Natural Language Processing

Sequence labeling

Yulia Tsvetkov

yuliats@cs.washington.edu
Announcements

- HW 2 review
Part of speech tagging

Part of Speech Tagger

- \( y_1 \): NOUN
- \( y_2 \): AUX
- \( y_3 \): VERB
- \( y_4 \): DET
- \( y_5 \): NOUN

Janet \( x_1 \)
will \( x_2 \)
back \( x_3 \)
the \( x_4 \)
bill \( x_5 \)
Why POS tagging

- Goal: resolve ambiguities
- Text-to-speech
  - record, lead, protest
- Lemmatization
  - saw/V → see, saw/N → saw
- Machine translation – the meaning of a word depends on its POS tag
  - “what is your address” vs. “we should address this comment”
  - “grand challenge” vs. “challenge the team”
- Sentiment analysis – adjectives are the major opinion holders
  - “good” vs. “bad”, “excellent” vs. “terrible”
- Preprocessing for harder disambiguation problems
  - syntactic parsing
  - semantic parsing
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)

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

- There are 7 children there.

There/PRON are/AUX 70/NUM children/NOUN there/ADV ./PUNC

- Preliminary findings were reported in today’s NY TIMES.

Preliminary/ADJ findings/NOUN were/AUX reported/VERB in/ADP today/NOUN ’s/PART NY/PROPN TIMES/PROPN ./PUNC
The 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
  - Greymatch maintained by Inria in Nancy
  - INESS maintained by the University of Bergen
- Download UD treebanks

<table>
<thead>
<tr>
<th>Open class words</th>
<th>Closed class words</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJ</td>
<td>ADP</td>
<td>PUNCT</td>
</tr>
<tr>
<td>ADV</td>
<td>AUX</td>
<td>SYM</td>
</tr>
<tr>
<td>INTJ</td>
<td>CCONJ</td>
<td>X</td>
</tr>
<tr>
<td>NOUN</td>
<td>DET</td>
<td></td>
</tr>
<tr>
<td>PROPN</td>
<td>NUM</td>
<td></td>
</tr>
<tr>
<td>VERB</td>
<td>PART</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRON</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SCONJ</td>
<td></td>
</tr>
</tbody>
</table>
# The Universal Dependencies

<table>
<thead>
<tr>
<th>tag</th>
<th>tag name</th>
<th>description</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJ</td>
<td>Adjective</td>
<td>noun modifiers describing properties</td>
<td>red, young, awesome</td>
</tr>
<tr>
<td>ADV</td>
<td>Adverb</td>
<td>verb modifiers of time, place, manner</td>
<td>very, slowly, home (as in “go home”), yesterday</td>
</tr>
<tr>
<td>NOUN</td>
<td>Noun</td>
<td>words for persons, places, things, etc.</td>
<td>algorithm, cat, mango, beauty</td>
</tr>
<tr>
<td>VERB</td>
<td>Verb</td>
<td>words for actions and processes</td>
<td>draw, provide, run</td>
</tr>
<tr>
<td>PROP</td>
<td>Proper noun</td>
<td>name of a person, organization, location, etc.</td>
<td>Regina, IBM, Vietnam</td>
</tr>
<tr>
<td>INTJ</td>
<td>Interjection</td>
<td>exclamation, greeting, yes/no response, etc.</td>
<td>oh, um, yes, hello</td>
</tr>
<tr>
<td>ADP</td>
<td>Adposition</td>
<td>(preposition/postposition): marks a noun’s spatial, temporal, or other relation</td>
<td>in, on, by, under</td>
</tr>
<tr>
<td>AUX</td>
<td>Auxiliary</td>
<td>helping a verb making tense, aspect, mood, etc.</td>
<td>can, may, should, are</td>
</tr>
<tr>
<td>CCONJ</td>
<td>Coordinating conjunction</td>
<td>joins two phrases or clauses</td>
<td>and, or, but</td>
</tr>
<tr>
<td>DET</td>
<td>Determiner</td>
<td>marks noun phrase properties</td>
<td>a, an, the, this</td>
</tr>
<tr>
<td>NUM</td>
<td>Numeral</td>
<td></td>
<td>one, two, first, second</td>
</tr>
<tr>
<td>PART</td>
<td>Particle</td>
<td>a preposition-like form used together with a verb</td>
<td>up, down, on, off, in, out, at, by</td>
</tr>
<tr>
<td>PRON</td>
<td>Pronoun</td>
<td>a shorthand for referring to an entity or event</td>
<td>you, she, who, I others</td>
</tr>
<tr>
<td>SCONJ</td>
<td>Subordinating conjunction</td>
<td>joins a main clause with a subordinate clause such as sentential complement</td>
<td>that, which</td>
</tr>
<tr>
<td>PUNCT</td>
<td>Punctuation</td>
<td></td>
<td>: ! ? ()</td>
</tr>
<tr>
<td>SYM</td>
<td>Symbol</td>
<td>various symbols</td>
<td>$ %</td>
</tr>
<tr>
<td>X</td>
<td>Other</td>
<td></td>
<td>all the rest...</td>
</tr>
</tbody>
</table>

**Universal Dependencies tagset (Nivre et al., 2016) (17 tags)**
## Ambiguities in POS tags

<table>
<thead>
<tr>
<th>Types</th>
<th>WSJ</th>
<th>Brown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unambiguous</td>
<td>44,432 (86%)</td>
<td>45,799 (85%)</td>
</tr>
<tr>
<td>Ambiguous</td>
<td>7,025 (14%)</td>
<td>8,050 (15%)</td>
</tr>
</tbody>
</table>

Yulia Tsvetkov
Ambiguities in POS tags

<table>
<thead>
<tr>
<th></th>
<th>WSJ</th>
<th>Brown</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Types:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unambiguous</td>
<td>44,432 (86%)</td>
<td>45,799 (85%)</td>
</tr>
<tr>
<td>Ambiguous</td>
<td>7,025 (14%)</td>
<td>8,050 (15%)</td>
</tr>
<tr>
<td><strong>Tokens:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unambiguous</td>
<td>577,421 (45%)</td>
<td>384,349 (33%)</td>
</tr>
<tr>
<td>Ambiguous</td>
<td>711,780 (55%)</td>
<td>786,646 (67%)</td>
</tr>
</tbody>
</table>
POS-tagging is a disambiguation task

- look at this small building in the back/NN
- the main player in the team took a back/JJ seat
- a majority of senators will back/VERB the decision
- we should enable the country to buy back/PART its debt
- I was twenty-one back/ADV then

- try yourself with an online POS tagger, e.g., https://parts-of-speech.info/
POS tagging algorithms

- We will introduce classic and state-of-the-art sequence labeling approaches
  - Hidden Markov Model (HMM) + Viterbi, generative
  - RNNs + Conditional Random Field (CRF), discriminative
Most frequent class baseline

- A baseline: assign each word its most probable (frequent) tag

- 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

\[ \hat{y}_i = \arg\max_{y \in \mathcal{L}} s(x, i, y) \]
Generative sequence labeling: Hidden Markov Models
Markov Chain: weather
Markov Chain: weather

- A model that assigns probabilities to sequences of random variables
  - Each variable can take a value from some finite set
- Markov assumption: only the current state matters for predicting the next state, all states before the current have no impact
- For a set of variables: $q_1, q_2, \ldots, q_i$

$$P(q_i = a|q_1, \ldots, q_{i-1}) = P(q_i = a|q_{i-1})$$
Markov Chain: weather

- Given initial probabilities of $[0.1, 0.7, 0.2]$ compute the probability for:
  - HOT WARM WARM HOT
  - HOT COLD HOT COLD
Markov chain

- Formally, a Markov chain specified by the following components

\[ Q = q_1 q_2 \ldots q_N \]
\[ A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn} \]
\[ \pi = \pi_1, \pi_2, \ldots, \pi_N \]

- A set of \( N \) states
- A transition probability matrix \( A \), each \( a_{ij} \) representing the probability of moving from state \( i \) to state \( j \), s.t.
  \[ \sum_{j=1}^{n} a_{ij} = 1 \quad \forall i \]
- 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: words

- only the current state matters for predicting the next state, all states before the current have no impact
Markov Chain: words

- only the current state matters for predicting the next state, all states before the current have no impact
Hidden Markov Models

- We use a Markov chain for computing $P$ for a sequence of observable events.
- In many cases the events we are interested in are hidden:
  - e.g., we don’t observe POS tags in a text.

![Diagram of Hidden Markov Model]

- $y_1$, $y_2$, $y_3$, $y_4$, $y_5$: hidden states
- $x_1$, $x_2$, $x_3$, $x_4$, $x_5$: observable states

Part of Speech Tagger

Janet, will, back, the, bill
Hidden Markov Models

Markov Assumption:  \[ P(q_i|q_1...q_{i-1}) = P(q_i|q_{i-1}) \]

Output Independence:  \[ P(o_i|q_1...q_i,...,q_T,o_1,...,o_i,...,o_T) = P(o_i|q_i) \]
Hidden Markov Models

\[ B_2 \]
- \( P("aardvark" \mid MD) \)
- \( P("will" \mid MD) \)
- \( P("the" \mid MD) \)
- \( P("back" \mid MD) \)
- \( P("zebra" \mid MD) \)

\[ B_1 \]
- \( P("aardvark" \mid VB) \)
- \( P("will" \mid VB) \)
- \( P("the" \mid VB) \)
- \( P("back" \mid VB) \)
- \( P("zebra" \mid VB) \)

\[ B_3 \]
- \( P("aardvark" \mid NN) \)
- \( P("will" \mid NN) \)
- \( P("the" \mid NN) \)
- \( P("back" \mid NN) \)
- \( P("zebra" \mid NN) \)
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. $\sum_{j=1}^{N} a_{ij} = 1 \ \forall i$
- $O = o_1 o_2 \ldots o_T$ a sequence of $T$ observations, each one drawn from a vocabulary $V = v_1, v_2, \ldots, v_V$
- $B = b_i(o_t)$ a sequence of observation likelihoods, also called emission probabilities, each expressing the probability of an observation $o_t$ being generated from a state $q_i$
- $\pi = \pi_1, \pi_2, \ldots, \pi_N$ 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, $\sum_{i=1}^{n} \pi_i = 1$
Types of HMMs

Bakis = left-to-right

Ergodic = fully-connected
Secretariat is expected to race tomorrow

Secretariat is expected to race tomorrow
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

- **$Q = q_1 q_2 \ldots q_N$** is a set of $N$ states
- **$A = a_{11} \ldots a_{ij} \ldots a_{NN}$** is a transition probability matrix $A$, each $a_{ij}$ representing the probability of moving from state $i$ to state $j$, s.t. $\sum_{j=1}^{N} a_{ij} = 1 \ \forall i$
- **$O = o_1 o_2 \ldots o_T$** is a sequence of $T$ observations, each one drawn from a vocabulary $V = v_1, v_2, \ldots, v_V$
- **$B = b_i(o_t)$** is a sequence of observation likelihoods, also called emission probabilities, each expressing the probability of an observation $o_t$ being generated from a state $q_i$
- **$\pi = \pi_1, \pi_2, \ldots, \pi_N$** is 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, $\sum_{i=1}^{n} \pi_i = 1$
HMMs: algorithms

Problem 1 (Likelihood): Given an HMM $\lambda = (A, B)$ and an observation sequence $O$, determine the likelihood $P(O|\lambda)$.

Problem 2 (Decoding): Given an observation sequence $O$ and an HMM $\lambda = (A, B)$, discover the best hidden state sequence $Q$.

Problem 3 (Learning): Given an observation sequence $O$ and the set of states in the HMM, learn the HMM parameters $A$ and $B$. 

Forward

Viterbi

Forward-backward; Baum-Welch
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 $B$. 
Decoding: Given as input an HMM $\lambda = (A, B)$ and sequence of observations $O = o_1, o_2, \ldots, o_n$, find the most probable sequence of states $Q = q_1, q_2, \ldots, q_n$

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(t_1^n \mid w_1^n)$$
Decoding: Given as input an HMM $\lambda = (A, B)$ and sequence of observations $O = o_1, o_2, \ldots, o_n$, find the most probable sequence of states $Q = q_1, q_2, \ldots, q_n$

$$
\hat{t}_1^n = \arg\max_{t_1^n} P(t_1^n | w_1^n)
= \arg\max_{t_1^n} \frac{P(w_1^n | t_1^n)P(t_1^n)}{P(w_1^n)}
$$
**Decoding**: Given as input an HMM $\lambda = (A, B)$ and sequence of observations $O = o_1, o_2, \ldots, o_n$, find the most probable sequence of states $Q = q_1, q_2, \ldots, q_n$.

\[
\hat{t}_1^n = \arg\max_{t_1^n} P(t_1^n \mid w_1^n)
\]

\[
= \arg\max_{t_1^n} \frac{P(w_1^n \mid t_1^n) P(t_1^n)}{P(w_1^n)}
\]

\[
= \arg\max_{t_1^n} P(w_1^n \mid t_1^n) P(t_1^n)
\]
HMM tagging as decoding

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

$$\hat{t}_1^n = \arg\max_{t_1^n} P(t_1^n | w_1^n)$$

$$= \arg\max_{t_1^n} \frac{P(w_1^n | t_1^n)P(t_1^n)}{P(w_1^n)}$$

$$= \arg\max_{t_1^n} P(w_1^n | t_1^n)P(t_1^n)$$

simplifying assumptions:
**Decoding:** Given as input an HMM $\lambda = (A, B)$ and sequence of observations $O = o_1, o_2, \ldots, o_n$, find the most probable sequence of states $Q = q_1, q_2, \ldots, q_n$

$$\hat{t}_1^n = \arg\max_{t_1^n} P(t_1^n | w_1^n)$$

$$= \arg\max_{t_1^n} \frac{P(w_1^n | t_1^n)P(t_1^n)}{P(w_1^n)}$$

$$= \arg\max_{t_1^n} P(w_1^n | t_1^n)P(t_1^n)$$

simplifying assumptions:

$$P(w_1^n | t_1^n) \approx \prod_{i=1}^{n} P(w_i | t_i)$$
**Decoding**: Given as input an HMM $\lambda = (A, B)$ and sequence of observations $O = o_1, o_2, ..., o_n$, find the most probable sequence of states $Q = q_1, q_2, ..., q_n$

$$\hat{t}_1^n = \arg\max_{t_1^n} P(t_1^n | w_1^n)$$

$$= \arg\max_{t_1^n} \frac{P(w_1^n | t_1^n)P(t_1^n)}{P(w_1^n)}$$

$$= \arg\max_{t_1^n} P(w_1^n | t_1^n)P(t_1^n)$$

simplifying assumptions:

$$P(w_1^n | t_1^n) \approx \prod_{i=1}^{n} P(w_i | t_i)$$

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i | t_{i-1})$$
HMM tagging as decoding

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

\[
\hat{t}_1^n = \arg\max_{t_1^n} P(t_1^n \mid w_1^n) = \arg\max_{t_1^n} \prod_{i=1}^{n} P(w_i \mid t_i) P(t_i \mid t_{i-1})
\]
HMM tagging as decoding

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

$$\hat{t}_1^n = \arg\max_{t_1^n} P(t_1^n | w_1^n) \approx \arg\max_{t_1^n} \prod_{i=1}^n P(w_i | t_i) P(t_i | t_{i-1})$$

How many possible choices?
Part of speech tagging example

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>suspect</th>
<th>the</th>
<th>present</th>
<th>forecast</th>
<th>is</th>
<th>pessimistic</th>
<th>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>noun</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td>.</td>
</tr>
<tr>
<td>adj.</td>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>adv.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>verb</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>num.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>det.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>punc.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
</tbody>
</table>

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

\[ v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t) \]
The Viterbi algorithm

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

- the previous Viterbi path probability from the previous time step
- the transition probability from previous state \( q_i \) to current state \( q_j \)
- the state observation likelihood of the observation symbol \( o_t \) given the current state \( j \)
The Viterbi algorithm

\[ \nu_t(j) = \max_{i=1}^{N} \nu_{t-1}(i) a_{ij} b_j(o_t) \]

- \( \nu_{t-1}(i) \) is the previous Viterbi path probability from the previous time step.
- \( a_{ij} \) is the transition probability from previous state \( q_i \) to current state \( q_j \).
- \( b_j(o_t) \) is the state observation likelihood of the observation symbol \( o_t \) given the current state \( j \).
The Viterbi algorithm

The previous Viterbi path probability from the previous time step
\( v_{t-1}(i) \)

The transition probability from previous state \( q_i \) to current state \( q_j \)
\( a_{ij} \)

The state observation likelihood of the observation symbol \( o_t \) given the current state \( j \)
\( b_j(o_t) \)

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

Previous Viterbi path probability

Transition probability

State observation likelihood
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
    viterbi[s,1] ← π_s * b_s(o_1)
    backpointer[s,1] ← 0

for each time step t from 2 to T do
    for each state s from 1 to N do
        viterbi[s,t] ← max_{s'} viterbi[s',t-1] * a_{s',s} * b_s(o_t)
        backpointer[s,t] ← argmax_{s'} viterbi[s',t-1] * a_{s',s} * b_s(o_t)

bestpathprob ← max_{s=1}^N viterbi[s,T]

bestpathpointer ← argmax_{s=1}^N viterbi[s,T]

bestpath ← the path starting at state bestpathpointer, that follows backpointer[] to states back in time

return bestpath, bestpathprob
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
    viterbi[s,1] ← \pi_s * b_s(o_1)
    backpointer[s,1] ← 0
for each time step t from 2 to T do
    for each state s from 1 to N do
        viterbi[s,t] ← \max_{s'} viterbi[s',t-1] * a_{s',s} * b_s(o_t)
        backpointer[s,t] ← \arg\max_{s'} viterbi[s',t-1] * a_{s',s} * b_s(o_t)
    bestpathprob ← \max_{s=1}^N viterbi[s,T]
    bestpathpointer ← \arg\max_{s=1}^N viterbi[s,T]
    bestpath ← the path starting at state bestpathpointer, that follows backpointer[] to states back in time
return bestpath, bestpathprob
```
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
  viterbi[s,1] ← π_s * b_s(o_1)
  backpointer[s,1] ← 0

for each time step t from 2 to T do
  for each state s from 1 to N do
    viterbi[s,t] ← \max_{s'}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)
    backpointer[s,t] ← arg\max_{s'}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)

bestpathprob ← \max_{s=1}^{N} viterbi[s,T]

bestpathpointer ← arg\max_{s=1}^{N} viterbi[s,T]

bestpath ← the path starting at state bestpathpointer, that follows backpointer[] to states back in time

return bestpath, bestpathprob
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
  $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
  $backpointer[s,1] \leftarrow 0$
for each time step $t$ from 2 to $T$ do
  for each state $s$ from 1 to $N$ do
    $viterbi[s,t] \leftarrow \max_{s' = 1}^{N} viterbi[s',t - 1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \arg\max_{s' = 1}^{N} viterbi[s',t - 1] * a_{s',s} * b_s(o_t)$
  $v_t(j) = \max_{i = 1}^{N} v_{t - 1}(i) a_{ij} b_j(o_t)$

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

bestpathpointer $\leftarrow \arg\max_{s = 1}^{N} viterbi[s,T]$

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

return bestpath, bestpathprob
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
    viterbi[s,1] ← π_s * b_s(o_1)
    backpointer[s,1] ← 0
for each time step t from 2 to T do
    for each state s from 1 to N do
        viterbi[s,t] ← \max_{s'} viterbi[s',t-1] * a_{s',s} * b_s(o_t)
        backpointer[s,t] ← \arg \max_{s'} viterbi[s',t-1] * a_{s',s} * b_s(o_t)
    bestpathprob ← \max_{s=1}^N viterbi[s,T]
    bestpathpointer ← \arg \max_{s=1}^N viterbi[s,T]
    bestpath ← the path starting at state bestpathpointer, that follows backpointer[] to states back in time
return bestpath, bestpathprob
```
The Viterbi algorithm

\[
v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)
\]
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
    viterbi[s, 1] ← πs * b_s(o_1)
    backpointer[s, 1] ← 0

for each time step t from 2 to T do
    for each state s from 1 to N do
        viterbi[s, t] ← \max_{s'} viterbi[s', t - 1] * a_{s',s} * b_s(o_t)
        backpointer[s, t] ← \arg\max_{s'} viterbi[s', t - 1] * a_{s',s} * b_s(o_t)

bestpathprob ← \max_{s=1}^{N} viterbi[s, T]

bestpathpointer ← \arg\max_{s=1}^{N} viterbi[s, T]

bestpath ← the path starting at state bestpathpointer, that follows backpointer[] to states back in time

return bestpath, bestpathprob

Computational complexity in N and T?
Beam search
HMMs: algorithms

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Problem 1 (Likelihood)</strong>:</td>
<td>Given an HMM $\lambda = (A, B)$ and an observation sequence $O$, determine the likelihood $P(O</td>
</tr>
<tr>
<td><strong>Problem 2 (Decoding)</strong>:</td>
<td>Given an observation sequence $O$ and an HMM $\lambda = (A, B)$, discover the best hidden state sequence $Q$.</td>
</tr>
<tr>
<td><strong>Problem 3 (Learning)</strong>:</td>
<td>Given an observation sequence $O$ and the set of states in the HMM, learn the HMM parameters $A$ and $B$.</td>
</tr>
</tbody>
</table>

From J&M
## HMMs: algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Problem 1 (Likelihood):</th>
<th>Problem 2 (Decoding):</th>
<th>Problem 3 (Learning):</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward</td>
<td>Given an HMM $\lambda = (A, B)$ and an observation sequence $O$, determine the likelihood $P(O</td>
<td>\lambda)$.</td>
<td>Given an observation sequence $O$ and an HMM $\lambda = (A, B)$, discover the best hidden state sequence $Q$.</td>
</tr>
<tr>
<td>Viterbi</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Forward-backward; Baum-Welch</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Forward algorithm

- Just sum instead of max!

\[ \alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t) \]
Viterbi

- n-best decoding
- relationship to sequence alignment

<table>
<thead>
<tr>
<th>Citation</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viterbi (1967)</td>
<td>information theory</td>
</tr>
<tr>
<td>Vintsyuk (1968)</td>
<td>speech processing</td>
</tr>
<tr>
<td>Needleman and Wunsch (1970)</td>
<td>molecular biology</td>
</tr>
<tr>
<td>Sakoe and Chiba (1971)</td>
<td>speech processing</td>
</tr>
<tr>
<td>Sankoff (1972)</td>
<td>molecular biology</td>
</tr>
<tr>
<td>Reichert et al. (1973)</td>
<td>molecular biology</td>
</tr>
<tr>
<td>Wagner and Fischer (1974)</td>
<td>computer science</td>
</tr>
</tbody>
</table>