Part-of-Speech Tagging

CMSC 723 / LING 723 / INST 725

MARINE CARPUAT
marine@cs.umd.edu
Today’s Agenda

• What are parts of speech (POS)?

• What is POS tagging?

• How to POS tag text automatically?
I LIKE TO VERB WORDS.

WHAT?

I TAKE NOUNS AND ADJECTIVES AND USE THEM AS VERBS. REMEMBER WHEN "ACCESS" WAS A THING? NOW IT'S SOMETHING YOU DO. IT GOT VERBED.

VERBING WEIRDS LANGUAGE.

MAYBE WE CAN EVENTUALLY MAKE LANGUAGE A COMPLETE IMPEDIMENT TO UNDERSTANDING.
Parts of Speech

• “Equivalence class” of linguistic entities
  – “Categories” or “types” of words
• Study dates back to the ancient Greeks
  – Dionysius Thrax of Alexandria (c. 100 BC)
  – 8 parts of speech: noun, verb, pronoun, preposition, adverb, conjunction, participle, article
  – Remarkably enduring list!
How do we define POS?

• By meaning
  – Verbs are actions
  – Adjectives are properties
  – Nouns are things

• By the syntactic environment
  – What occurs nearby?
  – What does it act as?

• By what morphological processes affect it
  – What affixes does it take?

• Combination of the above
Parts of Speech

• Open class
  – Impossible to completely enumerate
  – New words continuously being invented, borrowed, etc.

• Closed class
  – Closed, fixed membership
  – Reasonably easy to enumerate
  – Generally, short function words that “structure” sentences
Open Class POS

• Four major open classes in English
  – Nouns
  – Verbs
  – Adjectives
  – Adverbs

• All languages have nouns and verbs... but may not have the other two
Nouns

• Open class
  – New inventions all the time: muggle, webinar, ...

• Semantics:
  – Generally, words for people, places, things
  – But not always (bandwidth, energy, ...)

• Syntactic environment:
  – Occurring with determiners
  – Pluralizable, possessivizable

• Other characteristics:
  – Mass vs. count nouns
Verbs

• Open class
  – New inventions all the time: google, tweet, ...

• Semantics:
  – Generally, denote actions, processes, etc.

• Syntactic environment:
  – Intransitive, transitive, ditransitive
  – Alternations

• Other characteristics:
  – Main vs. auxiliary verbs
  – Gerunds (verbs behaving like nouns)
  – Participles (verbs behaving like adjectives)
Adjectives and Adverbs

• Adjectives
  – Generally modify nouns, e.g., tall girl

• Adverbs
  – A semantic and formal potpourri...
  – Sometimes modify verbs, e.g., sang beautifully
  – Sometimes modify adjectives, e.g., extremely hot
Closed Class POS

• Prepositions
  – In English, occurring before noun phrases
  – Specifying some type of relation (spatial, temporal, ...)
  – Examples: on the shelf, before noon

• Particles
  – Resembles a preposition, but used with a verb ("phrasal verbs")
  – Examples: find out, turn over, go on
Particle vs. Prepositions

He came *by* the office in a hurry  
(by = preposition)

He came *by* his fortune honestly  
(by = particle)

We ran *up* the phone bill  
(up = particle)

We ran *up* the small hill  
(up = preposition)

He lived *down* the block  
(down = preposition)

He never lived *down* the nicknames  
(down = particle)
More Closed Class POS

• Determiners
  – Establish reference for a noun
  – Examples: *a, an, the* (articles), *that, this, many, such, ...*

• Pronouns
  – Refer to person or entities: *he, she, it*
  – Possessive pronouns: *his, her, its*
  – Wh-pronouns: *what, who*
Closed Class POS: Conjunctions

• Coordinating conjunctions
  – Join two elements of “equal status”
  – Examples: cats and dogs, salad or soup

• Subordinating conjunctions
  – Join two elements of “unequal status”
  – Examples: We’ll leave after you finish eating. While I was waiting in line, I saw my friend.
  – Complementizers are a special case: I think that you should finish your assignment
Beyond English...

Chinese
No verb/adjective distinction! 漂亮: beautiful/to be beautiful

Riau Indonesian/Malay
No Articles
No Tense Marking
3rd person pronouns neutral to both gender and number
No features distinguishing verbs from nouns

Ayam (chicken) Makan (eat)
The chicken is eating
The chicken ate
The chicken will eat
The chicken is being eaten
Where the chicken is eating
How the chicken is eating
Somebody is eating the chicken
The chicken that is eating
Today’s Agenda

• What are parts of speech (POS)?

• What is POS tagging?

• How to POS tag text automatically?
POS Tagging: What’s the task?

• Process of assigning part-of-speech tags to words

• But what tags are we going to assign?
  – Coarse grained: noun, verb, adjective, adverb, ...
  – Fine grained: {proper, common} noun
  – Even finer-grained: {proper, common} noun ± animate

• Important issues to remember
  – Choice of tags encodes certain distinctions/non-distinctions
  – Tagsets will differ across languages!

• For English, Penn Treebank is the most common tagset
# Penn Treebank Tagset: 45 Tags

<table>
<thead>
<tr>
<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>coordin. conjunction</td>
<td>and, but, or</td>
<td>SYM</td>
<td>symbol</td>
<td>+, %, &amp;</td>
</tr>
<tr>
<td>CD</td>
<td>cardinal number</td>
<td>one, two, three</td>
<td>TO</td>
<td>“to”</td>
<td>to</td>
</tr>
<tr>
<td>DT</td>
<td>determiner</td>
<td>a, the</td>
<td>UH</td>
<td>interjection</td>
<td>ah, oops</td>
</tr>
<tr>
<td>EX</td>
<td>existential ‘there’</td>
<td>there</td>
<td>VB</td>
<td>verb, base form</td>
<td>eat</td>
</tr>
<tr>
<td>FW</td>
<td>foreign word</td>
<td>mea culpa</td>
<td>VBD</td>
<td>verb, past tense</td>
<td>ate</td>
</tr>
<tr>
<td>IN</td>
<td>preposition/sub-conj</td>
<td>of, in, by</td>
<td>VBG</td>
<td>verb, gerund</td>
<td>eating</td>
</tr>
<tr>
<td>JJ</td>
<td>adjective</td>
<td>yellow</td>
<td>VBN</td>
<td>verb, past participle</td>
<td>eaten</td>
</tr>
<tr>
<td>JJR</td>
<td>adj., comparative</td>
<td>bigger</td>
<td>VBP</td>
<td>verb, non-3sg pres</td>
<td>eat</td>
</tr>
<tr>
<td>JJS</td>
<td>adj., superlative</td>
<td>wildest</td>
<td>VBZ</td>
<td>verb, 3sg pres</td>
<td>eats</td>
</tr>
<tr>
<td>LS</td>
<td>list item marker</td>
<td>1, 2, One</td>
<td>WDT</td>
<td>wh-determiner</td>
<td>which, that</td>
</tr>
<tr>
<td>MD</td>
<td>modal</td>
<td>can, should</td>
<td>WP</td>
<td>wh-pronoun</td>
<td>what, who</td>
</tr>
<tr>
<td>NN</td>
<td>noun, sing. or mass</td>
<td>llama</td>
<td>WP$</td>
<td>possessive wh-</td>
<td>whose</td>
</tr>
<tr>
<td>NNS</td>
<td>noun, plural</td>
<td>llamas</td>
<td>WRB</td>
<td>wh-adverb</td>
<td>how, where</td>
</tr>
<tr>
<td>NNP</td>
<td>proper noun, singular</td>
<td>IBM</td>
<td>$</td>
<td>dollar sign</td>
<td>$</td>
</tr>
<tr>
<td>NNPS</td>
<td>proper noun, plural</td>
<td>Carolinas</td>
<td>#</td>
<td>pound sign</td>
<td>#</td>
</tr>
<tr>
<td>PDT</td>
<td>predeterminer</td>
<td>all, both</td>
<td>“</td>
<td>left quote</td>
<td>‘ or “</td>
</tr>
<tr>
<td>POS</td>
<td>possessive ending</td>
<td>’s</td>
<td>”</td>
<td>right quote</td>
<td>’ or ”</td>
</tr>
<tr>
<td>PRP</td>
<td>personal pronoun</td>
<td>I, you, he</td>
<td>(</td>
<td>left parenthesis</td>
<td>[, (, {, &lt;</td>
</tr>
<tr>
<td>PRPS</td>
<td>possessive pronoun</td>
<td>your, one’s</td>
<td>)</td>
<td>right parenthesis</td>
<td>], ), }, &gt;</td>
</tr>
<tr>
<td>RB</td>
<td>adverb</td>
<td>quickly, never</td>
<td>,</td>
<td>comma</td>
<td>,</td>
</tr>
<tr>
<td>RBR</td>
<td>adverb, comparative</td>
<td>faster</td>
<td>.</td>
<td>sentence-final punctuation</td>
<td>. ! ?</td>
</tr>
<tr>
<td>RBS</td>
<td>adverb, superlative</td>
<td>fastest</td>
<td>:</td>
<td>mid-sentence punctuation</td>
<td>: ; ... -- -</td>
</tr>
<tr>
<td>RP</td>
<td>particle</td>
<td>up, off</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Penn Treebank Tagset: Choices

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

• Distinctions and non-distinctions
  – Prepositions and subordinating conjunctions are tagged “IN” (“Although/IN I/PRP..”)
  – Except the preposition/complementizer “to” is tagged “TO”
Why do POS tagging?

• One of the most basic NLP tasks
  – Nicely illustrates principles of statistical NLP

• Useful for higher-level analysis
  – Needed for syntactic analysis
  – Needed for semantic analysis

• Sample applications that require POS tagging
  – Machine translation
  – Information extraction
  – Lots more...
Try your hand at tagging...

- The back door
- On my back
- Win the voters back
- Promised to back the bill
Try your hand at tagging...

• I hope **that** she wins
• **That** day was nice
• You can go **that** far
Why is POS tagging hard?

• Ambiguity!

  – Not just a lexical problem

  – Ambiguity in English
    • 11.5% of word types ambiguous in Brown corpus
    • 40% of word tokens ambiguous in Brown corpus
    • Annotator disagreement in Penn Treebank: 3.5%
Today’s Agenda

• What are parts of speech (POS)?

• What is POS tagging?

• How to POS tag text automatically?
POS tagging: how to do it?

• Given Penn Treebank, how would you build a system that can POS tag new text?

• Baseline: pick most frequent tag for each word type
  – 90% accuracy if train+test sets are drawn from Penn Treebank

• How can we do better?
Prediction problems

Given $x$, predict $y$

- **A book review**
  - Oh, man I love this book!
  - This book is so boring...
  - **Is it positive?**
    - yes
    - no

- **A tweet**
  - On the way to the park!
  - 公園に行くなう！
  - **Its language**
    - English
    - Japanese

- **A sentence**
  - I read a book
  - **Its syntactic parse**
    - S
      - VP
        - NP
          - I
          - read
          - a
          - book

- **Binary Prediction/Classification**
- **Multiclass Prediction/Classification**
- **Structured Prediction**
How can we POS tag automatically?

• POS tagging as multiclass classification
  – What is $x$? What is $y$?
  – What model and training algorithm can we use?
  – What kind of features can we use?

• POS tagging as sequence labeling
  – Models sequences of predictions
Hidden Markov Models

• Common approach to sequence labeling

• A finite state machine with probabilistic transitions

• Markov Assumption
  – next state only depends on the current state and independent of previous history
Hidden Markov Models (HMM) for POS tagging

• Probabilistic model for generating sequences
  – e.g., word sequences

• Assume
  – underlying set of hidden (unobserved) states in which
    the model can be (e.g., POS)
  – probabilistic transitions between states over time (e.g.,
    from POS to POS in order)
  – probabilistic generation of (observed) tokens from
    states (e.g., words generate for each POS)
HMM for POS tagging: intuition

Credit: Jordan Boyd Graber
HMM for POS tagging: intuition
HMM: Formal Specification

- **Q**: a finite set of $N$ states
  - $Q = \{q_0, q_1, q_2, q_3, \ldots\}$
- **$N \times N$ Transition probability matrix** $A = [a_{ij}]$
  - $a_{ij} = P(q_j|q_i), \Sigma a_{ij} = 1 \ \forall i$
- **Sequence of observations** $O = o_1, o_2, \ldots o_T$
  - Each drawn from a given set of symbols (vocabulary $V$)
- **$N \times |V|$ Emission probability matrix**, $B = [b_{it}]$
  - $b_{it} = b_i(o_t) = P(o_t|q_i), \Sigma b_{it} = 1 \ \forall i$
- **Start and end states**
  - An explicit start state $q_0$ or alternatively, a prior distribution over start states: $\{\pi_1, \pi_2, \pi_3, \ldots\}, \Sigma \pi_i = 1$
  - The set of final states: $q_F$
Let’s model the stock market...

Day: 1 2 3 4 5 6

Not observable!

Bull Bear S Bear S Bull

Here’s what you actually observe:

↑ ↓ ↔ ↑ ↓ ↔

↑: Market is up
↓: Market is down
↔: Market hasn’t changed

Credit: Jimmy Lin
Stock Market HMM

States?
Transitions?
Vocabulary?
Emissions?
Priors?

BEAR

BULL

STATIC
Stock Market HMM

States? ✓
Transitions? ✓
Vocabulary? 
Emissions? 
Priors?
Stock Market HMM

States? ✓
Transitions? ✓
Vocabulary? ✓
Emissions? 
Priors?

\[ V = \{ \uparrow, \downarrow, \leftrightarrow \} \]
Stock Market HMM

States? ✓
Transitions? ✓
Vocabulary? ✓
Emissions? ✓

P(↑ | Bear) = 0.1
P(↓ | Bear) = 0.6
P(↔ | Bear) = 0.3

P(↑ | Bull) = 0.7
P(↓ | Bull) = 0.1
P(↔ | Bull) = 0.2

P(↑ | Static) = 0.3
P(↓ | Static) = 0.3
P(↔ | Static) = 0.4

V = {↑, ↓, ↔}
Stock Market HMM

States? ✓
Transitions? ✓
Vocabulary? ✓
Emissions? ✓
Priors? ✓

\[
\begin{align*}
\pi_1 &= 0.5 \\
\pi_2 &= 0.2 \\
\pi_3 &= 0.3
\end{align*}
\]

\[
\begin{bmatrix}
P(\uparrow \mid \text{Bear}) = 0.1 \\
P(\downarrow \mid \text{Bear}) = 0.6 \\
P(\leftrightarrow \mid \text{Bear}) = 0.3
\end{bmatrix} \quad \begin{bmatrix}
P(\uparrow \mid \text{Bull}) = 0.7 \\
P(\downarrow \mid \text{Bull}) = 0.1 \\
P(\leftrightarrow \mid \text{Bull}) = 0.2
\end{bmatrix} \quad \begin{bmatrix}
P(\uparrow \mid \text{Static}) = 0.3 \\
P(\downarrow \mid \text{Static}) = 0.3 \\
P(\leftrightarrow \mid \text{Static}) = 0.4
\end{bmatrix}
\]

\[V = \{\uparrow, \downarrow, \leftrightarrow\}\]
Properties of HMMs

• The (first-order) Markov assumption holds

• The probability of an output symbol depends only on the state generating it

\[ P(o_t|q_1, q_2, \ldots, q_N, o_1, o_2, \ldots, o_T) = P(o_t|q_i) \]

• The number of states (N) does not have to equal the number of observations (T)
HMMs: Three Problems

• **Likelihood:** Given an HMM $\lambda = (A, B, \Pi)$, and a sequence of observed events $O$, find $P(O|\lambda)$

• **Decoding:** Given an HMM $\lambda = (A, B, \Pi)$, and an observation sequence $O$, find the most likely (hidden) state sequence

• **Learning:** Given a set of observation sequences and the set of states $Q$ in $\lambda$, compute the parameters $A$ and $B$
HMM Problem #1: Likelihood
Assuming $\lambda_{stock}$ models the stock market, how likely are we to observe the sequence of outputs?
Computing Likelihood

• First try:
  – Sum over all possible ways in which we could generate $O$ from $\lambda$

$$P(O|\lambda) = \sum_Q P(O, Q|\lambda) = \sum_Q P(O|Q, \lambda)P(Q|\lambda)$$

$$= \sum_{q_1, q_2 \ldots q_T} \pi_{q_1} b_{q_1}(o_1)a_{q_1 q_2} \ldots a_{q_{T-1} q_T} b_{q_T}(o_T)$$

Takes $O(N^T)$ time to compute!

– What’s the problem?

• Right idea, wrong algorithm!
Computing Likelihood

• What are we doing wrong?
  – State sequences may have a lot of overlap...
  – We’re recomputing the shared subsequences every time
  – Let’s store intermediate results and reuse them!
  – Can we do this?

• Sounds like a job for dynamic programming!
Forward Algorithm

• Use an $N \times T$ trellis or chart $[\alpha_{tj}]$

• Forward probabilities: $\alpha_{tj}$ or $\alpha_t(j)$
  – $= P($being in state $j$ after seeing $t$ observations$)$
  – $= P(o_1, o_2, ... \ o_t, q_t=j)$

• Each cell = $\sum$ extensions of all paths from other cells
  $\alpha_t(j) = \sum_i \alpha_{t-1}(i) \ a_{ij} \ b_j(o_t)$
  – $\alpha_{t-1}(i)$: forward path probability until $(t-1)$
  – $a_{ij}$: transition probability of going from state $i$ to $j$
  – $b_j(o_t)$: probability of emitting symbol $o_t$ in state $j$

• $P(O|\lambda) = \sum_i \alpha_T(i)$
Forward Algorithm: Formal Definition

- Initialization
  \[ \alpha_1(j) = \pi_j b_j(o_1), 1 \leq j \leq N \]

- Recursion
  \[ \alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t); 1 \leq j \leq N, 2 \leq t \leq T \]

- Termination
  \[ P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \]
Forward Algorithm

\[ O = \uparrow \downarrow \uparrow \]

find \( P(O | \lambda_{stock}) \)
Forward Algorithm

states

<table>
<thead>
<tr>
<th>Static</th>
<th>Bear</th>
<th>Bull</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image.png" alt="Diagram" /></td>
<td><img src="image.png" alt="Diagram" /></td>
<td><img src="image.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

- Static states at time $t=1$, $t=2$, and $t=3$.
- Bear states at time $t=1$, $t=2$, and $t=3$.
- Bull states at time $t=1$, $t=2$, and $t=3$.

The diagram illustrates the transitions between states over time.
Forward Algorithm: Initialization

\[ \alpha_1(j) = \pi_j b_j(o_1), 1 \leq j \leq N \]

- **Static** \( \alpha_1(\text{Static}) \):
  \[ 0.3 \times 0.3 = 0.09 \]

- **Bear** \( \alpha_1(\text{Bear}) \):
  \[ 0.5 \times 0.1 = 0.05 \]

- **Bull** \( \alpha_1(\text{Bull}) \):
  \[ 0.2 \times 0.7 = 0.14 \]

- **Time**:
  - t=1
  - t=2
  - t=3
Forward Algorithm: Recursion

\[ \alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t); 1 \leq j \leq N, 2 \leq t \leq T \]

Static

0.3 \times 0.3 = 0.09

Bears

0.5 \times 0.1 = 0.05

0.09 \times 0.4 \times 0.1 = 0.0036

0.05 \times 0.5 \times 0.1 = 0.0025

Bull

0.2 \times 0.7 = 0.14

0.14 \times 0.6 \times 0.1 = 0.0084

\[ \alpha_1(\text{Bull}) \times a_{\text{BullBull}} \times b_{\text{Bull}(\downarrow)} \]

.... and so on

t=1

\uparrow

t=2

\downarrow

t=3

\uparrow

time
Work through the rest of these numbers... 

Static

Bear

Bull

↑

= 0.09

= 0.05

0.2 × 0.7 = 0.14

0.5 × 0.1 = 0.05

0.3 × 0.3 = 0.09

0.0145

What’s the asymptotic complexity of this algorithm?
HMM Problem #2: Decoding
Decoding

Given $\lambda_{stock}$ as our model and $O$ as our observations, what are the most likely states the market went through to produce $O$?
Decoding

• “Decoding” because states are hidden

• First try:
  – Compute $P(O)$ for all possible state sequences, then choose sequence with highest probability
  – What’s the problem here?
Viterbi Algorithm

• “Decoding” = computing most likely state sequence
  – Another dynamic programming algorithm
  – Efficient: polynomial vs. exponential (brute force)

• Same idea as the forward algorithm
  – Store intermediate computation results in a trellis
  – Build new cells from existing cells
Viterbi Algorithm

• Use an $N \times T$ trellis $[v_{tj}]$
  – Just like in forward algorithm

• $v_{tj}$ or $v_t(j)$
  – $= P($in state $j$ after seeing $t$ observations and passing through the
    most likely state sequence so far$)$
  – $= P(q_1, q_2, \ldots q_{t-1}, q_{t=j}, o_1, o_2, \ldots o_t)$

• Each cell = extension of most likely path from other cells
  $v_t(j) = \max_i v_{t-1}(i) \cdot a_{ij} \cdot b_j(o_t)$
  – $v_{t-1}(i)$: Viterbi probability until $(t-1)$
  – $a_{ij}$: transition probability of going from state $i$ to $j$
  – $b_j(o_t)$: probability of emitting symbol $o_t$ in state $j$

• $P = \max_i v_T(i)$
Viterbi vs. Forward

• Maximization instead of summation over previous paths

• This algorithm is still missing something!
  – In forward algorithm, we only care about the probabilities
  – What’s different here?

• We need to store the most likely path (transition):
  – Use “backpointers” to keep track of most likely transition
  – At the end, follow the chain of backpointers to recover the most likely state sequence
Viterbi Algorithm: Formal Definition

- **Initialization**
  \[ v_1(j) = \pi_i b_i(o_1); 1 \leq i \leq N \]
  \[ BT_1(i) = 0 \]

- **Recursion**
  \[ v_t(j) = \max_{i=1}^{N} [v_{t-1}(i)a_{ij}] b_j(o_t); 1 \leq i \leq N, 2 \leq t \leq T \]
  \[ BT_1(i) = \arg \max_{i=1}^{N} [v_{t-1}(i)a_{ij}] \]

- **Termination**
  \[ P^* = \max_{1=1}^{N} v_T(j) \]
  \[ q_T^* = \arg \max_{1=i}^{N} v_T(j) \]
Viterbi Algorithm

\[ O = \uparrow \downarrow \uparrow \]

find most likely state sequence given \( \lambda_{stock} \)
Viterbi Algorithm

states

Static

Bear

Bull

↑
t=1

↓
t=2

↑
t=3

time
Viterbi Algorithm: Initialization

\[ v_1(j) = \pi_i b_i(o_1); 1 \leq i \leq N \]
\[ BT_1(i) = 0 \]

- **Static** \( \alpha_1(\text{Static}) \):
  \[ 0.3 \times 0.3 = 0.09 \]

- **Bear** \( \alpha_1(\text{Bear}) \):
  \[ 0.5 \times 0.1 = 0.05 \]

- **Bull** \( \alpha_1(\text{Bull}) \):
  \[ 0.2 \times 0.7 = 0.14 \]

- **Time**:
  - \( t=1 \)
  - \( t=2 \)
  - \( t=3 \)
Viterbi Algorithm: Recursion

\[ v_t(j) = \max_{i=1}^{N} [v_{t-1}(i) a_{ij}] b_j(o_t); \quad 1 \leq i \leq N, 2 \leq t \leq T \]

\[ BT_1(i) = \arg \max_{i=1}^{N} [v_{t-1}(i) a_{ij}] \]

Static

- 0.3 \times 0.3 = 0.09

Bear

- 0.5 \times 0.1 = 0.05

Bull

- 0.2 \times 0.7 = 0.14

Max

- 0.0084

States:

- t = 1
- t = 2
- t = 3

Time
Viterbi Algorithm: Recursion

\[ v_t(j) = \max_{i=1}^{N} [v_{t-1}(i) a_{ij}] b_j(o_t); 1 \leq i \leq N, 2 \leq t \leq T \]

\[ BT_1(i) = \arg \max_{i=1}^{N} [v_{t-1}(i) a_{ij}] \]

States:
- Static
- Bear
- Bull

Store backpointer

\[ 0.3 \times 0.3 = 0.09 \]
\[ 0.5 \times 0.1 = 0.05 \]
\[ 0.2 \times 0.7 = 0.14 \]
\[ 0.0084 \]

Time:
- \( t=1 \)
- \( t=2 \)
- \( t=3 \)

... and so on
Viterbi Algorithm: Recursion

Work through the rest of the algorithm...

<table>
<thead>
<tr>
<th>States</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static</td>
<td>t=1</td>
</tr>
<tr>
<td></td>
<td>t=2</td>
</tr>
<tr>
<td></td>
<td>t=3</td>
</tr>
<tr>
<td>Bear</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Bull</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Static:
  - 0.3 * 0.3 = 0.09

- Bear:
  - 0.5 * 0.1 = 0.05

- Bull:
  - 0.2 * 0.7 = 0.14
  - 0.0084

↑
→
↓
↑

- t=1
- t=2
- t=3
POS Tagging with HMMs
Modeling the problem

• What’s the problem?
  – The/DT grand/JJ jury/NN commmented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.

• What should the HMM look like?
  – States: part-of-speech tags \(t_1, t_2, ..., t_N\)
  – Output symbols: words \(w_1, w_2, ..., w_{|V|}\)
HMMs: Three Problems

- **Likelihood:** Given an HMM $\lambda = (A, B, \Pi)$, and a sequence of observed events $O$, find $P(O|\lambda)$

- **Decoding:** Given an HMM $\lambda = (A, B, \Pi)$, and an observation sequence $O$, find the most likely (hidden) state sequence

- **Learning:** Given a set of observation sequences and the set of states $Q$ in $\lambda$, compute the parameters $A$ and $B$
Today’s Agenda

• What are parts of speech (POS)?

• What is POS tagging?

• How to POS tag text automatically?
  – Sequence labeling problem
  – Decoding with Hidden Markov Models