Maximum Entropy Markov Models

Instructor: Wei Xu

Many slides from Michael Collins

The Language Modeling Problem

- w_i is the i'th word in a document
- Estimate a distribution $p(w_i|w_1, w_2, \ldots w_{i-1})$ given previous "history" $w_1, \ldots, w_{i-1}$.
- E.g., $w_1, \ldots, w_{i-1} =$

  Third, the notion "grammatical in English" cannot be identified in any way with the notion "high order of statistical approximation to English". It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence, in any statistical

Trigram Models

- Estimate a distribution $p(w_i | w_1, w_2, \ldots w_{i-1})$ given previous "history" $w_1, \ldots, w_{i-1} =$

Third, the notion "grammatical in English" cannot be identified in any way with the notion "high order of statistical approximation to English". It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence, in any statistical

- **Trigram estimates:**

$$\begin{aligned} q(\text{model}|w_1, \ldots w_{i-1}) &= \lambda_1 q_{ML}(\text{model}|w_{i-2} = \text{any}, w_{i-1} = \text{statistical}) + \\ &\quad \lambda_2 q_{ML}(\text{model}|w_{i-1} = \text{statistical}) + \\ &\quad \lambda_3 q_{ML}(\text{model}) \end{aligned}$$

where $\lambda_i \geq 0$, $\sum_i \lambda_i = 1$, $q_{ML}(y|x) = \frac{Count(x,y)}{Count(x)}$

Trigram Models

$$
\begin{aligned}
q(\text{model}|w_1, \ldots w_{i-1}) \quad = \quad & \lambda_1 q_{ML}(\text{model}|w_{i-2} = \text{any}, w_{i-1} = \text{statistical}) + \\
& \lambda_2 q_{ML}(\text{model}|w_{i-1} = \text{statistical}) + \\
& \lambda_3 q_{ML}(\text{model})
\end{aligned}
$$

- Makes use of only bigram, trigram, unigram estimates
- Many other "features" of $w_1, \ldots, w_{i-1}$ may be useful, e.g.,:

$$
\begin{array}{lll}
q_{ML}(\text{model} & | & w_{i-2} = \text{any}) \\
q_{ML}(\text{model} & | & w_{i-1} \text{ is an adjective}) \\
q_{ML}(\text{model} & | & w_{i-1} \text{ ends in ``ical''}) \\
q_{ML}(\text{model} & | & author = \text{Chomsky}) \\
q_{ML}(\text{model} & | & \text{``model'' does not occur somewhere in } w_1, \ldots w_{i-1}) \\
q_{ML}(\text{model} & | & \text{``grammatical'' occurs somewhere in } w_1, \ldots w_{i-1})
\end{array}
$$

A Naive Approach

$$
\begin{aligned}
&q(\text{model}|w_1, \ldots w_{i-1}) = \\
&\lambda_1 q_{ML}(\text{model}|w_{i-2} = \text{any}, w_{i-1} = \text{statistical}) + \\
&\lambda_2 q_{ML}(\text{model}|w_{i-1} = \text{statistical}) + \\
&\lambda_3 q_{ML}(\text{model}) + \\
&\lambda_4 q_{ML}(\text{model}|w_{i-2} = \text{any}) + \\
&\lambda_5 q_{ML}(\text{model}|w_{i-1} \text{ is an adjective}) + \\
&\lambda_6 q_{ML}(\text{model}|w_{i-1} \text{ ends in ``ical''}) + \\
&\lambda_7 q_{ML}(\text{model}|author = \text{Chomsky}) + \\
&\lambda_8 q_{ML}(\text{model}|\text{``model'' does not occur somewhere in } w_1, \ldots w_{i-1}) + \\
&\lambda_9 q_{ML}(\text{model}|\text{``grammatical'' occurs somewhere in } w_1, \ldots w_{i-1})
\end{aligned}
$$

This quickly becomes very unwieldy...

A Second Example: 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
...

A Second Example: Part-of-Speech Tagging

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 ??
{NN, NNS, Vt, Vi, IN, DT, ...}

- The task: model the distribution

similar to HMM, but different!

$$p(t_i | t_1, \ldots, t_{i-1}, w_1 \ldots w_n)$$

where t_i is the i'th tag in the sequence, w_i is the i'th word

A Second Example: Part-of-Speech Tagging

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

- The task: model the distribution

$$p(t_i|t_1, \ldots, t_{i-1}, w_1 \ldots w_n)$$

where t_i is the i'th tag in the sequence, w_i is the i'th word

- Again: many "features" of $t_1, \ldots, t_{i-1}, w_1 \ldots w_n$ may be relevant

$$\begin{array}{lcl}
q_{ML}(\text{NN} & | & w_i = \text{base}) \\
q_{ML}(\text{NN} & | & t_{i-1} \text{ is JJ}) \\
q_{ML}(\text{NN} & | & w_i \text{ ends in ``e''}) \\
q_{ML}(\text{NN} & | & w_i \text{ ends in ``se''}) \\
q_{ML}(\text{NN} & | & w_{i-1} \text{ is ``important''}) \\
q_{ML}(\text{NN} & | & w_{i+1} \text{ is ``from''})
\end{array}$$

Overview

- Log-linear models
- Parameter estimation in log-linear models
- Smoothing/regularization in log-linear models

The General Problem

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

Language Modeling

- x is a "history" $w_1, w_2, \ldots w_{i-1}$, e.g.,

  Third, the notion "grammatical in English" cannot be identified in any way with the notion "high order of statistical approximation to English". It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence, in any statistical

- y is an "outcome" w_i

Feature Vector Representations

- Aim is to provide a conditional probability $p(y \mid x)$ for "decision" y given "history" x
- A **feature** is a function $f_k(x, y) \in \mathbb{R}$
  (Often **binary features** or **indicator functions** $f_k(x, y) \in \{0, 1\}$).
- 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, y

Language Modeling

- x is a "history" $w_1, w_2, \dots w_{i-1}$, e.g.,

  Third, the notion "grammatical in English" cannot be identified in any way with the notion "high order of statistical approximation to English". It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence, in any statistical

- y is an "outcome" w_i
- Example features:

$$f_1(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model} \\ 0 & \text{otherwise} \end{cases}$$

unigram feature

$$f_2(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model} \text{ and } w_{i-1} = \texttt{statistical} \\ 0 & \text{otherwise} \end{cases}$$

bigram feature

$$f_3(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model},\ w_{i-2} = \texttt{any},\ w_{i-1} = \texttt{statistical} \\ 0 & \text{otherwise} \end{cases}$$

trigram feature

$$f_4(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model},\ w_{i-2} = \text{any} \\ 0 & \text{otherwise} \end{cases}$$

skip bigram feature

$$f_5(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model},\ w_{i-1} \text{ is an adjective} \\ 0 & \text{otherwise} \end{cases}$$

$$f_6(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model},\ w_{i-1} \text{ ends in ``ical''} \\ 0 & \text{otherwise} \end{cases}$$

$$f_7(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model},\ \text{author} = \text{Chomsky} \\ 0 & \text{otherwise} \end{cases}$$

$$f_8(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model},\ \text{``model'' is not in } w_1, \ldots w_{i-1} \\ 0 & \text{otherwise} \end{cases}$$

$$f_9(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model},\ \text{``grammatical'' is in } w_1, \ldots w_{i-1} \\ 0 & \text{otherwise} \end{cases}$$

Defining Features in Practice

- We had the following "trigram" feature:

$$f_3(x, y) = \begin{cases} 1 & \text{if } y = \texttt{model},\ w_{i-2} = \texttt{any},\ w_{i-1} = \texttt{statistical} \\ 0 & \text{otherwise} \end{cases}$$

- In practice, we would probably introduce one trigram feature for every trigram seen in the training data: i.e., for all trigrams (u, v, w) seen in training data, create a feature

$$f_{N(u,v,w)}(x, y) = \begin{cases} 1 & \text{if } y = w,\ w_{i-2} = u,\ w_{i-1} = v \\ 0 & \text{otherwise} \end{cases}$$

Do not include trigrams that are not seen in the training data

index of unique trigrams in training data

where $N(u, v, w)$ is a function that maps each (u, v, w) trigram to a different integer

The POS-Tagging Example

- Each x is a "history" of the form $\langle t_1, t_2, \ldots, t_{i-1}, w_1 \ldots w_n, i \rangle$
- Each y is a POS tag, such as NN, NNS, Vt, Vi, IN, DT, …
- We have m features $f_k(x, y)$ for $k = 1 \ldots m$

For example:

$$f_1(x, y) = \begin{cases} 1 & \text{if current word } w_i \text{ is base and } y = \texttt{Vt} \\ 0 & \text{otherwise} \end{cases}$$

word/tag pair

$$f_2(x, y) = \begin{cases} 1 & \text{if current word } w_i \text{ ends in } \texttt{ing} \text{ and } y = \texttt{VBG} \\ 0 & \text{otherwise} \end{cases}$$

$$\ldots$$

The Full Set of Features in Ratnaparkhi, 1996

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

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

- Spelling features for all prefixes/suffixes of length ≤ 4, e.g.,

$$f_{101}(x,y) = \begin{cases} 1 & \text{if current word } w_i \text{ ends in } \texttt{ing} \text{ and } y = \texttt{VBG} \\ 0 & \text{otherwise} \end{cases}$$

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

The Full Set of Features in Ratnaparkhi, 1996

- Contextual Features, e.g.,

$$f_{103}(x,y) = \begin{cases} 1 & \text{if } \langle t_{i-2}, t_{i-1}, y\rangle = \langle \text{DT, JJ, Vt}\rangle \\ 0 & \text{otherwise} \end{cases}$$

trigram tag feature

$$f_{104}(x,y) = \begin{cases} 1 & \text{if } \langle t_{i-1}, y\rangle = \langle \text{JJ, Vt}\rangle \\ 0 & \text{otherwise} \end{cases}$$

$$f_{105}(x,y) = \begin{cases} 1 & \text{if } \langle y\rangle = \langle \text{Vt}\rangle \\ 0 & \text{otherwise} \end{cases}$$

$$f_{106}(x,y) = \begin{cases} 1 & \text{if previous word } w_{i-1} = \textit{the} \text{ and } y = \text{Vt} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{107}(x,y) = \begin{cases} 1 & \text{if next word } w_{i+1} = \textit{the} \text{ and } y = \text{Vt} \\ 0 & \text{otherwise} \end{cases}$$

The Final Result

- We can come up with practically any questions (*features*) regarding history/tag pairs.
- For a given history $x \in \mathcal{X}$, each label in $\mathcal{Y}$ is mapped to a different feature vector

$$
\begin{aligned}
f(\langle \text{JJ, DT, } \langle \text{ Hispaniola, } \ldots \rangle, 6\rangle, \text{Vt}) &= 1001011001001100110 \\
f(\langle \text{JJ, DT, } \langle \text{ Hispaniola, } \ldots \rangle, 6\rangle, \text{JJ}) &= 0110010101011110010 \\
f(\langle \text{JJ, DT, } \langle \text{ Hispaniola, } \ldots \rangle, 6\rangle, \text{NN}) &= 0001111101001100100 \\
f(\langle \text{JJ, DT, } \langle \text{ Hispaniola, } \ldots \rangle, 6\rangle, \text{IN}) &= 0001011011000000010 \\
&\ldots
\end{aligned}
$$

often sparse with few 1's vs. 0's

Parameter Vectors

- Given features $f_k(x, y)$ for $k = 1 \ldots m$, also define a **parameter vector** $v \in \mathbb{R}^m$

all possible m-dimensional real value vectors

- Each (x, y) pair is then mapped to a "score"

$$v \cdot f(x, y) = \sum_k v_k f_k(x, y)$$

Recall Logistic/Softmax Regression!

Language Modeling

- x is a "history" $w_1, w_2, \ldots w_{i-1}$, e.g.,
  Third, the notion "grammatical in English" cannot be identified in any way with the notion "high order of statistical approximation to English". It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence, in any statistical
- Each possible y gets a different score:

$$\begin{aligned} v \cdot f(x, model) &= 5.6 & \qquad v \cdot f(x, the) &= -3.2 \\ v \cdot f(x, is) &= 1.5 & \qquad v \cdot f(x, of) &= 1.3 \\ v \cdot f(x, models) &= 4.5 & \qquad \ldots \end{aligned}$$

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 $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$
  (Often binary features or indicator functions
  $f_k : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$).
- 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

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

Softmax!

Why the name?

$$\log p(y \mid x; v) = \underbrace{v \cdot f(x, y)}_{\text{Linear term}} - \underbrace{\log \sum_{y' \in \mathcal{Y}} e^{v \cdot f(x, y')}}_{\text{Normalization term}}$$

Overview

- Log-linear models
- Parameter estimation in log-linear models
- Smoothing/regularization in log-linear models

Maximum-Likelihood Estimation

- Maximum-likelihood estimates given training sample $(x^{(i)}, y^{(i)})$ for $i = 1 \ldots n$, each $(x^{(i)}, y^{(i)}) \in \mathcal{X} \times \mathcal{Y}$:

$$v_{ML} = \text{argmax}_{v \in \mathbb{R}^m} L(v)$$

where

$$L(v) = \sum_{i=1}^{n} \log p(y^{(i)} \mid x^{(i)}; v) = \sum_{i=1}^{n} v \cdot f(x^{(i)}, y^{(i)}) - \sum_{i=1}^{n} \log \sum_{y' \in \mathcal{Y}} e^{v \cdot f(x^{(i)}, y')}$$

Calculating the Maximum-Likelihood Estimates

- Need to maximize:

$$L(v) = \sum_{i=1}^{n} v \cdot f(x^{(i)}, y^{(i)}) - \sum_{i=1}^{n} \log \sum_{y' \in \mathcal{Y}} e^{v \cdot f(x^{(i)}, y')}$$

- Calculating gradients:

$$\begin{aligned}
\frac{dL(v)}{dv_k} &= \sum_{i=1}^{n} f_k(x^{(i)}, y^{(i)}) - \sum_{i=1}^{n} \frac{\sum_{y' \in \mathcal{Y}} f_k(x^{(i)}, y') e^{v \cdot f(x^{(i)}, y')}}{\sum_{z' \in \mathcal{Y}} e^{v \cdot f(x^{(i)}, z')}} \\
&= \sum_{i=1}^{n} f_k(x^{(i)}, y^{(i)}) - \sum_{i=1}^{n} \sum_{y' \in \mathcal{Y}} f_k(x^{(i)}, y') \frac{e^{v \cdot f(x^{(i)}, y')}}{\sum_{z' \in \mathcal{Y}} e^{v \cdot f(x^{(i)}, z')}} \\
&= \underbrace{\sum_{i=1}^{n} f_k(x^{(i)}, y^{(i)})}_{\text{Empirical counts}} - \underbrace{\sum_{i=1}^{n} \sum_{y' \in \mathcal{Y}} f_k(x^{(i)}, y') p(y' \mid x^{(i)}; v)}_{\text{Expected counts}}
\end{aligned}$$

Gradient Ascent Methods

- Need to maximize $L(v)$ where

$$\frac{dL(v)}{dv} = \sum_{i=1}^{n} f(x^{(i)}, y^{(i)}) - \sum_{i=1}^{n} \sum_{y' \in \mathcal{Y}} f(x^{(i)}, y') p(y' \mid x^{(i)}; v)$$

Initialization: $v = 0$

Iterate until convergence:

- Calculate $\Delta = \frac{dL(v)}{dv}$
- Calculate $\beta_* = \text{argmax}_\beta L(v + \beta\Delta)$ (Line Search)
- Set $v \leftarrow v + \beta_*\Delta$

Conjugate Gradient Methods

- (Vanilla) gradient ascent can be very slow
- Conjugate gradient methods require calculation of gradient at each iteration, but do a line search in **a direction which is a function of the current gradient, and the previous step taken**.
- Conjugate gradient packages are widely available
  In general: they require a function

$$\texttt{calc\_gradient}(v) \rightarrow \left(L(v), \frac{dL(v)}{dv} \right)$$

  and that's about it!

Overview

- Log-linear models
- Parameter estimation in log-linear models
- Smoothing/regularization in log-linear models

Smoothing in Log-Linear Models

- Say we have a feature:

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

- In training data, base is seen 3 times, with Vt every time
- Maximum likelihood solution satisfies

$$\sum_i f_{100}(x^{(i)}, y^{(i)}) = \sum_i \sum_y p(y \mid x^{(i)}; v) f_{100}(x^{(i)}, y)$$

$\Rightarrow p(\mathsf{Vt} \mid x^{(i)}; v) = 1$ for any history $x^{(i)}$ where $w_i =$ base
$\Rightarrow v_{100} \rightarrow \infty$ at maximum-likelihood solution (most likely)
$\Rightarrow p(\mathsf{Vt} \mid x; v) = 1$ for any test data history x where $w =$ base

Regularization

- Modified loss function

$$L(v) = \sum_{i=1}^{n} v \cdot f(x^{(i)}, y^{(i)}) - \sum_{i=1}^{n} \log \sum_{y' \in \mathcal{Y}} e^{v \cdot f(x^{(i)}, y')} - \frac{\lambda}{2} \sum_{k=1}^{m} v_k^2$$

- Calculating gradients:

$$\frac{dL(v)}{dv_k} = \underbrace{\sum_{i=1}^{n} f_k(x^{(i)}, y^{(i)})}_{\text{Empirical counts}} - \underbrace{\sum_{i=1}^{n} \sum_{y' \in \mathcal{Y}} f_k(x^{(i)}, y') p(y' \mid x^{(i)}; v)}_{\text{Expected counts}} - \lambda v_k$$

- Can run conjugate gradient methods as before
- Adds a penalty for large weights

Experiments with Regularization

- [Chen and Rosenfeld, 1998]: apply log-linear models to language modeling: Estimate $q(w_i \mid w_{i-2}, w_{i-1})$

- Unigram, bigram, trigram features, e.g.,

$$f_1(w_{i-2}, w_{i-1}, w_i) = \begin{cases} 1 & \text{if trigram is } \texttt{(the,dog,laughs)} \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(w_{i-2}, w_{i-1}, w_i) = \begin{cases} 1 & \text{if bigram is } \texttt{(dog,laughs)} \\ 0 & \text{otherwise} \end{cases}$$

$$f_3(w_{i-2}, w_{i-1}, w_i) = \begin{cases} 1 & \text{if unigram is } \texttt{(laughs)} \\ 0 & \text{otherwise} \end{cases}$$

$$q(w_i \mid w_{i-2}, w_{i-1}) = \frac{e^{f(w_{i-2}, w_{i-1}, w_i) \cdot v}}{\sum_w e^{f(w_{i-2}, w_{i-1}, w) \cdot v}}$$

Experiments with Gaussian Priors

- In regular (unregularized) log-linear models, if all n-gram features are included, then it's equivalent to maximum-likelihood estimates!

$$q(w_i \mid w_{i-2}, w_{i-1}) = \frac{Count(w_{i-2}, w_{i-1}, w_i)}{Count(w_{i-2}, w_{i-1})}$$

- [Chen and Rosenfeld, 1998]: with regularization, get very good results. Performs as well as or better than standardly used "discounting methods" (see lecture 2).
- Downside: computing $\sum_w e^{f(w_{i-2},w_{i-1},w)\cdot v}$ is SLOW.

Log Linear Models for Tagging

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 = 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 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

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

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

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:

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

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

A Trigram Log-Linear Tagger:

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

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

Independence assumptions

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

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

A Trigram Log-Linear Tagger:

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

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

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

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/DT 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 $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$
(Often **binary features** or **indicator functions** $f : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}$).

- Say we have m features f_k for $k = 1 \dots m$
$\Rightarrow$ 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, $\ldots\}$
- We have m features $f_k : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ for $k = 1 \ldots m$

For example:

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

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

$$\ldots$$

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

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

$\ldots$

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 ≤ 4, e.g.,

$$f_{101}(h,t) = \begin{cases} 1 & \text{if current word } w_i \text{ ends in } \texttt{ing} \text{ and } t = \texttt{VBG} \\ 0 & \text{otherwise} \end{cases}$$

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

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

- Contextual Features, e.g.,

$$f_{103}(h,t) = \begin{cases} 1 & \text{if } \langle t_{-2}, t_{-1}, t \rangle = \langle \text{DT, JJ, Vt} \rangle \\ 0 & \text{otherwise} \end{cases}$$

$$f_{104}(h,t) = \begin{cases} 1 & \text{if } \langle t_{-1}, t \rangle = \langle \text{JJ, Vt} \rangle \\ 0 & \text{otherwise} \end{cases}$$

$$f_{105}(h,t) = \begin{cases} 1 & \text{if } \langle t \rangle = \langle \text{Vt} \rangle \\ 0 & \text{otherwise} \end{cases}$$

$$f_{106}(h,t) = \begin{cases} 1 & \text{if previous word } w_{i-1} = \textit{the} \text{ and } t = \text{Vt} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{107}(h,t) = \begin{cases} 1 & \text{if next word } w_{i+1} = \textit{the} \text{ and } t = \text{Vt} \\ 0 & \text{otherwise} \end{cases}$$

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

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

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

$$v^* = \text{argmax}_v \left(\underbrace{\sum_i \log p(y_i|x_i; v)}_{Log-Likelihood} - \underbrace{\frac{\lambda}{2} \sum_k v_k^2}_{Regularizer} \right)$$

(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 $w_1 \ldots w_n$, find

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

We assume that p takes the form

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

(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

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

that is,

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

A Recursive Definition

Base case:

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

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

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

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

The Viterbi Algorithm with Backpointers

Input: a sentence $w_1 \ldots w_n$, 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 $u \in \mathcal{S}_{k-1}$, $v \in \mathcal{S}_k$,

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

- 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 $t_1 \ldots t_n$

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/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
```

⇒ "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" $\mid$ question$) =$
$e(\text{ 2.6)} \mid \text{question})\times$
$e(What \mid \text{question})\times$
$e(configuration \mid \text{question})\times$
$e(of \mid \text{question})\times$
$e(serial \mid \text{question})\times$
$\ldots$

- 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`

  $\Rightarrow$

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

- Use a language model again:

$$
\begin{aligned}
&p(\text{“2.6) What configuration of serial cable should I use”} \mid \text{question}) = \\
&e(\text{begins-with-number} \mid \text{question}) \times \\
&e(\text{contains-alphanum} \mid \text{question}) \times \\
&e(\text{contains-nonspace} \mid \text{question}) \times \\
&e(\text{contains-number} \mid \text{question}) \times \\
&e(\text{prev-is-blank} \mid \text{question}) \times
\end{aligned}
$$

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

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

  - Estimate

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

    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