Log-Linear Models for Tagging (Maximum-entropy Markov Models (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:

SC

CC

SL

CL

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
 - = Start Company
 - = Continue Company
 - = Start Location
 - = 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

 We have an input sentence w_[1:n] = w₁, w₂,..., w_n (w_i is the *i*'th word in the sentence)

- We have an input sentence w_[1:n] = w₁, w₂,..., w_n (w_i is the *i*'th word in the sentence)
- We have a tag sequence t_[1:n] = t₁, t₂,..., t_n (t_i is the *i*'th tag in the sentence)

- We have an input sentence w_[1:n] = w₁, w₂,..., w_n (w_i is the *i*'th word in the sentence)
- We have a tag sequence t_[1:n] = t₁, t₂,..., 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 $p(t_1 \dots t_n, w_1 \dots w_n)$)

- We have an input sentence w_[1:n] = w₁, w₂,..., w_n (w_i is the *i*'th word in the sentence)
- We have a tag sequence t_[1:n] = t₁, t₂,..., 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 $p(t_1 \dots t_n, w_1 \dots w_n)$)

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

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

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

A Trigram Log-Linear Tagger:

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

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

A Trigram Log-Linear Tagger:

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

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

Independence assumptions

• We take
$$t_0 = t_{-1} = *$$

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

A Trigram Log-Linear Tagger:

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

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

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 $\ref{eq:main_star}$ $\mathcal{Y} = \{NN, NNS, Vt, Vi, IN, DT, \dots\}$

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
- $\blacktriangleright~\mathcal{X}$ 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 .

- $\blacktriangleright t_{-2}, t_{-1} = \mathsf{DT}, \ \mathsf{JJ}$
- $\blacktriangleright \ w_{[1:n]} = \langle Hispaniola, quickly, became, \dots, Hemisphere, . \rangle$
- ► *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 : X × Y → ℝ
 (Often binary features or indicator functions f : X × Y → {0,1}).
- Say we have m features f_k for $k = 1 \dots m$ $\Rightarrow A$ 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 $\langle t_{-2}, t_{-1}, w_{[1:n]}, i \rangle$

•
$$\mathcal{Y} = \{NN, NNS, Vt, Vi, IN, DT, \dots\}$$

 \blacktriangleright We have m features $f_k: \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ for $k = 1 \dots m$

For example:

 $\begin{array}{l} f_1(\langle \mathsf{JJ}, \, \mathsf{DT}, \, \langle \, \mathsf{Hispaniola}, \, \dots \, \rangle, \, \mathsf{6} \rangle, \mathsf{Vt}) = 1 \\ f_2(\langle \mathsf{JJ}, \, \mathsf{DT}, \, \langle \, \mathsf{Hispaniola}, \, \dots \, \rangle, \, \mathsf{6} \rangle, \mathsf{Vt}) = 0 \end{array}$

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

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

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

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

$$f_{101}(h,t) = \left\{ egin{array}{cc} 1 & {
m if \ current \ word \ } w_i \ {
m ends \ in \ ing \ and \ } t = {
m VBG} \\ 0 & {
m otherwise} \end{array}
ight.$$

 $f_{102}(h,t) = \begin{cases} 1 & \text{if current word } w_i \text{ starts with pre and } t = NN \\ 0 & \text{otherwise} \end{cases}$

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

► Contextual Features, e.g.,

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

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 : X × Y → ℝ
 (Often binary features or indicator functions f : X × Y → {0,1}).
- Say we have m features f_k for $k = 1 \dots m$ \Rightarrow A feature vector $f(x, y) \in \mathbb{R}^m$ for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.
- \blacktriangleright 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...n. Then search for

$$v^* = \operatorname{argmax}_{v} \left(\underbrace{\sum_{i} \log p(y_i | x_i; v)}_{\textit{Log-Likelihood}} - \underbrace{\frac{\lambda}{2} \sum_{k} v_k^2}_{\textit{Regularizer}} \right)$$

(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 $w_1 \dots w_n$, find

$$\arg\max_{t_1\dots t_n} p(t_1\dots t_n \mid w_1\dots w_n)$$

We assume that \boldsymbol{p} takes the form

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

(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

- \blacktriangleright Define n to be the length of the sentence
- Define

$$r(t_1 \dots t_k) = \prod_{i=1}^k q(t_i | t_{i-2}, t_{i-1}, w_{[1:n]}, i)$$

Define a dynamic programming table

$$\pi(k, u, v) =$$
 maximum probability of a tag sequence ending
in tags u, v at position k

that is,

$$\pi(k, u, v) = \max_{\langle t_1, \dots, t_{k-2} \rangle} r(t_1 \dots t_{k-2}, u, v)$$

A Recursive Definition

Base case:

$$\pi(0, *, *) = 1$$

Recursive definition:

For any $k \in \{1 \dots n\}$, for any $u \in S_{k-1}$ and $v \in S_k$:

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

where \mathcal{S}_k is the set of possible tags at position k

The Viterbi Algorithm with Backpointers

Input: a sentence $w_1 \dots w_n$, log-linear model that provides $q(v|t, u, w_{[1:n]}, i)$ for any tag-trigram t, u, v, for any $i \in \{1 \dots n\}$ Initialization: Set $\pi(0, *, *) = 1$. **Algorithm:** For $k = 1 \dots n$. For $u \in \mathcal{S}_{k-1}$, $v \in \mathcal{S}_k$, $\pi(k, u, v) = \max_{t \in \mathcal{S}_{k-2}} \left(\pi(k-1, t, u) \times q(v|t, u, w_{[1:n]}, k) \right)$ $bp(k, u, v) = \arg \max_{t \in S_{k-2}} \left(\pi(k-1, t, u) \times q(v|t, u, w_{[1:n]}, k) \right)$ • Set $(t_{n-1}, t_n) = \arg \max_{(u,v)} \pi(n, u, v)$

- For $k = (n-2) \dots 1$, $t_k = bp(k+2, t_{k+1}, t_{k+2})$
- **Return** the tag sequence $t_1 \dots t_n$

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(word|tag)

is difficult in this domain

FAQ Segmentation: McCallum et. al

<head>X-NNTP-POSTER: NewsHound v1.33 <head> <head>Archive name: acorn/fag/part2 <head>Frequency: monthly <head> <question>2.6) What configuration of serial cable should I use <answer> Here follows a diagram of the necessary connections <answer> <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

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

<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) | question)× e(What | question)× e(configuration | question)× e(of | question)× e(serial | question)×

• 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

 \Rightarrow

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

Use a language model again:

 $\begin{array}{l} p(``2.6) \mbox{ What configuration of serial cable should I use'' | question)} = \\ e(\mbox{begins-with-number | question}) \times \\ e(\mbox{contains-alphanum | question}) \times \\ e(\mbox{contains-nonspace | question}) \times \\ e(\mbox{contains-number | question}) \times \\ e(\mbox{prev-is-blank | question}) \times \end{array}$

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

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)

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

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

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

$$p(t_1 \dots t_n | w_1 \dots w_n) = \prod_{i=1}^n p(t_i | t_{i-2}, t_{i-1}, w_1 \dots w_n)$$

Estimate

$$p(t_i|t_{i-2},t_{i-1},w_1\ldots w_n)$$

- using a log-linear model
- For a test sentence $w_1 \dots w_n$, use the Viterbi algorithm to find

$$\arg\max_{t_1\dots t_n} \left(\prod_{i=1}^n p(t_i|t_{i-2}, t_{i-1}, w_1\dots w_n) \right)$$

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