Natural Language Processing
Sequence labeling

Sofia Serrano
sofias6@cs.washington.edu

Credit to Yulia Tsvetkov and Noah Smith for slides
Announcements

- A2 is out for < 10 more full days! Please start early!
- A1 grades will be released sometime today(?)
  - We’ll be accepting regrade requests for A1 for a week (Feb. 8 through Feb. 15)
- Quiz 4 will go out at 2:20pm today
  - 5 multiple-choice questions
  - Will cover lexical semantics, neural networks we’ve seen so far, and sequence labeling content up through the end of Monday’s lecture
  - Remember that you’re allowed to use your notes
  - From now on (including this quiz), you’ll have **15 minutes** to complete the quiz
  - From now on (including this quiz), quizzes will be available from 2:20pm on Wednesdays to 2:20pm on **Fridays**
Midterm eval feedback: Takeaways

What’s working:

● Lectures
● Asynchronous availability of slides/lecture recordings

What could use work:

● Quizzes– quiz-taking time window was the repeated issue that came up here
● Walking through equations in a bit more detail/using more visualizations when possible
● Computing setup instructions that assume less familiarity with git, etc.
Generative sequence labeling: Hidden Markov Models
Markov Chain: words

the future is independent of the past given the present

\[ \pi = [0.1, 0.7, 0.2] \]
Hidden Markov Models

- In the 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

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
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 (HMMs)

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

Secretariat is expected to race tomorrow.

Secretariat is expected to race tomorrow.
HMM parameters

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

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 \( Q \).

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

“What is the best sequence of tags (given the parameters of this HMM) that corresponds to ‘Janet will back the bill?’”
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 $Q = q_1, q_2, \ldots, q_n$.

$$\hat{t}_1^n = \arg\max_{t_1^n} P(t_1^n \mid w_1^n)$$
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 $Q = q_1, q_2, \ldots, q_n$

$$\hat{t}_1^n = \arg \max_{t^n_1} P(t^n_1 | w^n_1)$$

$$= \arg \max_{t^n_1} \frac{P(w^n_1 | t^n_1)P(t^n_1)}{P(w^n_1)}$$
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 $Q = q_1, q_2, \ldots, q_n$.

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

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

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

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

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

$$
P(t_1^n) \approx \prod_{i=1}^n P(t_i | t_{i-1})
$$
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 $Q = q_1, q_2, \ldots, q_n$

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

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

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

How many possible choices?
Could we brute force this?

(Imagine all these circles are colored in)

Janet will back the bill.
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
Building up the Viterbi algorithm

The best possible paths ending in any specific tag at position $i$

- Noun
- Adjective
- Verb
Building up the Viterbi algorithm

The best possible paths ending in any specific tag at position i

How do we use those to assemble the best possible paths ending in any specific tag at position i + 1?
Building up the Viterbi algorithm

The best possible paths ending in any specific tag at position i

Noun
Adjective
Verb

Noun
Adjective
Verb

How do we use those to assemble the best possible paths ending in any specific tag at position i + 1?
Building up the Viterbi algorithm

The best possible paths ending in any specific tag at position i

How do we use those to assemble the best possible paths ending in any specific tag at position i + 1?
Building up the Viterbi algorithm

The best possible paths ending in any specific tag at position $i$

- Noun
- Adjective
- Verb

How do we use those to assemble the best possible paths ending in any specific tag at position $i + 1$?
Building up the Viterbi algorithm

The best possible paths ending in any specific tag at position i

How do we use those to assemble the best possible paths ending in any specific tag at position i + 1?
Building up the Viterbi algorithm

The best possible paths ending in any specific tag at position $i$

- Pick the max of those three quantities (one per path from the previous timestep)
Building up the Viterbi algorithm

The best possible paths ending in any specific tag at position \( i \)

- Noun
- Adjective
- Verb

Pick the max of those three quantities (one per path from the previous timestep)

Store:
- That max score
- AND the preceding tag that was used to produce that max score (“backpointer”)
The Viterbi algorithm

$v_{t-1}(i)$ 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 $j$

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

- $v_{t-1}(i)$: 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 $j$

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

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

- \( v_{t-1}(i) \) 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

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

- \( v_{t-1}(i) \): 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

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

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

● Transitioning to an end state
  ○ Not all POS tags are equally likely to end a sequence! (For example, DET (corresponding to words like “the”))
  ○ Our algorithm should account for that!
  ○ Solution? Multiply candidate best paths at last time step by \( p(<\text{EOS}> \mid \text{tag}) \)

● Assembling your best tag sequence using that max score at the last timestep
  ○ Here’s where we use our backpointers!
  ○ Just trace backwards by using the backpointers (that extra piece of information that we stored about how we got to our current tag at timestep \( i + 1 \) :)
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
  \( viterbi[s,1] \leftarrow \pi_s \times b_s(o_1) \)
  backpointer\([s,1] \leftarrow 0 \)

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

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

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

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

return bestpath, bestpathprob

Computational complexity in \( N \) and \( T \)?
HMMs: algorithms

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

“What is the best sequence of tags (given the parameters of this HMM) that corresponds to ‘Janet will back the bill?’”

From J&M
### 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 $B$. |

“What is the probability (given the parameters of this HMM) of observing the text ‘Janet will back the bill?’”
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>