Natural Language Processing

Hidden Markov Models

Joakim Nivre

Uppsala University
Department of Linguistics and Philology
joakim.nivre@lingfil.uu.se

Sequence Classification

- Part-of-speech tagging is a sequence classification problem
  - Input: Sequence $w_1, \ldots, w_n$ of words
  - Output: Sequence $t_1, \ldots, t_n$ of tags
- Why not just take one word at a time?

$$\textbf{for } i = 1 \textbf{ to } n \textbf{ do } \operatorname*{argmax}_{t_i} P(t_i | w_i, t_{i-1})$$

- Because the desired solution may be:

$$\operatorname*{argmax}_{t_1, \ldots, t_n} P(t_1, \ldots, t_n \mid w_1, \ldots, w_n)$$

- Two approaches:
  - Local optimization – simple, efficient, but may be suboptimal
  - Global optimization – may be computationally challenging

Hidden Markov Models

- Markov models are probabilistic sequence models used for problems such as:
  1. Speech recognition
  2. Spell checking
  3. Part-of-speech tagging
  4. Named entity recognition
- A Markov model traverses a sequence of states emitting signals
- If the signal sequence does not uniquely determine the state sequence, the model is said to be hidden
- Markov models have efficient algorithms for optimization

Hidden Markov Models

- A Markov model consists of five elements:
  1. A finite set of states $\Omega = \{s_1, \ldots, s_k\}$
  2. A finite signal alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$
  3. Initial probabilities $P(s)$ (for every $s \in \Omega$) defining the probability of starting in state s
  4. Transition probabilities $P(s_i \mid s_j)$ (for every $(s_i, s_j) \in \Omega^2$) defining the probability of going from state s_j to state s_i
  5. Emission probabilities $P(\sigma \mid s)$ (for every $(\sigma, s) \in \Sigma \times \Omega$) defining the probability of emitting symbol σ in state s

Hidden Markov Models

- A Markov model consists of five elements:
  1. A finite set of states $\Omega = \{s_1, \ldots, s_k\}$
  2. A finite signal alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$
  3. Initial probabilities $P(s)$ (for every $s \in \Omega$) defining the probability of starting in state s
  4. Transition probabilities $P(s_i \mid s_j)$ (for every $(s_i, s_j) \in \Omega^2$) defining the probability of going from state s_j to state s_i
  5. Emission probabilities $P(\sigma \mid s)$ (for every $(\sigma, s) \in \Sigma \times \Omega$) defining the probability of emitting symbol σ in state s
- We can get rid of initial probabilities by adding a start state

Hidden Markov Models

- A Markov model consists of four elements:
  1. A finite set of states $\Omega = \{s_0, s_1, \ldots, s_k\}$
  2. A finite signal alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ (for every $s \in \Omega$) defining the probability of starting in state s
  3. Transition probabilities $P(s_i \mid s_j)$ (for every $(s_i, s_j) \in \Omega^2$) defining the probability of going from state s_j to state s_i
  4. Emission probabilities $P(\sigma \mid s)$ (for every $(\sigma, s) \in \Sigma \times \Omega$) defining the probability of emitting symbol σ in state s

Markov Assumptions

- State transitions are assumed to be independent of everything except the current state:

$$P(s_1, \ldots, s_n) = \prod_{i=1}^{n} P(s_i \mid s_{i-1})$$

- Signal emissions are assumed to be independent of everything except the current state:

$$P(s_1, \ldots, s_n, \sigma_1, \ldots, \sigma_n) = P(s_1, \ldots, s_n) \prod_{i=1}^{n} P(\sigma_i \mid s_i)$$

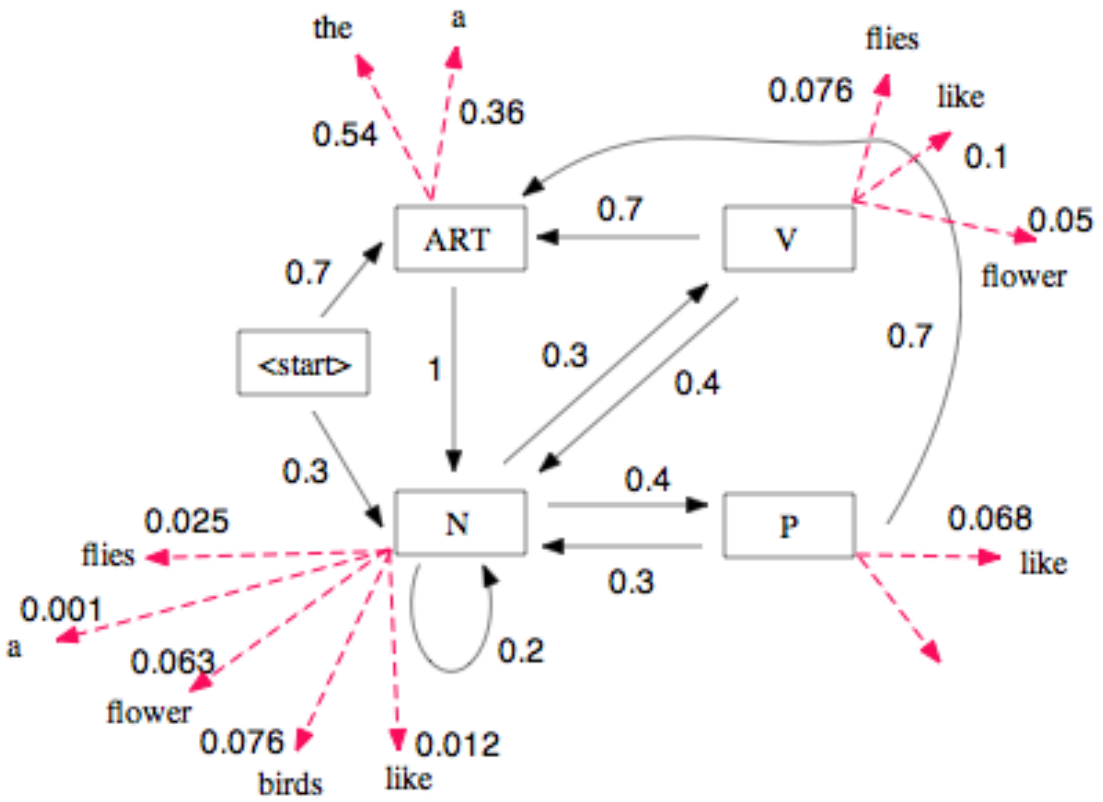
Tagging with an HMM

- Modeling assumptions:
  1. States represent tags (or tag sequences)
  2. Signals represent words
- Probability model (first-order):

$$P(w_1, \ldots, w_n, t_1, \ldots, t_n) = \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

  - An n-th order model conditions tags on n preceding tags
- Ambiguity:
  - The same word (signal) generated by different tags (states)
  - Language is ambiguous $\Leftrightarrow$ Markov model is hidden

A Simple HMM


the
a
0.54
0.36
flies
0.076
like
0.1
0.05
flower
ART
V
0.7
0.7
<start>
1
0.3
0.4
0.7
0.3
0.4
N
P
0.025
flies
0.068
like
0.3
0.001
a
0.063
flower
0.2
0.076
birds
0.012
like

A Versatile Model

Problem	State	Signal
Part-of-speech tagging	Tag	Word
Speech recognition	Phoneme	Sound
Optical character recognition	Character	Pixels
Predictive text entry	Character	Keystroke
⋮	⋮	⋮

Problems for HMMs

- Decoding = finding the optimal state sequence

$$\underset{s_1,\ldots,s_n}{\operatorname{argmax}} P(s_1, \ldots, s_n, \sigma_1, \ldots, \sigma_n)$$

- Learning = estimating the model parameters

$$\hat{P}(s_i|s_j) \quad \text{for all } s_i, s_j \qquad\qquad \hat{P}(\sigma|s) \quad \text{for all } s, \sigma$$

Quiz

- In a simple substitution cipher, every letter in the alphabet is replaced by exactly one symbol and that symbol is never used to represent another letter. For example, mummy would be written abaac in a cipher where m $\rightarrow$ a; u $\rightarrow$ b; and y $\rightarrow$ c.
- In a homophonic substitution cipher, more than one symbol may be used to replace the same letter but never to represent another letter. For example, mummy could be written abcde in a cipher where m $\rightarrow$ a, c, or d; u $\rightarrow$ b; and y $\rightarrow$ e.
- Suppose we use HMMs to relate sequences of plaintext characters and sequences of cipher characters. Which of the following HMMs are hidden?

	Cipher	State	Signal
1	Simple	Plaintext	Cipher
2	Simple	Cipher	Plaintext
3	Homophonic	Plaintext	Cipher
4	Homophonic	Cipher	Plaintext

Natural Language Processing

HMM Decoding

Joakim Nivre

Uppsala University
Department of Linguistics and Philology
joakim.nivre@lingfil.uu.se

Decoding

- Decoding = finding the optimal state sequence

$$\operatorname*{argmax}_{s_1,\ldots,s_n} P(s_1,\ldots,s_n,\sigma_1,\ldots,\sigma_n)$$

- Difficulty:
  - Maximizing over all possible state sequences
  - The number $|\Omega|^n$ of sequences grows exponentially
- Key observation for dynamic programming:
  - Solution of size n contains solution of size $n-1$

$$\operatorname{argmax}_{s_1,\ldots,s_n} P(s_1,\ldots,s_n,\sigma_1,\ldots,\sigma_n) =$$
$$\operatorname{argmax}_{s_n} \operatorname{argmax}_{s_1,\ldots,s_{n-1}} P(s_1,\ldots,s_{n-1},\sigma_1,\ldots,\sigma_{n-1})P(s_n|s_{n-1})P(\sigma_n|s_n)$$

The Viterbi Algorithm

- Best probability and preceding state at step i and state s:

$$\begin{aligned} \text{viterbi}(i, s) &\Leftrightarrow \max_{s'} P(\sigma_1, \ldots, \sigma_i, s_{i-1} = s', s_i = s) \\ \text{ptr}(i, s) &\Leftrightarrow \text{argmax}_{s'} P(\sigma_1, \ldots, \sigma_i, s_{i-1} = s', s_i = s) \end{aligned}$$

The Viterbi Algorithm

- Best probability and preceding state at step i and state s:

$$\begin{aligned} \text{viterbi}(i,s) &\Leftrightarrow \max_{s'} P(\sigma_1,\ldots,\sigma_i, s_{i-1}=s', s_i=s) \\ \text{ptr}(i,s) &\Leftrightarrow \text{argmax}_{s'} P(\sigma_1,\ldots,\sigma_i, s_{i-1}=s', s_i=s) \end{aligned}$$

- Compute for all i and s recursively:

$$\begin{aligned} \text{viterbi}(1,s) &= P(s|s_0)P(\sigma_1|s) \\ \text{ptr}(1,s) &= s_0 \\ \\ \text{viterbi}(i,s) &= \max_{s'} \text{viterbi}(i-1,s')P(s|s')P(\sigma_i|s) \\ \text{ptr}(i,s) &= \text{argmax}_{s'} \text{viterbi}(i-1,s')P(s|s')P(\sigma_i|s) \end{aligned}$$

The Viterbi Algorithm

- Best probability and preceding state at step i and state s:

$$\begin{aligned} \text{viterbi}(i,s) &\Leftrightarrow \max_{s'} P(\sigma_1,\ldots,\sigma_i, s_{i-1}=s', s_i=s) \\ \text{ptr}(i,s) &\Leftrightarrow \text{argmax}_{s'} P(\sigma_1,\ldots,\sigma_i, s_{i-1}=s', s_i=s) \end{aligned}$$

- Compute for all i and s recursively:

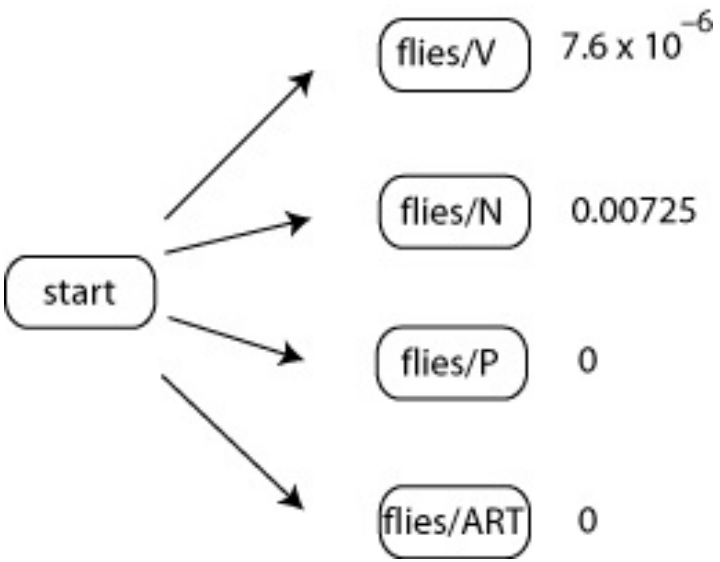
$$\begin{aligned} \text{viterbi}(1,s) &= P(s|s_0)P(\sigma_1|s) \\ \text{ptr}(1,s) &= s_0 \end{aligned}$$

$$\begin{aligned} \text{viterbi}(i,s) &= \max_{s'} \text{viterbi}(i-1,s')P(s|s')P(\sigma_i|s) \\ \text{ptr}(i,s) &= \text{argmax}_{s'} \text{viterbi}(i-1,s')P(s|s')P(\sigma_i|s) \end{aligned}$$

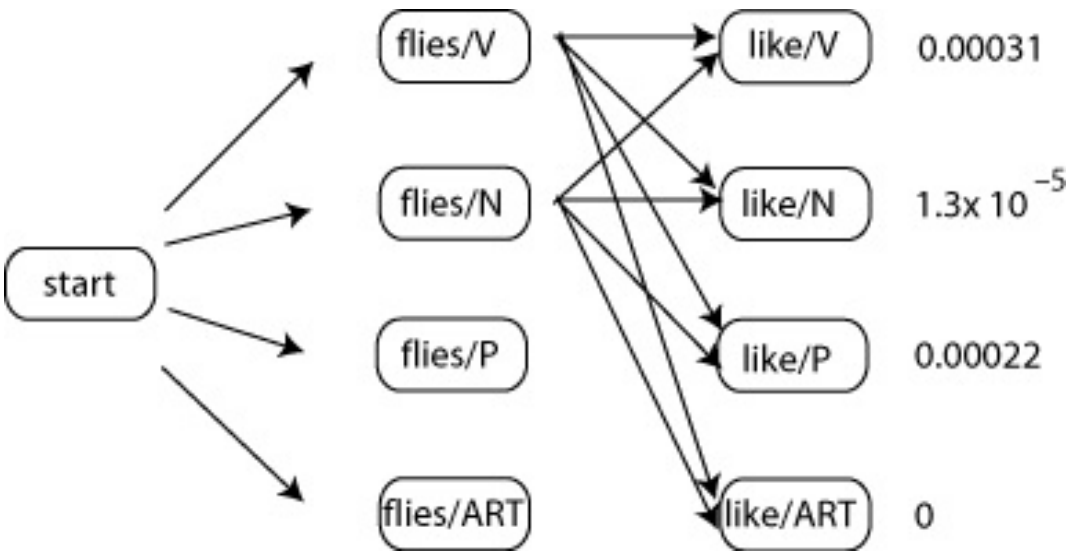
- Find best end state and follow pointers backwards:

$$\max_s \text{viterbi}(n,s) = \max_{s_1,\ldots,s_n} P(\sigma_1,\ldots,\sigma_n,s_1,\ldots,s_n)$$

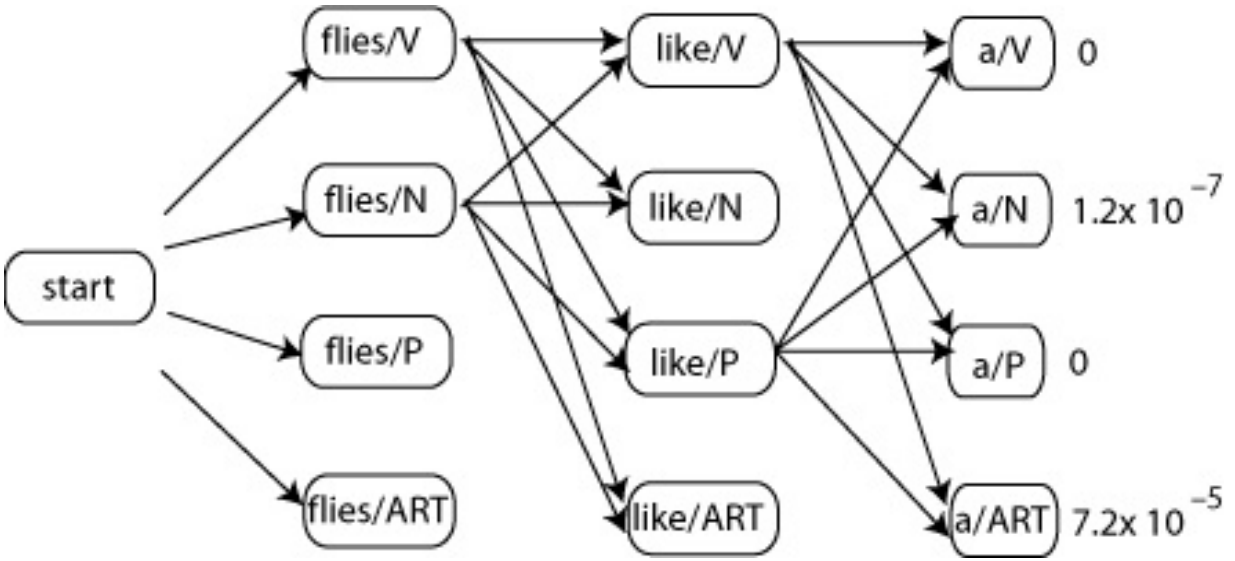
Viterbi Example



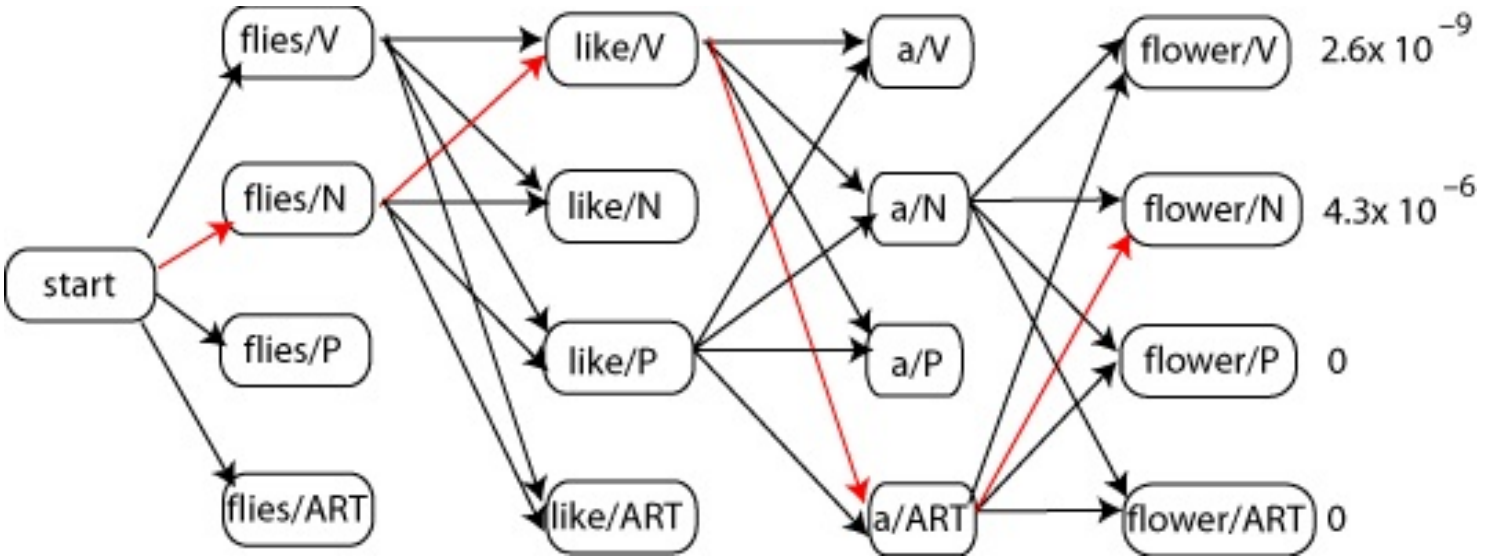
Viterbi Example



Viterbi Example

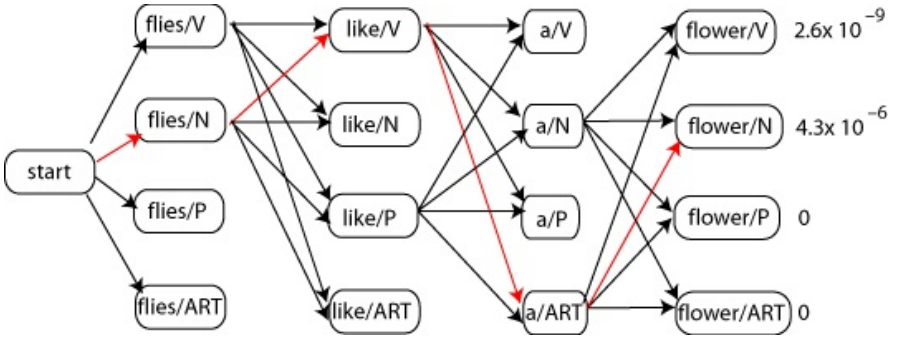


Viterbi Example



Quiz

- Consider the following Viterbi computation:



- What is viterbi$(4, N)$?
  1. $2.6 \cdot 10^{-9}$
  2. $4.3 \cdot 10^{-6}$
  3. 0
- What is ptr$(4, N)$?
  1. V
  2. N
  3. ART

Natural Language Processing

HMM Learning

Joakim Nivre

Uppsala University
Department of Linguistics and Philology
joakim.nivre@lingfil.uu.se

Learning

- Learning = estimating the model parameters

$$\hat{P}(s_i|s_j) \quad \text{for all } s_i, s_j \qquad\qquad \hat{P}(\sigma|s) \quad \text{for all } s, \sigma$$

- Supervised learning:
  - Given a labeled training corpus, we can estimate parameters using (smoothed) relative frequencies
- Weakly supervised learning:
  - Given a dictionary and an unlabeled training corpus, we can use Expectation-Maximization to estimate parameters

Maximum Likelihood Estimation

- With labeled data $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, we maximize the joint likelihood of input and output:

$$\underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{n} P_\theta(y_i) P_\theta(x_i|y_i)$$

- With unlabeled data $D = \{x_1, \ldots, x_n\}$, we can instead maximize the marginal likelihood of the input:

$$\underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{n} \sum_{y \in \Omega_Y} P_\theta(y) P_\theta(x_i|y)$$

- In this case, Y is a hidden variable that is marginalized out

Expectation-Maximization

- Goal:

$$\underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{n} \sum_{y \in \Omega_Y} P_\theta(y) P_\theta(x_i|y)$$

- Problem:
  - Maximizing a product of sums is mathematically hard
  - In most cases, there is no closed form solution
- Solution:
  - Use numerical approximation methods
  - Most common approach: Expectation-Maximization (EM)

Expectation-Maximization

- Basic observations:
  - If we know $f(x, y)$, we can find the MLE θ.
  - If we know $f(x)$ and the MLE θ, we can derive $f(x, y)$.
  - If $P_\theta(y|x) = p$ and $f(x) = n$, then $f(x, y) = np$.
- Basic idea behind EM:
  1. Start by guessing θ
  2. Compute expected counts $E[f(x, y)] = P_\theta(y|x)f(x)$
  3. Find MLE θ given expectation $E[f(x, y)]$
  4. Repeat steps 2 and 3 until convergence

EM for Hidden Markov Models

- Computing expectations:

$$\begin{aligned} E[f(s,s')] &= \sum_{i=1}^{n-1} P(\sigma_1,\ldots,\sigma_n, s_i = s, s_{i+1} = s') \\ E[f(s,\sigma)] &= \sum_{i=1}^{n} P(\sigma_1,\ldots,\sigma_{i-1}, \sigma_i = \sigma,\ldots,\sigma_n, s_i = s) \end{aligned}$$

- Difficulty:
  - Summing over all possible state sequences
  - The number $|\Omega|^n$ of sequences grows exponentially
- Sounds familiar?
  - We can use dynamic programming again
  - The forward-backward algorithm

The Forward-Backward Algorithm

- Probability of signal sequence (Forward):

$$\begin{aligned} \alpha(i,s) &\Leftrightarrow P(\sigma_1,\ldots,\sigma_i,s_i=s) \\ \alpha(1,s) &= P(s|s_0)P(\sigma_1|s) \\ \alpha(i,s) &= \textstyle\sum_{s'} \alpha(i-1,s')P(s|s')P(\sigma_i|s) \\ \textstyle\sum_s \alpha(n,s) &= P(\sigma_1,\ldots,\sigma_n) \end{aligned}$$

- Probability of signal sequence (Backward):

$$\begin{aligned} \beta(i,s) &\Leftrightarrow P(\sigma_{i+1},\ldots,\sigma_n|s_i=s) \\ \beta(n,s) &= 1 \\ \beta(i,s) &= \textstyle\sum_{s'} \beta(i+1,s')P(s'|s)P(\sigma_{i+1}|s') \\ \textstyle\sum_s \beta(1,s)P(s|s_0)P(\sigma_1|s) &= P(\sigma_1,\ldots,\sigma_n) \end{aligned}$$

- Expected counts of hidden states (Forward-Backward):

$$\begin{aligned} E[C(s,s')] &= \textstyle\sum_{i=1}^{n} \alpha(i,s)P(s'|s)P(\sigma_{i+1}|s')\beta(i+1,s') \\ E[C(s,\sigma)] &= \textstyle\sum_{i:\sigma_i=\sigma} \alpha(i,s)P(\sigma_i|s)\beta(i,s) \end{aligned}$$

Quiz

- Assume we have the following tagged corpus:

  a/DET can/NOUN can/AUX trash/VERB a/DET trash/NOUN can/NOUN

- What is the MLE of $P(\text{AUX}|\text{NOUN})$?
  1. 0.0
  2. 0.5
  3. 1.0
- What is the MLE of $P(\text{trash}|\text{NOUN})$?
  1. 0.33
  2. 0.5
  3. 0.67