# Log-Linear Models for Tagging <br> (Maximum-entropy Markov Models <br> (MEMMs)) 

Michael Collins, Columbia University

## 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 ./.

| N | $=$ Noun |
| :--- | :--- |
| V | $=$ Verb |
| P | $=$ Preposition |
| Adv | $=$ Adverb |
| Adj | $=$ Adjective |

## 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 $\quad=$ No entity
SC $=$ Start Company
CC $\quad=$ Continue Company
SL $\quad=$ Start Location
CL $\quad=$ 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 Huricane/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/algorithm that maps new sentences to their tag sequences.


## Overview

- Recap: The Tagging Problem
- Log-linear taggers


## 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 $w_{[1: n]}=w_{1}, w_{2}, \ldots, w_{n}$ ( $w_{i}$ 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

$$
p\left(t_{1}, t_{2}, \ldots, t_{n} \mid w_{1}, w_{2}, \ldots, w_{n}\right)
$$

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

## 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

$$
p\left(t_{1}, t_{2}, \ldots, t_{n} \mid w_{1}, w_{2}, \ldots, w_{n}\right)
$$

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

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

$$
t_{[1: n]}^{*}=\operatorname{argmax}_{t_{[1: n]}} p\left(t_{[1: n]} \mid w_{[1: n]}\right)
$$

## How to model $p\left(t_{[1: n]} \mid w_{[1: n]}\right)$ ?

## A Trigram Log-Linear Tagger:

$p\left(t_{[1: n]} \mid w_{[1: n]}\right)=\prod_{j=1}^{n} p\left(t_{j} \mid w_{1} \ldots w_{n}, t_{1} \ldots t_{j-1}\right) \quad$ Chain rule

## How to model $p\left(t_{[1: n]} \mid w_{[1: n]}\right)$ ?

## A Trigram Log-Linear Tagger:

$p\left(t_{[1: n]} \mid w_{[1: n]}\right)=\prod_{j=1}^{n} p\left(t_{j} \mid w_{1} \ldots w_{n}, t_{1} \ldots t_{j-1}\right) \quad$ Chain rule

$$
=\prod_{j=1}^{n} p\left(t_{j} \mid w_{1}, \ldots, w_{n}, t_{j-2}, t_{j-1}\right)
$$

Independence assumptions

- We take $t_{0}=t_{-1}=*$


## How to model $p\left(t_{[1: n]} \mid w_{[1: n]}\right)$ ?

## A Trigram Log-Linear Tagger:

$$
p\left(t_{[1: n]} \mid w_{[1: n]}\right)=\prod_{j=1}^{n} p\left(t_{j} \mid w_{1} \ldots w_{n}, t_{1} \ldots t_{j-1}\right) \quad \text { Chain rule }
$$

$$
=\prod_{j=1}^{n} p\left(t_{j} \mid w_{1}, \ldots, w_{n}, t_{j-2}, t_{j-1}\right)
$$

Independence assumptions

- We take $t_{0}=t_{-1}=$ *
- Independence assumption: each tag only depends on previous two tags

$$
p\left(t_{j} \mid w_{1}, \ldots, w_{n}, t_{1}, \ldots, t_{j-1}\right)=p\left(t_{j} \mid w_{1}, \ldots, w_{n}, t_{j-2}, t_{j-1}\right)
$$

## 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 ??
$\mathcal{Y}=\{\mathrm{NN}, \mathrm{NNS}, \mathrm{Vt}, \mathrm{Vi}, \mathrm{IN}, \mathrm{DT}, \ldots\}$


## Representation: Histories

- A history is a 4-tuple $\left\langle t_{-2}, t_{-1}, w_{[1: n]}, i\right\rangle$
- $t_{-2}, t_{-1}$ 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/DT important/JJ base/?? from which Spain expanded its empire into the rest of the Western Hemisphere.

- $t_{-2}, t_{-1}=\mathrm{DT}, \mathrm{JJ}$
- $w_{[1: n]}=\langle$ Hispaniola,quickly, became, ..., Hemisphere, ..
- $i=6$

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 $f: \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ (Often binary features or indicator functions $f: \mathcal{X} \times \mathcal{Y} \rightarrow\{0,1\})$.
- Say we have $m$ features $f_{k}$ for $k=1 \ldots m$ $\Rightarrow$ 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 $\left\langle t_{-2}, t_{-1}, w_{[1: n]}, i\right\rangle$
- $\mathcal{Y}=\{$ NN, NNS, Vt, Vi, IN, DT,..$\}$
- We have $m$ features $f_{k}: \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ for $k=1 \ldots m$

For example:

$$
\begin{aligned}
& f_{1}(h, t)= \begin{cases}1 & \text { if current word } w_{i} \text { is base and } t=\mathrm{Vt} \\
0 & \text { otherwise }\end{cases} \\
& f_{2}(h, t)= \begin{cases}1 & \text { if current word } w_{i} \text { ends in ing and } t=\mathrm{VBG} \\
0 & \text { otherwise }\end{cases}
\end{aligned}
$$

## The Full Set of Features in [(Ratnaparkhi, 96)]

- Word/tag features for all word/tag pairs, e.g.,

$$
f_{100}(h, t)= \begin{cases}1 & \text { if current word } w_{i} \text { is base and } t=\mathrm{Vt} \\ 0 & \text { otherwise }\end{cases}
$$

- Spelling features for all prefixes/suffixes of length $\leq 4$, e.g.,

$$
\begin{aligned}
& f_{101}(h, t)= \begin{cases}1 & \text { if current word } w_{i} \text { ends in ing and } t=\mathrm{VBG} \\
0 & \text { otherwise }\end{cases} \\
& f_{102}(h, t)= \begin{cases}1 & \text { if current word } w_{i} \text { starts with pre and } t=\mathrm{NN} \\
0 & \text { otherwise }\end{cases}
\end{aligned}
$$

## The Full Set of Features in [(Ratnaparkhi, 96)]

- Contextual Features, e.g.,

$$
\begin{aligned}
& f_{103}(h, t)= \begin{cases}1 & \text { if }\left\langle t_{-2}, t_{-1}, t\right\rangle=\langle\mathrm{DT}, \mathrm{JJ}, \mathrm{~V} \mathrm{t}\rangle \\
0 & \text { otherwise }\end{cases} \\
& f_{104}(h, t)= \begin{cases}1 & \text { if }\left\langle t_{-1}, t\right\rangle=\langle\mathrm{JJ}, \mathrm{~V} \mathrm{t}\rangle \\
0 & \text { otherwise }\end{cases} \\
& f_{105}(h, t)= \begin{cases}1 & \text { if }\langle t\rangle=\langle\mathrm{V} \mathrm{t}\rangle \\
0 & \text { otherwise }\end{cases} \\
& f_{106}(h, t)= \begin{cases}1 & \text { if previous word } w_{i-1}=\text { the and } t=\mathrm{Vt} \\
0 & \text { otherwise }\end{cases} \\
& f_{107}(h, t)= \begin{cases}1 & \text { if next word } w_{i+1}=\text { the and } t=\mathrm{Vt} \\
0 & \text { otherwise }\end{cases}
\end{aligned}
$$

## 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 $k=1 \ldots m$
$\Rightarrow$ 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

$$
p(y \mid x ; v)=\frac{e^{v \cdot f(x, y)}}{\sum_{y^{\prime} \in \mathcal{Y}} e^{v \cdot f\left(x, y^{\prime}\right)}}
$$

## Training the Log-Linear Model

- To train a log-linear model, we need a training set $\left(x_{i}, y_{i}\right)$ for $i=1 \ldots n$. Then search for

$$
v^{*}=\operatorname{argmax}_{v}(\underbrace{\sum_{i} \log p\left(y_{i} \mid x_{i} ; v\right)}_{\text {Log-Likelihood }}-\underbrace{\frac{\lambda}{2} \sum_{k} v_{k}^{2}}_{\text {Regularizer }})
$$

(see last lecture on log-linear models)

- 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

$$
\arg \max _{t_{1} \ldots t_{n}} p\left(t_{1} \ldots t_{n} \mid w_{1} \ldots w_{n}\right)
$$

We assume that $p$ takes the form

$$
p\left(t_{1} \ldots t_{n} \mid w_{1} \ldots w_{n}\right)=\prod_{i=1}^{n} q\left(t_{i} \mid t_{i-2}, t_{i-1}, w_{[1: n]}, i\right)
$$

(In our case $q\left(t_{i} \mid t_{i-2}, t_{i-1}, w_{[1: n]}, i\right)$ is the estimate from a log-linear model.)

## The Viterbi Algorithm

- Define $n$ to be the length of the sentence
- Define

$$
r\left(t_{1} \ldots t_{k}\right)=\prod_{i=1}^{k} q\left(t_{i} \mid t_{i-2}, t_{i-1}, w_{[1: n]}, i\right)
$$

- Define a dynamic programming table
$\pi(k, u, v)=$ maximum probability of a tag sequence ending in tags $u, v$ at position $k$
that is,

$$
\pi(k, u, v)=\max _{\left\langle t_{1}, \ldots, t_{k-2}\right\rangle} r\left(t_{1} \ldots t_{k-2}, u, v\right)
$$

## A Recursive Definition

Base case:

$$
\pi\left(0,{ }^{*}, *\right)=1
$$

## Recursive definition:

For any $k \in\{1 \ldots n\}$, for any $u \in \mathcal{S}_{k-1}$ and $v \in \mathcal{S}_{k}$ :

$$
\pi(k, u, v)=\max _{t \in \mathcal{S}_{k-2}}\left(\pi(k-1, t, u) \times q\left(v \mid t, u, w_{[1: n]}, k\right)\right)
$$

where $\mathcal{S}_{k}$ 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 $q\left(v \mid t, u, w_{[1: n]}, i\right)$ for any tag-trigram $t, u, v$, for any $i \in\{1 \ldots n\}$
Initialization: Set $\pi\left(0,{ }^{*},{ }^{*}\right)=1$.

## Algorithm:

- For $k=1 \ldots n$,
- For $u \in \mathcal{S}_{k-1}, v \in \mathcal{S}_{k}$,

$$
\begin{aligned}
\pi(k, u, v) & =\max _{t \in \mathcal{S}_{k-2}}\left(\pi(k-1, t, u) \times q\left(v \mid t, u, w_{[1: n]}, k\right)\right) \\
\operatorname{bp}(k, u, v) & =\arg \max _{t \in \mathcal{S}_{k-2}}\left(\pi(k-1, t, u) \times q\left(v \mid t, u, w_{[1: n]}, k\right)\right)
\end{aligned}
$$

- Set $\left(t_{n-1}, t_{n}\right)=\arg \max _{(u, v)} \pi(n, u, v)$
- For $k=(n-2) \ldots 1, t_{k}=b p\left(k+2, t_{k+1}, t_{k+2}\right)$
- Return the tag sequence $t_{1} \ldots t_{n}$


## 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

$$
p(w o r d \mid t a g)
$$

is difficult in this domain

## FAQ Segmentation: McCallum et. al

<head>X-NNTP-POSTER: NewsHound v1. 33
<head>
<head>Archive name: acorn/faq/part2
<head>Frequency: monthly
<head>
<question>2.6) What configuration of serial cable should I use <answer>
<answer> Here follows a diagram of the necessary connections <answer>programs to work properly. They are as far as I know t <answer>agreed upon by commercial comms software developers fo <answer>
<answer> Pins 1, 4, and 8 must be connected together inside <answer>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

<head>X-NNTP-POSTER: NewsHound v1.33
<head>
<head>Archive name: acorn/faq/part2
<head>Frequency: monthly
<head>
<question>2.6) What configuration of serial cable should I use

Here follows a diagram of the necessary connections
$\Rightarrow$ "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

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

- First solution for $p$ (word $\mid$ tag $)$ :
$p$ ("2.6) What configuration of serial cable should I use" | question) $=$ $e(2.6) \mid$ question $) \times$
$e($ What | question $) \times$
$e($ configuration $\mid$ question $) \times$
$e(o f \mid$ question $) \times$
$e($ serial | question $) \times$
- i.e. have a language model for each $\operatorname{tag}$


## 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 $\Rightarrow$
<question>begins-with-number contains-alphanum contains-nonspace contains-number prev-is-blank
- Use a language model again:
$p$ ("2.6) What configuration of serial cable should I use" | question) $=$
$e$ (begins-with-number | question) $\times$
$e($ contains-alphanum | question) $\times$
$e($ contains-nonspace | question $) \times$
$e($ contains-number | question $) \times$
$e($ prev-is-blank | question $) \times$


## FAQ Segmentation: Results

| Method | Precision | Recall |
| :--- | :--- | :--- |
| ME-Stateless | 0.038 | 0.362 |
| TokenHMM | 0.276 | 0.140 |
| FeatureHMM | 0.413 | 0.529 |
| MEMM | 0.867 | 0.681 |

- Precision and recall results are for recovering segments


## FAQ Segmentation: Results

| Method | Precision | Recall |
| :--- | :--- | :--- |
| ME-Stateless | 0.038 | 0.362 |
| TokenHMM | 0.276 | 0.140 |
| FeatureHMM | 0.413 | 0.529 |
| MEMM | 0.867 | 0.681 |

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


## FAQ Segmentation: Results

| Method | Precision | Recall |
| :--- | :--- | :--- |
| ME-Stateless | 0.038 | 0.362 |
| TokenHMM | 0.276 | 0.140 |
| FeatureHMM | 0.413 | 0.529 |
| MEMM | 0.867 | 0.681 |

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


## FAQ Segmentation: Results

| Method | Precision | Recall |
| :--- | :--- | :--- |
| ME-Stateless | 0.038 | 0.362 |
| TokenHMM | 0.276 | 0.140 |
| FeatureHMM | 0.413 | 0.529 |
| MEMM | 0.867 | 0.681 |

- Precision and recall results are for recovering segments
- ME-stateless is a log-linear model that treats every sentence seperately (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

| Method | Precision | Recall |
| :--- | :--- | :--- |
| ME-Stateless | 0.038 | 0.362 |
| TokenHMM | 0.276 | 0.140 |
| FeatureHMM | 0.413 | 0.529 |
| MEMM | 0.867 | 0.681 |

- Precision and recall results are for recovering segments
- ME-stateless is a log-linear model that treats every sentence seperately (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

$$
p\left(t_{1} \ldots t_{n} \mid w_{1} \ldots w_{n}\right)=\prod_{i=1}^{n} p\left(t_{i} \mid t_{i-2}, t_{i-1}, w_{1} \ldots w_{n}\right)
$$

- Estimate

$$
p\left(t_{i} \mid t_{i-2}, t_{i-1}, w_{1} \ldots w_{n}\right)
$$

using a log-linear model

- For a test sentence $w_{1} \ldots w_{n}$, use the Viterbi algorithm to find

$$
\arg \max _{t_{1} \ldots t_{n}}\left(\prod_{i=1}^{n} p\left(t_{i} \mid t_{i-2}, t_{i-1}, w_{1} \ldots w_{n}\right)\right)
$$

- Key advantage over HMM taggers: flexibility in the features they can use

