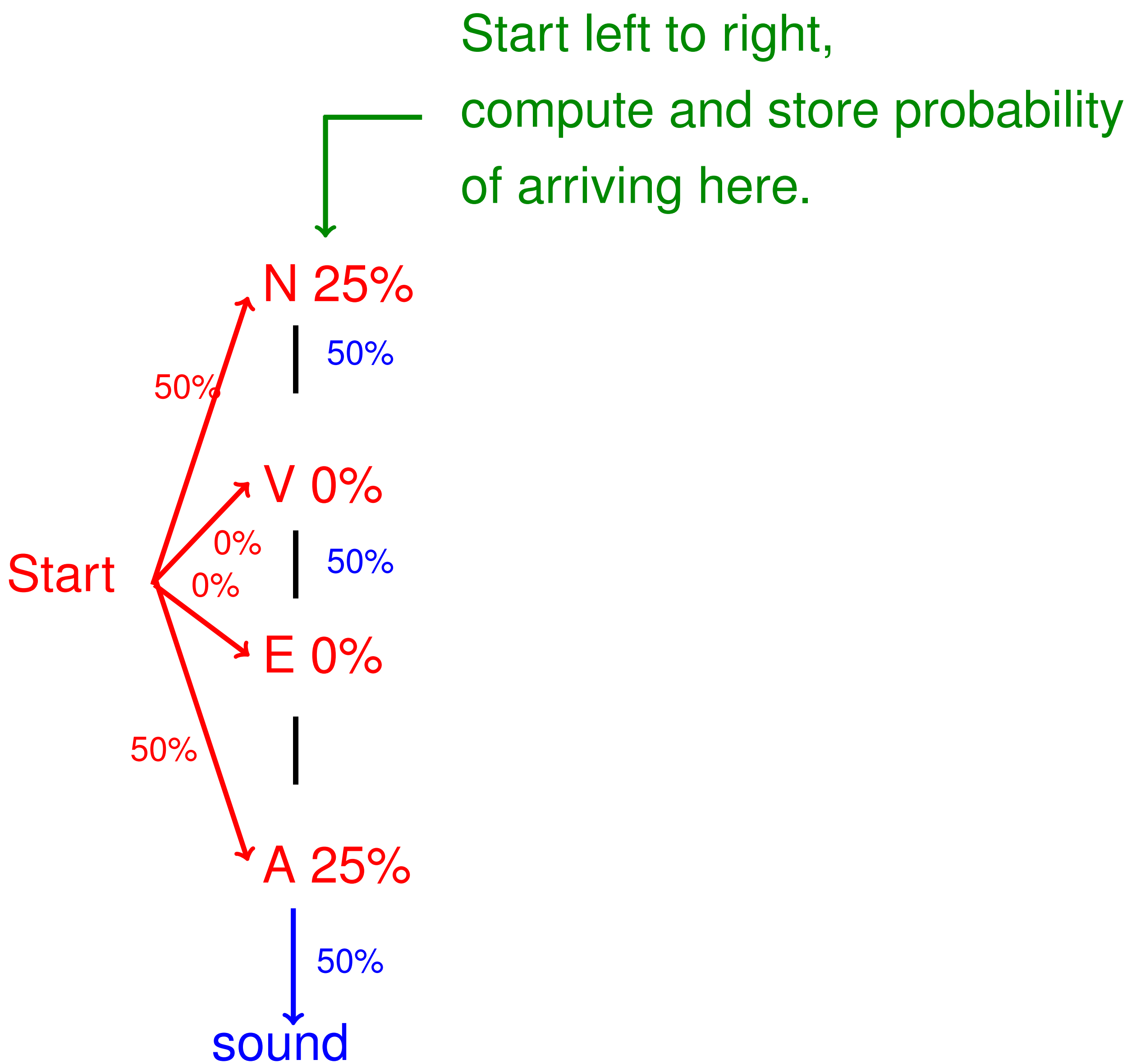

Idea: Store at each node the probability of the maximal path that leads there.

# Viterbi Algorithm

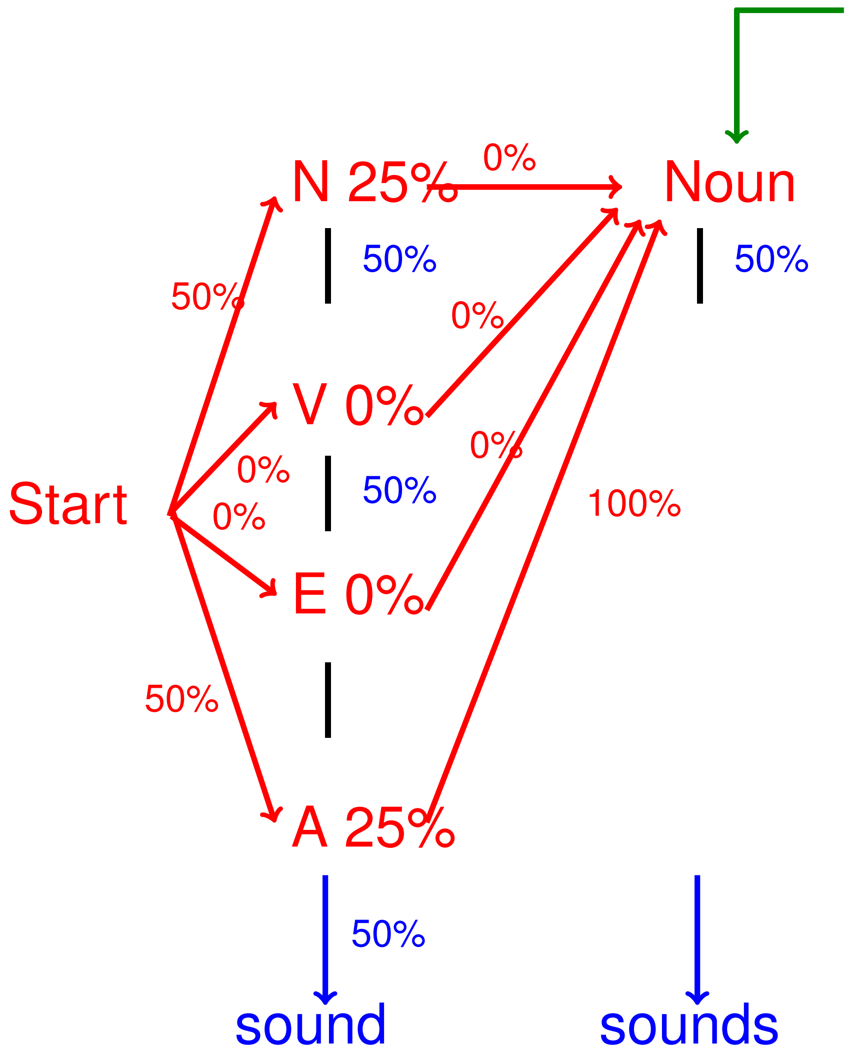



- For each word $w$
  - for each tag $t$
    - for each preceding tag $t'$
      - compute $P(t') \times P(t|t') \times P(w|t)$
    - store the maximal probability at $t, w$

# Viterbi Algorithm



# Viterbi Algorithm




Find prev that maximizes

$P(prev) \times P(t|prev) \quad \times P(w|t)$

$prev = Noun$ 25%*0%*50%

$prev = Verb$ 0%*0%*50%

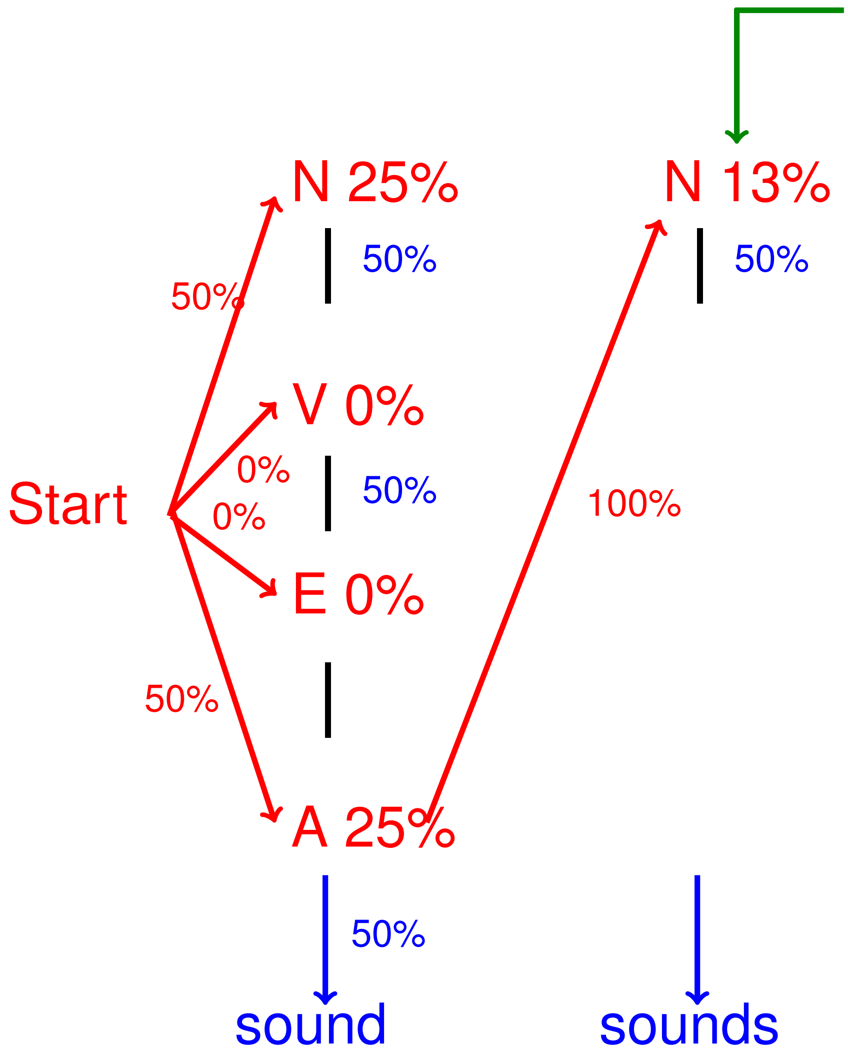
$prev = End$ 0%*0%*50%

$prev = Adj$ 25%*100%*50%

# Viterbi Algorithm

Find prev that maximizes

$P(prev) \times P(t|prev) \quad \times P(w|t)$



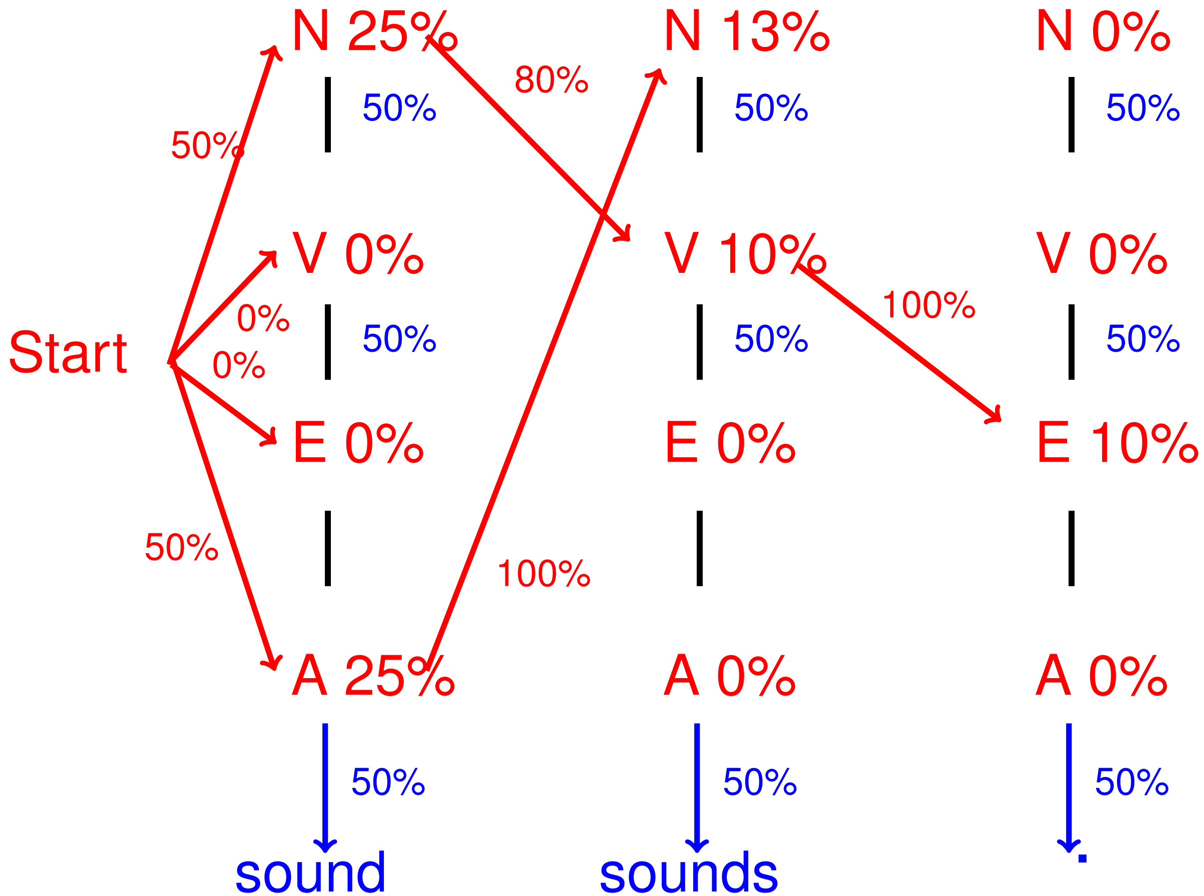

$prev = Noun$ 25%*0%*50%

$prev = Verb$ 0%*0%*50%

$prev = End$ 0%*0%*50%

$prev = Adj$ 25%*100%*50%

# Viterbi Algorithm




Read best path by following arrows backwards:
Start N V E

$$O(S \times T^2)$$

# Where do we get the HMM?

Estimate probabilities from manually annotated corpus:

Elvis/PN sings/Verb

Elvis/PN ./End

Priscilla/PN laughs/Verb

"Transition probability", Probability of PN followed by a Verb

$P(Verb|PN) = \frac{2}{3}$ $\qquad P(End|PN) = \frac{1}{3}$ $\qquad \ldots$

$P(Elvis|PN) = \frac{2}{3}$ $\qquad P(sings|Verb) = \frac{1}{2}$ $\qquad \ldots$

"Emission probability", Probability of PN is 'Elvis'

# Def: Probabilistic POS Tagging

Given a sentence and transition and emission probabilities, Probabilistic POS Tagging computes the sequence of tags that has maximal probability (in an HMM).

$\vec{X}$ = Elvis sings

$P(Elvis, sings, PN, N) = 0.01$

$P(Elvis, sings, V, N) = 0.01$ winner

$P(Elvis, sings, PN, V) = 0.1$

...

# Probabilistic POS Tagging

- Probabilistic POS tagging uses Hidden Markov Models
- General performance very good (>95% acc.)
- Several POS taggers are available

  - Stanford POS tagger
  - MBT: Memory-based Tagger
  - TreeTagger
  - ACOPOST
  - YamCha
  - ...

end>70

(HMMs and the Viterbi algorithm serve a wide variety of other tasks, in particular NLP at all levels of granularity, e.g., in speech processing)

# Research Questions

How can we deal with

- evil cases?
  - the word “blue” has 4 letters.
  - pre- and post-secondary
  - look it up
  - The Duchess was entertaining last night.

    [Wikipedia/POS tagging]
- unknown words?
- new languages?

end>70

# POS Tagging helps pattern IE

We can choose to match the placeholders with only nouns or proper nouns:

"X invents a Y"

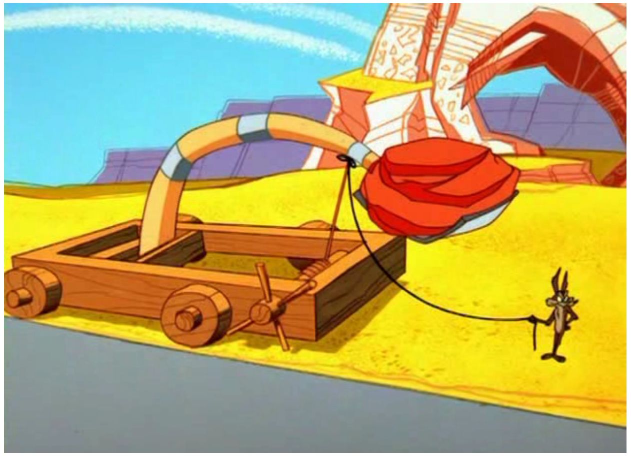
Coyote invents a catapult

invents(Coyote,catapult)

Coyote invents a really cool thing.

~~invents(Coyote,really)~~

# POS-tags can generalize patterns



“X invents a ADJ X”

Coyote invents a cool catapult — match

Coyote invents a great catapult — match

Coyote invents a super-duper catapult. — match

# Phrase structure is a problem

“X invents a ADJ X”

Coyote invents a very great catapult.

no match

Coyote, who is very hungry,
invents a great catapult.

no match

# References

Ramage: HMM Fundamentals

Web data mining class