# CS11-711: Algorithms for NLP

## Sequence labeling, POS tagging, Viterbi

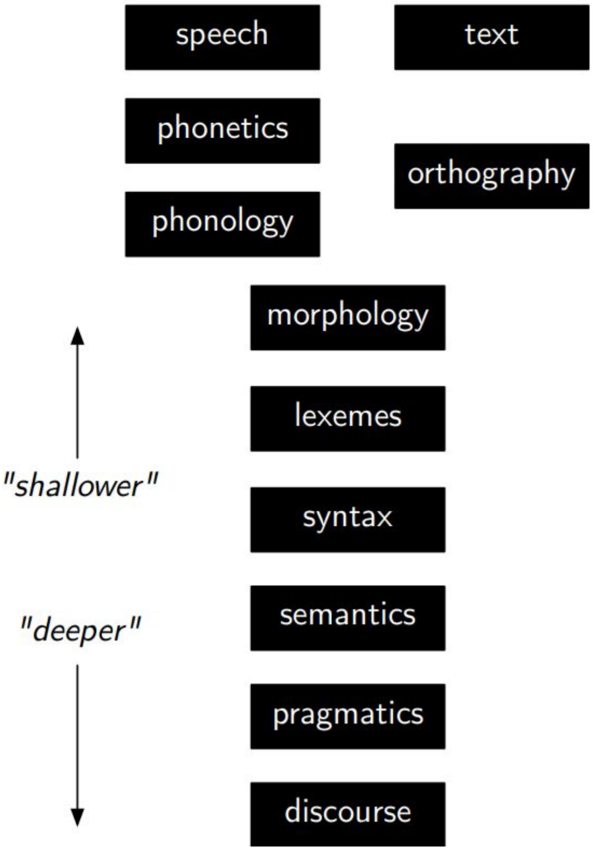
Yulia Tsvetkov

Carnegie Mellon University
Language Technologies Institute

# Readings for today's lecture

- J&M SLP3 https://web.stanford.edu/~jurafsky/slp3/8.pdf
- Collins (2011) http://www.cs.columbia.edu/~mcollins/hmms-spring2013.pdf

# Levels of linguistic knowledge




| | |
|---|---|
| Phonetics | The study of the sounds of human language |
| Phonology | The study of sound systems in human language |
| Morphology | The study of the formation and internal structure of words |
| Syntax | The study of the formation and internal structure of sentences |
| Semantics | The study of the meaning of sentences |
| Pragmatics | The study of the way sentences with their semantic meanings are used for particular communicative goals |

# Part of speech tagging

| | | | | | |
|---|---|---|---|---|---|
| **PART OF SPEECH** | DT | VBZ | DT | JJ | NN |
| **WORDS** | This | is | a | simple | sentence |

# Sequence labeling problems

## Map a sequence of words to a sequence of labels

- Part-of-speech tagging (Church, 1988; Brants, 2000)
- Named entity recognition (Bikel et al., 1999)
- Text chunking and shallow parsing (Ramshaw and Marcus, 1995)
- Word alignment of parallel text (Vogel et al., 1996)
- Compression (Conroy and O'Leary, 2001)
- Acoustic models, discourse segmentation, etc.

# Parts of speech

- Open classes
  - nouns
  - verbs
  - adjectives
  - adverbs
- Closed classes
  - prepositions
  - determiners
  - pronouns
  - conjunctions
  - auxiliary verbs

# Parts of speech, more fine-grained classes

- Open classes
  - nouns
    - proper
    - common
      - count
      - mass
  - verbs
  - adjectives
  - adverbs
    - directional
    - degree
    - manner
    - temporal

*Actually*, I ran home *extremely quickly yesterday*

# Parts of speech, closed classes

**prepositions:** on, under, over, near, by, at, from, to, with
**particles:** up, down, on, off, in, out, at, by
**determiners:** a, an, the
**conjunctions:** and, but, or, as, if, when
**pronouns:** she, who, I, others
**auxiliary verbs:** can, may, should, are
**numerals:** one, two, three, first, second, third

# Part of speech tagsets

- Penn treebank tagset (Marcus et al., 1993)

| Tag | Description | Example | Tag | Description | Example | Tag | Description | Example |
|---|---|---|---|---|---|---|---|---|
| CC | coordinating conjunction | and, but, or | PDT | predeterminer | all, both | VBP | verb non-3sg present | eat |
| CD | cardinal number | one, two | POS | possessive ending | 's | VBZ | verb 3sg pres | eats |
| DT | determiner | a, the | PRP | personal pronoun | I, you, he | WDT | wh-determ. | which, that |
| EX | existential 'there' | there | PRP$ | possess. pronoun | your, one's | WP | wh-pronoun | what, who |
| FW | foreign word | mea culpa | RB | adverb | quickly | WP$ | wh-possess. | whose |
| IN | preposition/ subordin-conj | of, in, by | RBR | comparative adverb | faster | WRB | wh-adverb | how, where |
| JJ | adjective | yellow | RBS | superlatv. adverb | fastest | $ | dollar sign | $ |
| JJR | comparative adj | bigger | RP | particle | up, off | # | pound sign | # |
| JJS | superlative adj | wildest | SYM | symbol | +,%, & | “ | left quote | ‘ or “ |
| LS | list item marker | 1, 2, One | TO | “to” | to | ” | right quote | ’ or ” |
| MD | modal | can, should | UH | interjection | ah, oops | ( | left paren | [, (, {, < |
| NN | sing or mass noun | llama | VB | verb base form | eat | ) | right paren | ], ), }, > |
| NNS | noun, plural | llamas | VBD | verb past tense | ate | , | comma | , |
| NNP | proper noun, sing. | IBM | VBG | verb gerund | eating | . | sent-end punc | . ! ? |
| NNPS | proper noun, plu. | Carolinas | VBN | verb past part. | eaten | : | sent-mid punc | : ; ... – - |

# Example of POS tagging

The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.

**There/EX** are/VBP 70/CD children/NNS **there/RB**

Preliminary/JJ findings/NNS were/VBD **reported/VBN** in/IN today/NN **'s/POS** New/NNP England/NNP Journal/NNP of/IN Medicine/NNP ./.

# The Universal Dependencies

## Universal Dependencies

Universal Dependencies (UD) is a framework for consistent annotation of grammar (parts of speech, morphological features, and syntactic dependencies) across different human languages. UD is an open community effort with over 300 contributors producing more than 150 treebanks in 90 languages. If you're new to UD, you should start by reading the first part of the Short Introduction and then browsing the annotation guidelines.

- Short introduction to UD
- UD annotation guidelines
- More information on UD:
  - How to contribute to UD
  - Tools for working with UD
  - Discussion on UD
  - UD-related events
- Query UD treebanks online:
  - SETS treebank search maintained by the University of Turku
  - PML Tree Query maintained by the Charles University in Prague
  - Kontext maintained by the Charles University in Prague
  - Grew-match maintained by Inria in Nancy
  - INESS maintained by the University of Bergen
- Download UD treebanks

| Open class words | Closed class words | Other |
|---|---|---|
| ADJ | ADP | PUNCT |
| ADV | AUX | SYM |
| INTJ | CCONJ | X |
| NOUN | DET | |
| PROPN | NUM | |
| VERB | PART | |
| | PRON | |
| | SCONJ | |

# Why POS tagging

- Goal: resolve ambiguities
- Text-to-speech
  - record, lead, protest
- Lemmatization
  - saw/V → see, saw/N → saw
- Preprocessing for harder disambiguation problems
  - syntactic parsing
  - semantic parsing

# Ambiguities in POS tags

| Types: | | WSJ | | Brown | |
|---|---|---|---|---|---|
| **Unambiguous** | (1 tag) | 44,432 | (**86%**) | 45,799 | (**85%**) |
| **Ambiguous** | (2+ tags) | 7,025 | (**14%**) | 8,050 | (**15%**) |

# Ambiguities in POS tags

| **Types:** | | **WSJ** | | **Brown** | |
|---|---|---|---|---|---|
| **Unambiguous** | (1 tag) | 44,432 | (**86%**) | 45,799 | (**85%**) |
| **Ambiguous** | (2+ tags) | 7,025 | (**14%**) | 8,050 | (**15%**) |
| **Tokens:** | | | | | |
| **Unambiguous** | (1 tag) | 577,421 | (**45%**) | 384,349 | (**33%**) |
| **Ambiguous** | (2+ tags) | 711,780 | (**55%**) | 786,646 | (**67%**) |

# Most frequent class baseline

- Assigning each token to the class it occurred in most often in the training set
- Always compare a classifier against a baseline at least as good as the most frequent class baseline
- The WSJ training corpus and test on sections 22-24 of the same corpus the most-frequent-tag baseline achieves an accuracy of 92.34%.
- 97% tag accuracy achievable by most algorithms (HMMs, MEMMs, neural networks, rule-based algorithms)

# Sequence labeling as text classification

$$\hat{y}_i = \operatorname*{argmax}_{y \in \mathcal{L}} s(\boldsymbol{x}, i, y)$$

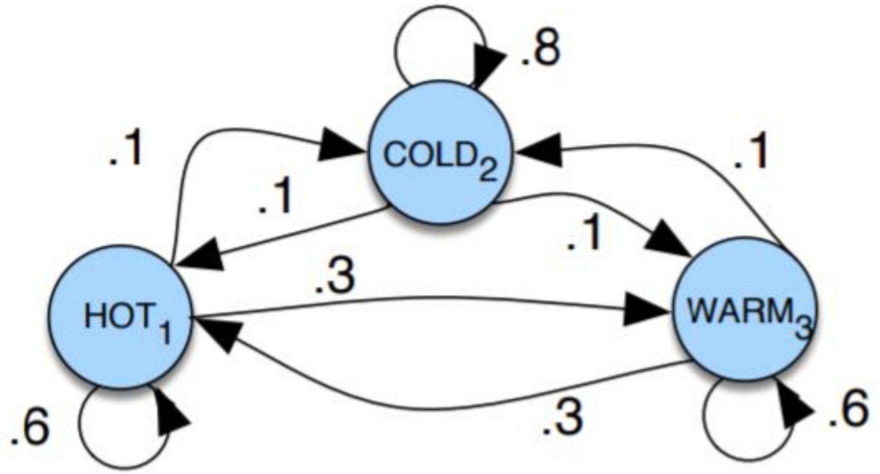
# Generative sequence labeling: Hidden Markov Models

# Markov chain

Formally, a Markov chain is specified by the following components:

| | |
|---|---|
| $Q = q_1 q_2 \ldots q_N$ | a set of $N$ **states** |
| $A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$ | a **transition probability matrix** $A$, each $a_{ij}$ representing the probability of moving from state $i$ to state $j$, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$ |
| $\pi = \pi_1, \pi_2, \ldots, \pi_N$ | an **initial probability distribution** over states. $\pi_i$ is the probability that the Markov chain will start in state $i$. Some states $j$ may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$ |

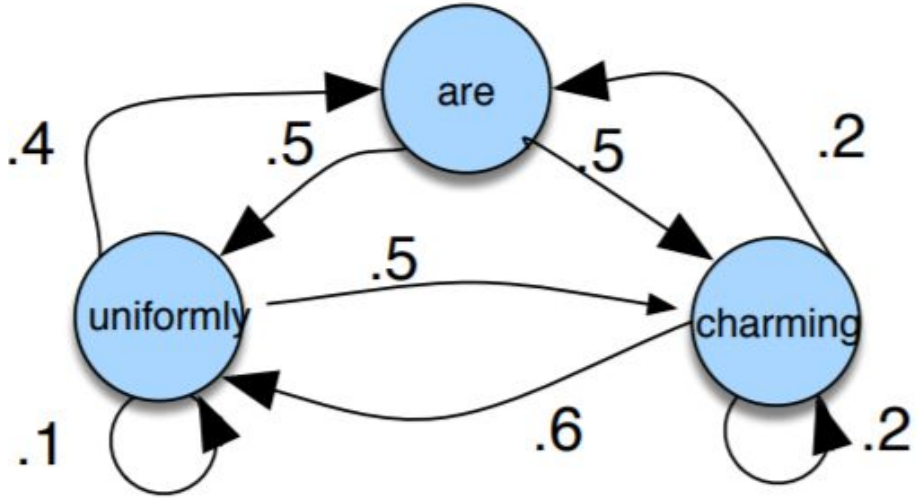
# Markov Chain: weather




**Markov Assumption:** $P(q_i = a | q_1 ... q_{i-1}) = P(q_i = a | q_{i-1})$

the future is independent of the past given the present

# Markov Chain: words

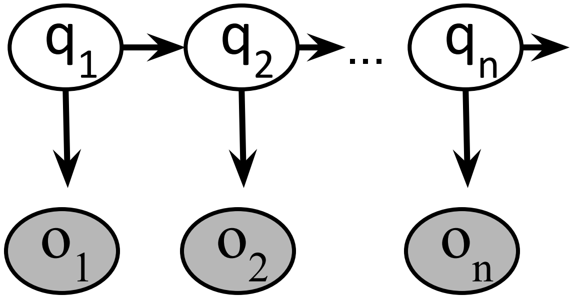



$\pi = [0.1, 0.7, 0.2]$

the future is independent of the past given the present

# Hidden Markov Models

- In real world many events are not observable
- Speech recognition: we observe acoustic features but not the phones
- POS tagging: we observe words but not the POS tags



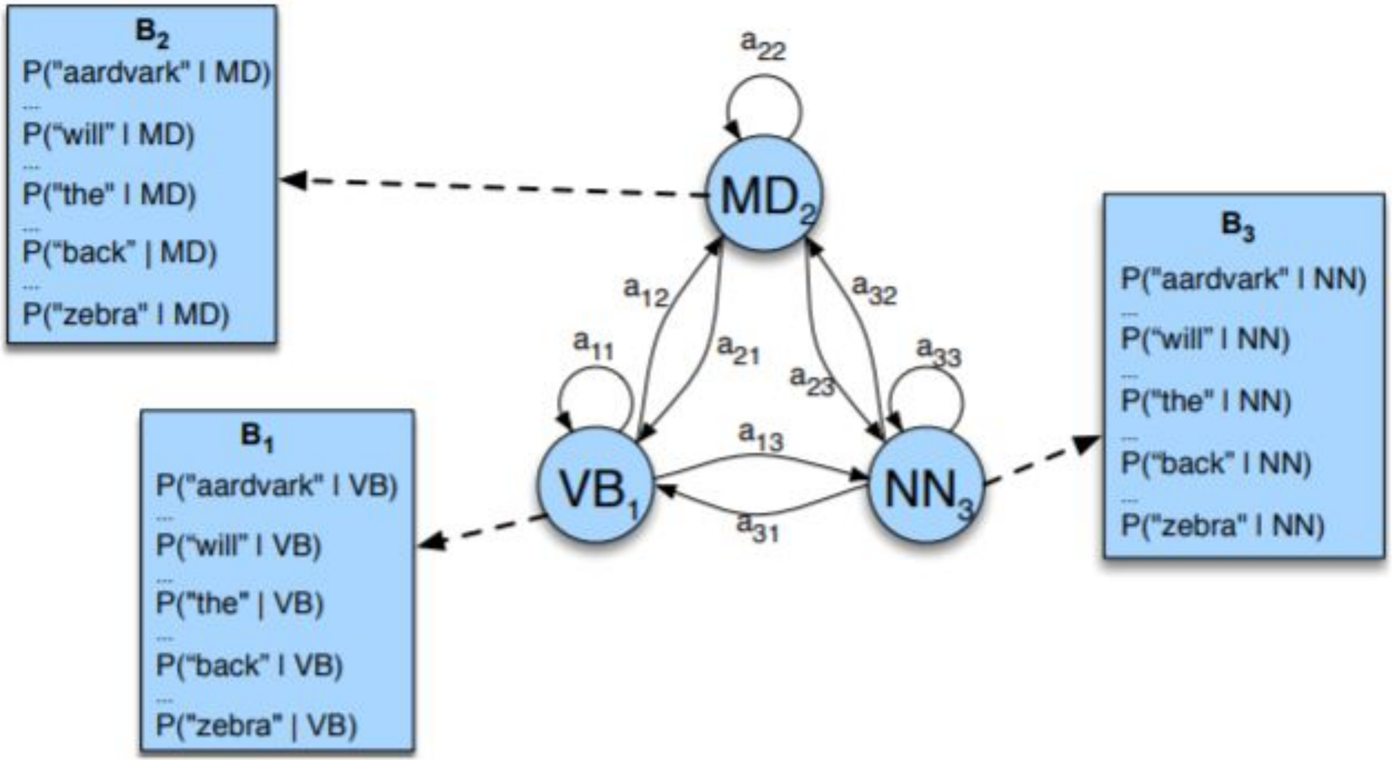

$$\textbf{Markov Assumption:} \quad P(q_i|q_1...q_{i-1}) = P(q_i|q_{i-1})$$

$$\textbf{Output Independence:} \quad P(o_i|q_1 \ldots q_i, \ldots, q_T, o_1, \ldots, o_i, \ldots, o_T) = P(o_i|q_i)$$

# Hidden Markov Models (HMMs)

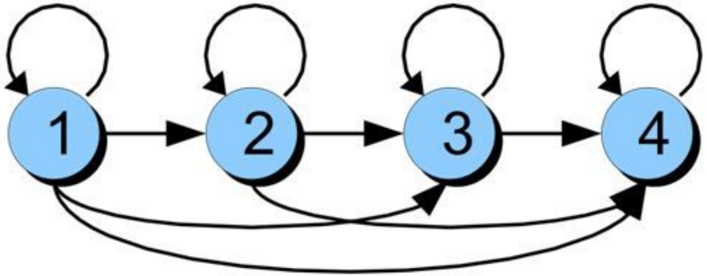
| | |
|---|---|
| $Q = q_1 q_2 \ldots q_N$ | a set of $N$ **states** |
| $A = a_{11} \ldots a_{ij} \ldots a_{NN}$ | a **transition probability matrix** $A$, each $a_{ij}$ representing the probability of moving from state $i$ to state $j$, s.t. $\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$ |
| $O = o_1 o_2 \ldots o_T$ | a sequence of $T$ **observations**, each one drawn from a vocabulary $V = v_1, v_2, ..., v_V$ |
| $B = b_i(o_t)$ | a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation $o_t$ being generated from a state $q_i$ |
| $\pi = \pi_1, \pi_2, ..., \pi_N$ | an **initial probability distribution** over states. $\pi_i$ is the probability that the Markov chain will start in state $i$. Some states $j$ may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$ |

# HMM example

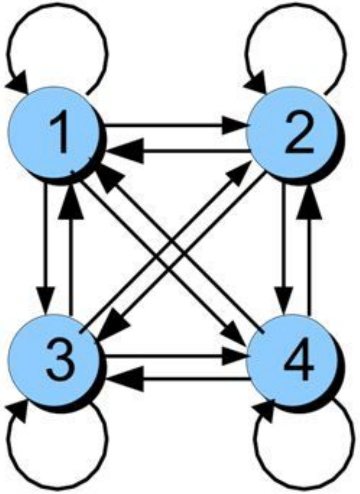


B2
P("aardvark" I MD)
...
P("will" I MD)
...
P("the" I MD)
...
P("back" | MD)
...
P("zebra" I MD)
a22
MD2
a12
a21
a32
a23
a11
a33
a13
a31
VB1
NN3
B1
P("aardvark" I VB)
...
P("will" I VB)
...
P("the" | VB)
...
P("back" I VB)
...
P("zebra" | VB)
B3
P("aardvark" I NN)
...
P("will" I NN)
...
P("the" I NN)
...
P("back" I NN)
...
P("zebra" I NN)

# Types of HMMs




Bakis = left-to-right




Ergodic =
fully-connected

- + many more

# HMMs in language technologies

- Part-of-speech tagging (Church, 1988; Brants, 2000)
- Named entity recognition (Bikel et al., 1999) and other information extraction tasks
- Text chunking and shallow parsing (Ramshaw and Marcus, 1995)
- Word alignment of parallel text (Vogel et al., 1996)
- Acoustic models in speech recognition (emissions are continuous)
- Discourse segmentation (labeling parts of a document)

# HMM parameters

| | |
|---|---|
| $Q = q_1 q_2 \ldots q_N$ | a set of $N$ **states** |
| $A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$ | a **transition probability matrix** $A$, each $a_{ij}$ representing the probability of moving from state $i$ to state $j$, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$ |
| $O = o_1 o_2 \ldots o_T$ | a sequence of $T$ **observations**, each one drawn from a vocabulary $V = v_1, v_2, ..., v_V$ |
| $B = b_i(o_t)$ | a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation $o_t$ being generated from a state $i$ |
| $q_0, q_F$ | a special **start state** and **end (final) state** that are not associated with observations, together with transition probabilities $a_{01} a_{02} \ldots a_{0n}$ out of the start state and $a_{1F} a_{2F} \ldots a_{nF}$ into the end state |

# HMMs: algorithms

| | | |
|---|---|---|
| Forward | **Problem 1 (Likelihood):** | Given an HMM $\lambda = (A, B)$ and an observation sequence $O$, determine the likelihood $P(O\|\lambda)$. |
| Viterbi | **Problem 2 (Decoding):** | Given an observation sequence $O$ and an HMM $\lambda = (A, B)$, discover the best hidden state sequence $Q$. |
| Forward–Backward; Baum–Welch | **Problem 3 (Learning):** | Given an observation sequence $O$ and the set of states in the HMM, learn the HMM parameters $A$ and $B$. |

# HMM tagging as decoding

**Decoding**: Given as input an HMM $\lambda = (A, B)$ and a sequence of observations $O = o_1, o_2, ..., o_T$, find the most probable sequence of states $Q = q_1 q_2 q_3 \ldots q_T$.

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} \overbrace{P(w_i | t_i)}^{\text{emission}} \overbrace{P(t_i | t_{i-1})}^{\text{transition}}$$

# HMM tagging as decoding

**Decoding**: Given as input an HMM $\lambda = (A, B)$ and a sequence of observations $O = o_1, o_2, ..., o_T$, find the most probable sequence of states $Q = q_1 q_2 q_3 \ldots q_T$.

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} \overbrace{P(w_i|t_i)}^{\text{emission}} \overbrace{P(t_i|t_{i-1})}^{\text{transition}}$$

How many possible choices?

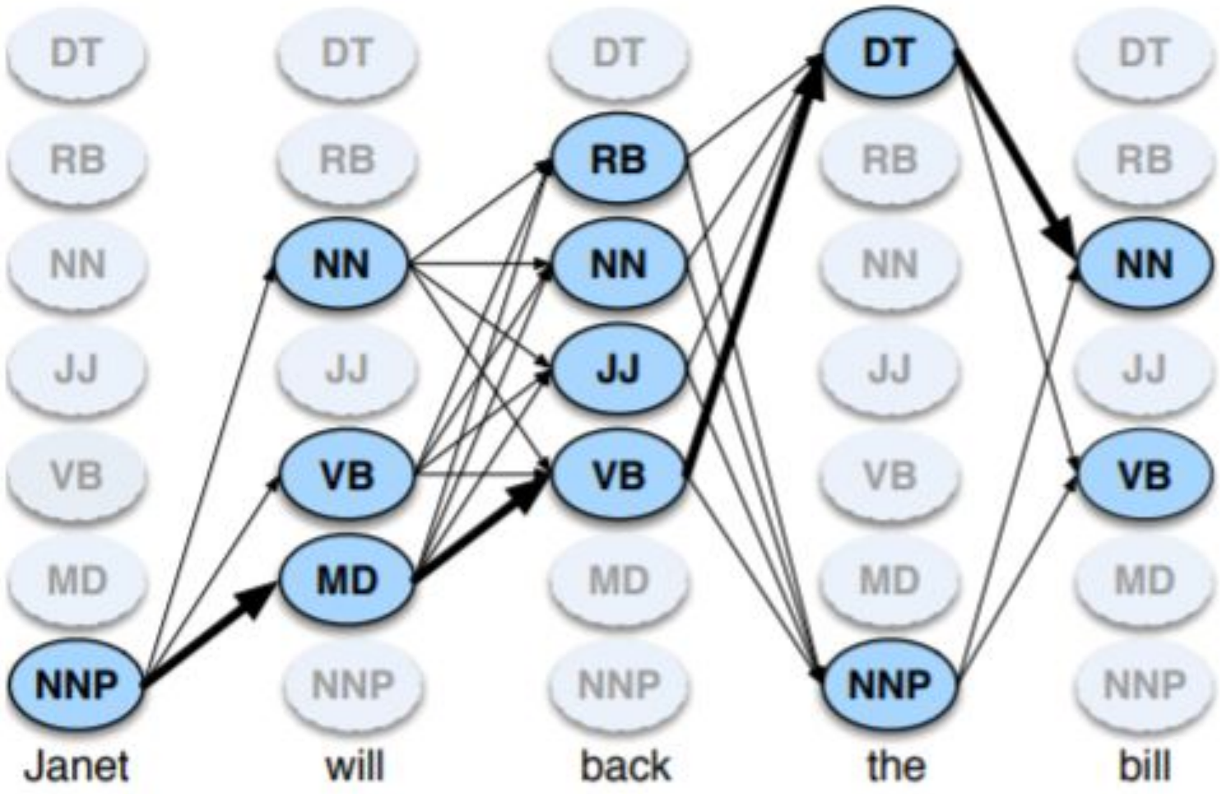
# Part of speech tagging example

| | I | suspect | the | present | forecast | is | pessimistic | . |
|---|---|---|---|---|---|---|---|---|
| noun | • | • | • | • | • | • | | |
| adj. | | • | | • | • | | • | |
| adv. | | | | • | | | | |
| verb | | • | | • | • | • | | |
| num. | • | | | | | | | |
| det. | | | • | | | | | |
| punc. | | | | | | | | • |

With this very simple tag set, $7^8 = 5.7$ million labelings.
(Even restricting to the possibilities above, 288 labelings.)

Slide credit: Noah Smith

# The Viterbi algorithm



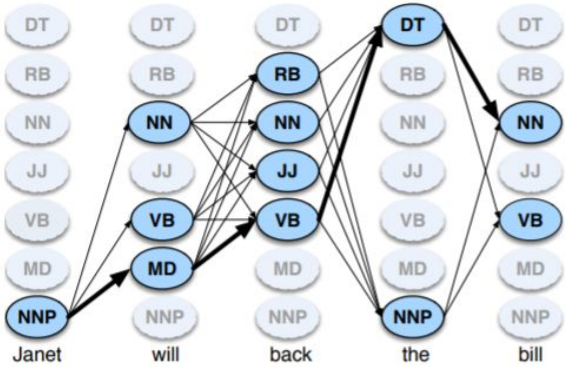
# The Viterbi algorithm

**function** VITERBI(*observations* of len *T*,*state-graph* of len *N*) **returns** *best-path*, *path-prob*

create a path probability matrix *viterbi[N,T]*
**for** each state *s* **from** 1 **to** *N* **do** ; initialization step
$viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
$backpointer[s,1] \leftarrow 0$
**for** each time step *t* **from** 2 **to** *T* **do** ; recursion step
**for** each state *s* **from** 1 **to** *N* **do**
$$viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$
$$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$ ; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$ ; termination step
*bestpath* ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
**return** *bestpath*, *bestpathprob*

Complexity?

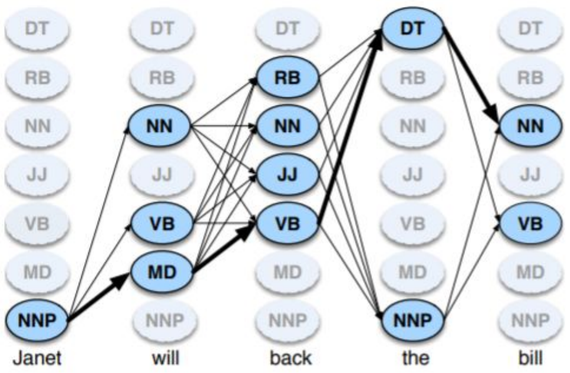
# The Viterbi algorithm




$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

| | |
|---|---|
| $v_{t-1}(i)$ | the **previous Viterbi path probability** from the previous time step |
| $a_{ij}$ | the **transition probability** from previous state $q_i$ to current state $q_j$ |
| $b_j(o_t)$ | the **state observation likelihood** of the observation symbol $o_t$ given the current state $j$ |

# The Viterbi algorithm




$$v_t(j) \; = \; \max_{i=1}^{N} v_{t-1}(i) \, a_{ij} \, b_j(o_t)$$

| | NNP | MD | VB | JJ | NN | RB | DT |
|---|---|---|---|---|---|---|---|
| <s> | 0.2767 | 0.0006 | 0.0031 | 0.0453 | 0.0449 | 0.0510 | 0.2026 |
| **NNP** | 0.3777 | 0.0110 | 0.0009 | 0.0084 | 0.0584 | 0.0090 | 0.0025 |
| **MD** | 0.0008 | 0.0002 | 0.7968 | 0.0005 | 0.0008 | 0.1698 | 0.0041 |
| **VB** | 0.0322 | 0.0005 | 0.0050 | 0.0837 | 0.0615 | 0.0514 | 0.2231 |
| **JJ** | 0.0366 | 0.0004 | 0.0001 | 0.0733 | 0.4509 | 0.0036 | 0.0036 |
| **NN** | 0.0096 | 0.0176 | 0.0014 | 0.0086 | 0.1216 | 0.0177 | 0.0068 |
| **RB** | 0.0068 | 0.0102 | 0.1011 | 0.1012 | 0.0120 | 0.0728 | 0.0479 |
| **DT** | 0.1147 | 0.0021 | 0.0002 | 0.2157 | 0.4744 | 0.0102 | 0.0017 |

| | Janet | will | back | the | bill |
|---|---|---|---|---|---|
| **NNP** | 0.000032 | 0 | 0 | 0.000048 | 0 |
| **MD** | 0 | 0.308431 | 0 | 0 | 0 |
| **VB** | 0 | 0.000028 | 0.000672 | 0 | 0.000028 |
| **JJ** | 0 | 0 | 0.000340 | 0 | 0 |
| **NN** | 0 | 0.000200 | 0.000223 | 0 | 0.002337 |
| **RB** | 0 | 0 | 0.010446 | 0 | 0 |
| **DT** | 0 | 0 | 0 | 0.506099 | 0 |

# The Viterbi algorithm

**function** VITERBI(*observations* of len $T$,*state-graph* of len $N$) **returns** *best-path*, *path-prob*

create a path probability matrix *viterbi[N,T]*

**for** each state $s$ **from** 1 **to** $N$ **do** ; initialization step

$\quad viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$

$\quad backpointer[s,1] \leftarrow 0$

**for** each time step $t$ **from** 2 **to** $T$ **do** ; recursion step

**for** each state $s$ **from** 1 **to** $N$ **do**

$$viterbi[s,t] \leftarrow \max_{s'=1}^{N} \; viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

$$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} \; viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

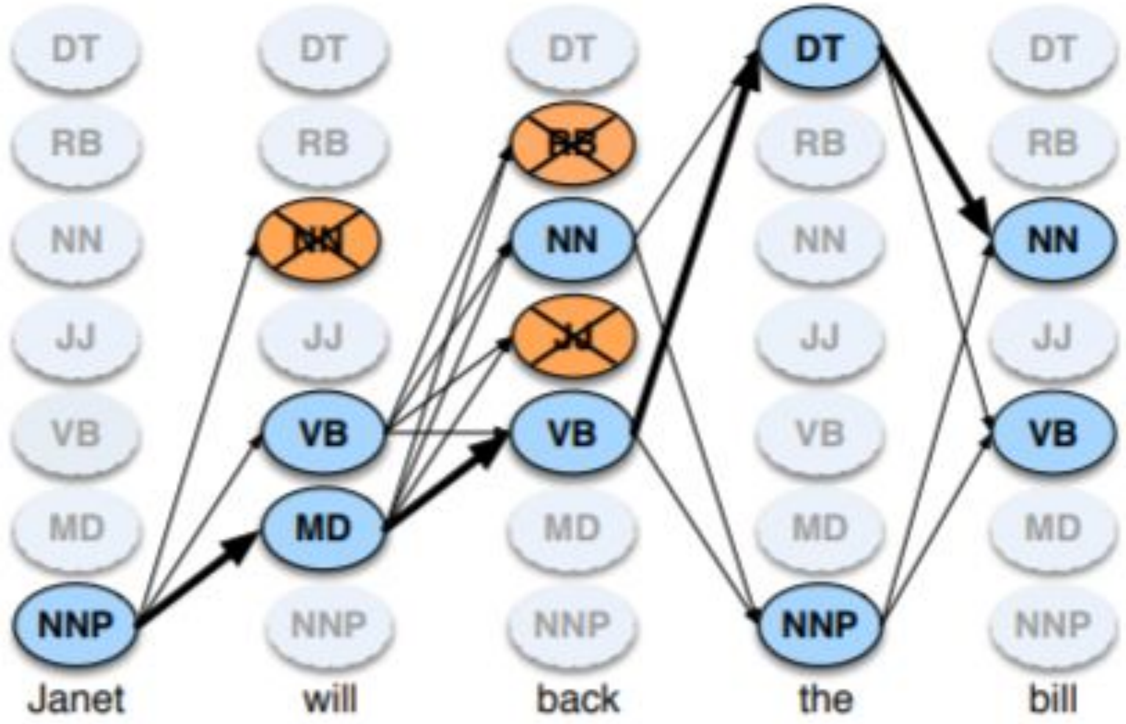
$bestpathprob \leftarrow \max_{s=1}^{N} \; viterbi[s,T]$ ; termination step

$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} \; viterbi[s,T]$ ; termination step

*bestpath* $\leftarrow$ the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time

**return** *bestpath*, *bestpathprob*

# Beam search

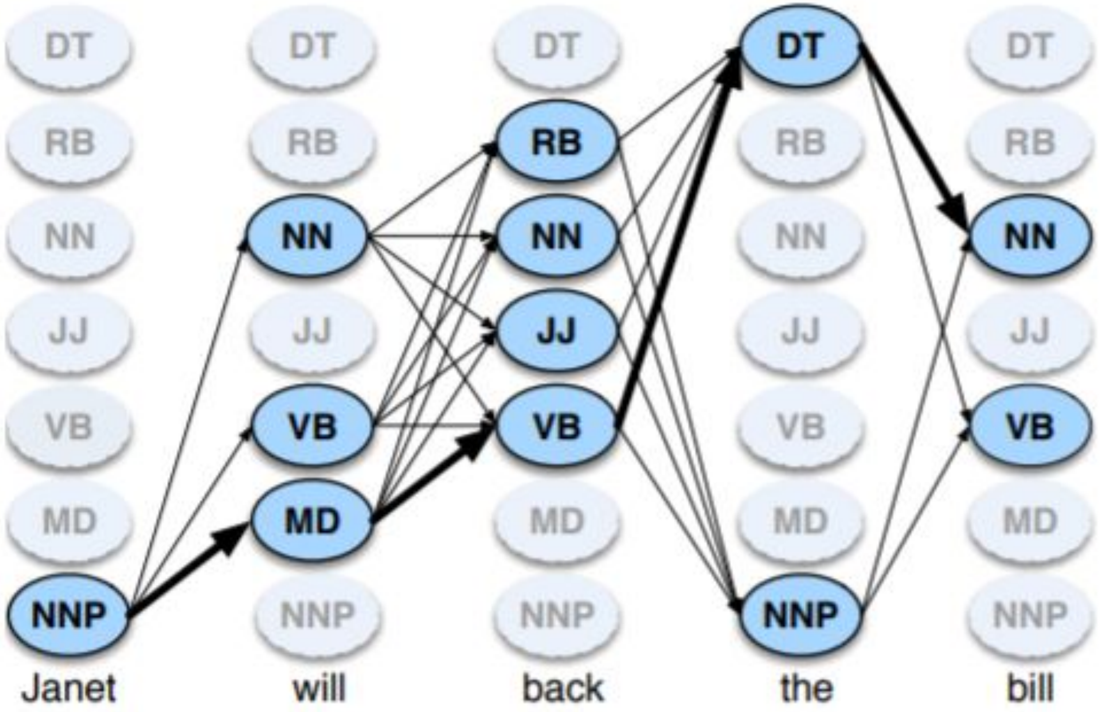


# HMMs: algorithms

| | | |
|---|---|---|
| **Forward** | **Problem 1 (Likelihood):** | Given an HMM $\lambda = (A, B)$ and an observation sequence $O$, determine the likelihood $P(O\|\lambda)$. |
| **Viterbi** | **Problem 2 (Decoding):** | Given an observation sequence $O$ and an HMM $\lambda = (A, B)$, discover the best hidden state sequence $Q$. |
| **Forward–Backward; Baum–Welch** | **Problem 3 (Learning):** | Given an observation sequence $O$ and the set of states in the HMM, learn the HMM parameters $A$ and $B$. |

From J&M

# The Forward algorithm




sum instead of max

# Viterbi

- n-best decoding
- relationship to sequence alignment

| Citation | Field |
| --- | --- |
| Viterbi (1967) | information theory |
| Vintsyuk (1968) | speech processing |
| Needleman and Wunsch (1970) | molecular biology |
| Sakoe and Chiba (1971) | speech processing |
| Sankoff (1972) | molecular biology |
| Reichert et al. (1973) | molecular biology |
| Wagner and Fischer (1974) | computer science |