Maximum Entropy Markov Models (log-linear model for tagging)

Instructor: Wei Xu

Many slides from Michael Collins and Yejin Choi
Part-of-Speech Tagging

INPUT:
Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

OUTPUT:
Profits/N soared/V at/P Boeing/N Co./N ,/, easily/ADV topping/V forecasts/N on/P Wall/N Street/N ,/, as/P their/POSS CEO/N Alan/N Mulally/N announced/V first/ADJ quarter/N results/N ./.
Named Entity Recognition

**INPUT:** Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

**OUTPUT:** Profits soared at [Company Boeing Co.], easily topping forecasts on [Location Wall Street], as their CEO [Person Alan Mulally] announced first quarter results.
Named Entity Extraction as Tagging

INPUT:
Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

OUTPUT:
Profits/NA soared/NA at/NA Boeing/SC Co./CC ,/NA easily/NA topping/NA forecasts/NA on/NA Wall/SL Street/CL ,/NA as/NA their/NA CEO/NA Alan/SP Mulally/CP announced/NA first/NA quarter/NA results/NA ./NA

NA = No entity
SC = Start Company
CC = Continue Company
SL = Start Location
CL = Continue Location
Our Goal

Training set:
1 Pierre/NNP Vinken/NNP, 61/CD years/NNS old/JJ, will/MD join/VB the/DT board/NN as/IN a/DT nonexecutive/JJ director/NN Nov./NNP 29/CD.
2 Mr./NNP Vinken/NNP is/VBZ chairman/NN of/IN Elsevier/NNP N.V./NNP, the/DT Dutch/NNP publishing/VBG group/NN.
3 Rudolph/NNP Agnew/NNP, 55/CD years/NNS old/JJ and/CC chairman/NN of/IN Consolidated/NNP Gold/NNP Fields/NNP PLC/NNP, was/VBD named/VBN a/DT nonexecutive/JJ director/NN of/IN this/DT British/JJ industrial/JJ conglomerate/NN.

... 38,219 It/PRP is/VBZ also/RB pulling/VBG 20/CD people/NNS out/IN of/IN Puerto/NNP Rico/NNP, who/WP were/VBD helping/VBG Hurricane/NNP Hugo/NNP victims/NNS, and/CC sending/VBG them/PRP to/TO San/NNP Francisco/NNP instead/RB.

From the training set, induce a function/algorithim that maps new sentences to their tag sequences.
Overview

- Recap: The Tagging Problem
- Log-linear taggers
Tagging (Sequence Labeling)

- Given a sequence (in NLP, words), assign appropriate labels to each word.
- Many NLP problems can be viewed as sequence labeling:
  - POS Tagging
  - Chunking
  - Named Entity Tagging
- Labels of tokens are dependent on the labels of other tokens in the sequence, particularly their neighbors

Plays well with others.

VBZ   RB   IN   NNS
Log-Linear Models for Tagging

- We have an input sentence $w_{[1:n]} = w_1, w_2, \ldots, w_n$
  ($w_i$ is the $i$'th word in the sentence)
Log-Linear Models for Tagging

- We have an input sentence $w_{[1:n]} = w_1, w_2, \ldots, w_n$
  ($w_i$ is the $i$’th word in the sentence)

- We have a tag sequence $t_{[1:n]} = t_1, t_2, \ldots, t_n$
  ($t_i$ is the $i$’th tag in the sentence)
Log-Linear Models for Tagging

- We have an input sentence \( w_{[1:n]} = w_1, w_2, \ldots, w_n \)
  \((w_i \text{ is the } i^{\text{th}} \text{ word in the sentence})\)

- We have a tag sequence \( t_{[1:n]} = t_1, t_2, \ldots, t_n \)
  \((t_i \text{ is the } i^{\text{th}} \text{ tag in the sentence})\)

- We'll use an log-linear model to define
  \[
p(t_1, t_2, \ldots, t_n | w_1, w_2, \ldots, w_n)
  \]
  for any sentence \( w_{[1:n]} \) and tag sequence \( t_{[1:n]} \) of the same length.
  \((\text{Note: contrast with HMM that defines } p(t_1 \ldots t_n, w_1 \ldots w_n))\)
Log-Linear Models for Tagging

- We have an input sentence $w_{[1:n]} = w_1, w_2, \ldots, w_n$
  ($w_i$ is the $i$'th word in the sentence)

- We have a tag sequence $t_{[1:n]} = t_1, t_2, \ldots, t_n$
  ($t_i$ is the $i$'th tag in the sentence)

- We'll use an log-linear model to define

  $$p(t_1, t_2, \ldots, t_n | w_1, w_2, \ldots, w_n)$$

  for any sentence $w_{[1:n]}$ and tag sequence $t_{[1:n]}$ of the same length.
  (Note: contrast with HMM that defines $p(t_1 \ldots t_n, w_1 \ldots w_n)$)

- Then the most likely tag sequence for $w_{[1:n]}$ is

  $$t_{[1:n]}^* = \text{argmax}_{t_{[1:n]}} p(t_{[1:n]} | w_{[1:n]})$$
How to model $p(t_{[1:n]}|w_{[1:n]})$?

A Trigram Log-Linear Tagger:

$$p(t_{[1:n]}|w_{[1:n]}) = \prod_{j=1}^{n} p(t_j | w_1 \ldots w_n, t_1 \ldots t_{j-1})$$

Chain rule
How to model \( p(t_{[1:n]} | w_{[1:n]}) \)?

**A Trigram Log-Linear Tagger:**

\[
p(t_{[1:n]} | w_{[1:n]}) = \prod_{j=1}^{n} p(t_j | w_1 \ldots w_n, t_1 \ldots t_{j-1})
\]

Chain rule

\[
= \prod_{j=1}^{n} p(t_j | w_1, \ldots, w_n, t_{j-2}, t_{j-1})
\]

Independence assumptions

- We take \( t_0 = t_{-1} = * \)
How to model $p(t_{[1:n]}|w_{[1:n]})$?

A Trigram Log-Linear Tagger:

$$p(t_{[1:n]}|w_{[1:n]}) = \prod_{j=1}^{n} p(t_j | w_1 \ldots w_n, t_{1} \ldots t_{j-1})$$

Chain rule

$$= \prod_{j=1}^{n} p(t_j | w_1, \ldots, w_n, t_{j-2}, t_{j-1})$$

Independence assumptions

- We take $t_0 = t_{-1} = *$

- Independence assumption: each tag only depends on previous two tags

$$p(t_j|w_1, \ldots, w_n, t_1, \ldots, t_{j-1}) = p(t_j|w_1, \ldots, w_n, t_{j-2}, t_{j-1})$$
An Example

Hispaniola/NNP quickly/RB became/VB an/DT important/JJ base/?? from which Spain expanded its empire into the rest of the Western Hemisphere.

- There are many possible tags in the position ??
  \[ \mathcal{Y} = \{ \text{NN, NNS, Vt, Vi, IN, DT, ...} \} \]
Representation: Histories

- A **history** is a 4-tuple \( \langle t_{-2}, t_{-1}, w_{[1:n]}, i \rangle \)
- \( t_{-2}, t_{-1} \) are the previous two tags.
- \( w_{[1:n]} \) are the \( n \) words in the input sentence.
- \( i \) is the index of the word being tagged
- \( \mathcal{X} \) is the set of all possible histories

Hispaniola/NNP quickly/RB became/VB an/DTD important/JJ base/?? from which Spain expanded its empire into the rest of the Western Hemisphere.

- \( t_{-2}, t_{-1} = DT, JJ \)
- \( w_{[1:n]} = \langle Hispaniola, quickly, became, \ldots, Hemisphere, . \rangle \)
- \( i = 6 \)
Recap: Feature Vector Representations in Log-Linear Models

- We have some input domain \( \mathcal{X} \), and a finite label set \( \mathcal{Y} \). Aim is to provide a conditional probability \( p(y \mid x) \) for any \( x \in \mathcal{X} \) and \( y \in \mathcal{Y} \).

- A **feature** is a function \( f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R} \) (Often **binary features** or **indicator functions** \( f : \mathcal{X} \times \mathcal{Y} \to \{0, 1\} \)).

- Say we have \( m \) features \( f_k \) for \( k = 1 \ldots m \) ⇒ A **feature vector** \( f(x, y) \in \mathbb{R}^m \) for any \( x \in \mathcal{X} \) and \( y \in \mathcal{Y} \).
An Example (continued)

- $\mathcal{X}$ is the set of all possible histories of form $\langle t_{-2}, t_{-1}, w_{1:n}, i \rangle$
- $\mathcal{Y} = \{\text{NN, NNS, Vt, Vi, IN, DT, \ldots}\}$
- We have $m$ features $f_k : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ for $k = 1 \ldots m$

For example:

\[
f_1(h, t) = \begin{cases} 
1 & \text{if current word } w_i \text{ is base and } t = \text{Vt} \\
0 & \text{otherwise}
\end{cases}
\]

\[
f_2(h, t) = \begin{cases} 
1 & \text{if current word } w_i \text{ ends in } \text{ing} \text{ and } t = \text{VBG} \\
0 & \text{otherwise}
\end{cases}
\]

\[
f_1(\langle \text{JJ, DT, } \langle \text{Hispaniola, } \ldots, \rangle, 6 \rangle, \text{Vt}) = 1
\]

\[
f_2(\langle \text{JJ, DT, } \langle \text{Hispaniola, } \ldots, \rangle, 6 \rangle, \text{Vt}) = 0
\]

- Analogy to $e(\text{base} \mid \text{Vt})$ in HMMs
- Difficult for HMMs

Hispaniola/NNP quickly/RB became/VB an/DT important/JJ base/?? from which Spain expanded its empire into the rest of the Western Hemisphere.
Log-Linear Models

- We have some input domain $\mathcal{X}$, and a finite label set $\mathcal{Y}$. Aim is to provide a conditional probability $p(y \mid x)$ for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.

- A feature is a function $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ (Often binary features or indicator functions $f : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}$).

- Say we have $m$ features $f_k$ for $k = 1 \ldots m$ ⇒ A feature vector $f(x, y) \in \mathbb{R}^m$ for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.

- We also have a parameter vector $v \in \mathbb{R}^m$

- We define

\[
p(y \mid x; v) = \frac{e^{v \cdot f(x, y)}}{\sum_{y' \in \mathcal{Y}} e^{v \cdot f(x, y')}}
\]
Training the Log-Linear Model

- To train a log-linear model, we need a training set \((x_i, y_i)\) for \(i = 1 \ldots n\). Then search for

\[
\nu^* = \arg\max_{\nu} \left( \sum_i \log p(y_i | x_i; \nu) - \frac{\lambda}{2} \sum_k \nu_k^2 \right)
\]

- Training set is simply all history/tag pairs seen in the training data
The Viterbi Algorithm

Problem: for an input \( w_1 \ldots w_n \), find

\[
\arg \max_{t_1 \ldots t_n} p(t_1 \ldots t_n | w_1 \ldots w_n)
\]

We assume that \( p \) takes the form

\[
p(t_1 \ldots t_n | w_1 \ldots w_n) = \prod_{i=1}^{n} q(t_i | t_{i-2}, t_{i-1}, w_{[1:n]}, i)
\]

(In our case \( q(t_i | t_{i-2}, t_{i-1}, w_{[1:n]}, i) \) is the estimate from a log-linear model.)
The Viterbi Algorithm

- Define $n$ to be the length of the sentence
- Define
  \[ r(t_1 \ldots t_k) = \prod_{i=1}^{k} q(t_i | t_{i-2}, t_{i-1}, w_{[1:n]}, i) \]
- Define a dynamic programming table
  \[ \pi(k, u, v) = \text{maximum probability of a tag sequence ending} \]
  \[ \text{in tags } u, v \text{ at position } k \]
  that is,
  \[ \pi(k, u, v) = \max_{\{t_1, \ldots, t_{k-2}\}} r(t_1 \ldots t_{k-2}, u, v) \]
A Recursive Definition

Base case:

\[ \pi(0, *, *) = 1 \]

Recursive definition:
For any \( k \in \{1 \ldots n\} \), for any \( u \in S_{k-1} \) and \( v \in S_k \):

\[ \pi(k, u, v) = \max_{t \in S_{k-2}} \left( \pi(k - 1, t, u) \times q(v|t, u, w_{1:n}, k) \right) \]

where \( S_k \) is the set of possible tags at position \( k \)
The Viterbi Algorithm with Backpointers

Input: a sentence $w_1 \ldots w_n$, log-linear model that provides $q(v|t, u, w_{[1:n]}, i)$ for any tag-trigram $t, u, v$, for any $i \in \{1 \ldots n\}$

Initialization: Set $\pi(0, *, *) = 1$.

Algorithm:

- For $k = 1 \ldots n$,
  - For $u \in S_{k-1}$, $v \in S_k$,
    \[
    \pi(k, u, v) = \max_{t \in S_{k-2}} \left( \pi(k - 1, t, u) \times q(v|t, u, w_{[1:n]}, k) \right)
    \]
    \[
    bp(k, u, v) = \arg \max_{t \in S_{k-2}} \left( \pi(k - 1, t, u) \times q(v|t, u, w_{[1:n]}, k) \right)
    \]
  - Set $(t_{n-1}, t_n) = \arg \max_{(u, v)} \pi(n, u, v)$
  - For $k = (n - 2) \ldots 1$, $t_k = bp(k + 2, t_{k+1}, t_{k+2})$
  - Return the tag sequence $t_1 \ldots t_n$
The HMM State Lattice / Trellis

```
The HMM State Lattice / Trellis
```

```
START       Fed           raises       interest       rates       STOP
```

```
e(Fed|N)
e(raises|V)
e(interest|V)
e(rates|J)
e(STOP|V)
```

```
q(N|V)
q(V|N)
q(V|V)
q(J|V)
q(V|J)
```
The MEMM State Lattice / Trellis

x = START       Fed           raises       interest         rates         STOP
The perceptron algorithm
- Iteratively processes the training set, reacting to training errors
- Can be thought of as trying to drive down training error

The (online) perceptron algorithm:
- Start with zero weights
- Visit training instances \((x_i, y_i)\) one by one
  - Make a prediction
    \[
    y^* = \arg \max_y w \cdot \phi(x_i, y)
    \]
    - If correct \((y^* = y_i)\): no change, goto next example!
    - If wrong: adjust weights
      \[
      w = w + \phi(x_i, y_i) - \phi(x_i, y^*)
      \]

Challenge: How to compute argmax efficiently?
The Perceptron State Lattice / Trellis

\[ w \cdot \Phi(x, 1, N, J) + w \cdot \Phi(x, 2, V, N) + w \cdot \Phi(x, 3, V, V) + w \cdot \Phi(x, 4, J, J) + w \cdot \Phi(x, 5, V, J) \]

x = START Fed raises interest rates STOP
Decoding

- **Linear Perceptron**  
  \[ s^* = \arg \max_s w \cdot \Phi(x, s) \]
  - Features must be local, for \( x = x_1 \ldots x_m \), and \( s = s_1 \ldots s_m \)
  \[ \Phi(x, s) = \sum_{j=1}^{m} \phi(x, j, s_{j-1}, s_j) \]
  - Define \( \pi(i, s_i) \) to be the max score of a sequence of length \( i \) ending in tag \( s_i \)
    \[ \pi(i, s_i) = \max_{s_{i-1}} w \cdot \phi(x, i, s_{i-1}, s_i) + \pi(i - 1, s_{i-1}) \]

- **Viterbi algorithm (HMMs):**
  \[ \pi(i, s_i) = \max_{s_{i-1}} e(x_i | s_i) q(s_i | s_{i-1}) \pi(i - 1, s_{i-1}) \]

- **Viterbi algorithm (Maxent):**
  \[ \pi(i, s_i) = \max_{s_{i-1}} p(s_i | s_{i-1}, x_1 \ldots x_m) \pi(i - 1, s_{i-1}) \]
FAQ Segmentation: McCallum et. al

- McCallum et. al compared HMM and log-linear taggers on a FAQ Segmentation task

- Main point: in an HMM, modeling

  \[ p(\text{word}|\text{tag}) \]

  is difficult in this domain
FAQ Segmentation: McCallum et. al

X-NNTP-POSTER: NewsHound v1.33
Archive name: acorn/faq/part2
Frequency: monthly

2.6) What configuration of serial cable should I use

Here follows a diagram of the necessary connections programs to work properly. They are as far as I know that agreed upon by commercial comms software developers for

Pins 1, 4, and 8 must be connected together inside
is to avoid the well known serial port chip bugs. The
FAQ Segmentation: Line Features

- begins-with-number
- begins-with-ordinal
- begins-with-punctuation
- begins-with-question-word
- begins-with-subject
- blank
- contains-alphanum
- contains-bracketed-number
- contains-http
- contains-non-space
- contains-number
- contains-pipe
- contains-question-mark
- ends-with-question-mark
- first-alpha-is-capitalized
- indented-1-to-4
FAQ Segmentation: The Log-Linear Tagger

X-NNTP-POSTER: Newshound v1.33

Archive name: acorn/faq/part2

Frequency: monthly

2.6) What configuration of serial cable should I use

Here follows a diagram of the necessary connections

⇒ “tag=question;prev=head;begins-with-number”
  “tag=question;prev=head;contains-alphanum”
  “tag=question;prev=head;contains-nonspace”
  “tag=question;prev=head;contains-number”
  “tag=question;prev=head;prev-is-blank”
FAQ Segmentation: An HMM Tagger

2.6) What configuration of serial cable should I use

- First solution for $p(\text{word} \mid \text{tag})$:

$$p(\text{"2.6) What configuration of serial cable should I use"} \mid \text{question}) =$$

$$e(\text{2.6} \mid \text{question}) \times$$

$$e(\text{What} \mid \text{question}) \times$$

$$e(\text{configuration} \mid \text{question}) \times$$

$$e(\text{of} \mid \text{question}) \times$$

$$e(\text{serial} \mid \text{question}) \times$$

$$\ldots$$

- i.e. have a **language model** for each *tag*
FAQ Segmentation: McCallum et. al

- Second solution: first map each sentence to string of features:

  <question>2.6) What configuration of serial cable should I use

  ⇒

  <question>begins-with-number contains-alphanum contains-nonspace contains-number prev-is-blank

- Use a language model again:

\[
p(\text{"2.6) What configuration of serial cable should I use"} \mid \text{question}) = \\
\quad e(\text{begins-with-number} \mid \text{question}) \times \\
\quad e(\text{contains-alphanum} \mid \text{question}) \times \\
\quad e(\text{contains-nonspace} \mid \text{question}) \times \\
\quad e(\text{contains-number} \mid \text{question}) \times \\
\quad e(\text{prev-is-blank} \mid \text{question}) \times
\]
FAQ Segmentation: Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>ME-Stateless</td>
<td>0.038</td>
<td>0.362</td>
</tr>
<tr>
<td>TokenHMM</td>
<td>0.276</td>
<td>0.140</td>
</tr>
<tr>
<td>FeatureHMM</td>
<td>0.413</td>
<td>0.529</td>
</tr>
<tr>
<td>MEMM</td>
<td>0.867</td>
<td>0.681</td>
</tr>
</tbody>
</table>

- Precision and recall results are for recovering segments
FAQ Segmentation: Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>ME-Stateless</td>
<td>0.038</td>
<td>0.362</td>
</tr>
<tr>
<td>TokenHMM</td>
<td>0.276</td>
<td>0.140</td>
</tr>
<tr>
<td>FeatureHMM</td>
<td>0.413</td>
<td>0.529</td>
</tr>
<tr>
<td>MEMM</td>
<td>0.867</td>
<td>0.681</td>
</tr>
</tbody>
</table>

- Precision and recall results are for recovering segments
- ME-stateless is a log-linear model that treats every sentence separately (no dependence between adjacent tags)
FAQ Segmentation: Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>ME-Stateless</td>
<td>0.038</td>
<td>0.362</td>
</tr>
<tr>
<td>TokenHMM</td>
<td>0.276</td>
<td>0.140</td>
</tr>
<tr>
<td>FeatureHMM</td>
<td>0.413</td>
<td>0.529</td>
</tr>
<tr>
<td>MEMM</td>
<td>0.867</td>
<td>0.681</td>
</tr>
</tbody>
</table>

- Precision and recall results are for recovering segments
- ME-stateless is a log-linear model that treats every sentence separately (no dependence between adjacent tags)
- TokenHMM is an HMM with first solution we’ve just seen
## FAQ Segmentation: Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>ME-Stateless</td>
<td>0.038</td>
<td>0.362</td>
</tr>
<tr>
<td>TokenHMM</td>
<td>0.276</td>
<td>0.140</td>
</tr>
<tr>
<td>FeatureHMM</td>
<td>0.413</td>
<td>0.529</td>
</tr>
<tr>
<td>MEMM</td>
<td>0.867</td>
<td>0.681</td>
</tr>
</tbody>
</table>

- Precision and recall results are for recovering segments
- ME-stateless is a log-linear model that treats every sentence separately (no dependence between adjacent tags)
- TokenHMM is an HMM with first solution we've just seen
- FeatureHMM is an HMM with second solution we've just seen
FAQ Segmentation: Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>ME-Stateless</td>
<td>0.038</td>
<td>0.362</td>
</tr>
<tr>
<td>TokenHMM</td>
<td>0.276</td>
<td>0.140</td>
</tr>
<tr>
<td>FeatureHMM</td>
<td>0.413</td>
<td>0.529</td>
</tr>
<tr>
<td>MEMM</td>
<td>0.867</td>
<td>0.681</td>
</tr>
</tbody>
</table>

- Precision and recall results are for recovering segments
- ME-stateless is a log-linear model that treats every sentence separately (no dependence between adjacent tags)
- TokenHMM is an HMM with first solution we've just seen
- FeatureHMM is an HMM with second solution we've just seen
- MEMM is a log-linear trigram tagger (MEMM stands for “Maximum-Entropy Markov Model”)
Summary

- Key ideas in log-linear taggers:
  - Decompose
    \[ p(t_1 \ldots t_n \mid w_1 \ldots w_n) = \prod_{i=1}^{n} p(t_i \mid t_{i-2}, t_{i-1}, w_1 \ldots w_n) \]
  - Estimate
    \[ p(t_i \mid t_{i-2}, t_{i-1}, w_1 \ldots w_n) \]
    using a log-linear model
  - For a test sentence \( w_1 \ldots w_n \), use the Viterbi algorithm to find
    \[ \arg \max_{t_1 \ldots t_n} \left( \prod_{i=1}^{n} p(t_i \mid t_{i-2}, t_{i-1}, w_1 \ldots w_n) \right) \]
- Key advantage over HMM taggers: flexibility in the features they can use