

# Tagging Problems, and Hidden Markov Models

Michael Collins, Columbia University

# Overview

- ▶ The Tagging Problem
- ▶ Generative models, and the noisy-channel model, for supervised learning
- ▶ Hidden Markov Model (HMM) taggers
  - ▶ Basic definitions
  - ▶ Parameter estimation
  - ▶ The Viterbi algorithm

# 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/**VCN** 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/algorithm that maps new sentences to their tag sequences.

# Two Types of Constraints

Influential/JJ members/NNS of/IN the/DT House/NNP Ways/NNP and/CC Means/NNP Committee/NNP introduced/VBD legislation/NN that/WDT would/MD restrict/VB how/WRB the/DT new/JJ savings-and-loan/NN bailout/NN agency/NN can/MD raise/VB capital/NN ./.

- ▶ “Local”: e.g., *can* is more likely to be a modal verb MD rather than a noun NN
- ▶ “Contextual”: e.g., a noun is much more likely than a verb to follow a determiner
- ▶ Sometimes these preferences are in conflict:

*The trash can is in the garage*

# Overview

- ▶ The Tagging Problem
- ▶ Generative models, and the noisy-channel model, for supervised learning
- ▶ Hidden Markov Model (HMM) taggers
  - ▶ Basic definitions
  - ▶ Parameter estimation
  - ▶ The Viterbi algorithm



# Supervised Learning Problems

- ▶ We have training examples  $x^{(i)}, y^{(i)}$  for  $i = 1 \dots m$ . Each  $x^{(i)}$  is an input, each  $y^{(i)}$  is a label.
- ▶ Task is to learn a function  $f$  mapping inputs  $x$  to labels  $f(x)$

# Supervised Learning Problems

- ▶ We have training examples  $x^{(i)}, y^{(i)}$  for  $i = 1 \dots m$ . Each  $x^{(i)}$  is an input, each  $y^{(i)}$  is a label.
- ▶ Task is to learn a function  $f$  mapping inputs  $x$  to labels  $f(x)$
- ▶ Conditional models:
  - ▶ Learn a distribution  $p(y|x)$  from training examples
  - ▶ For any test input  $x$ , define  $f(x) = \arg \max_y p(y|x)$

# Generative Models

- ▶ We have training examples  $x^{(i)}, y^{(i)}$  for  $i = 1 \dots m$ . Task is to learn a function  $f$  mapping inputs  $x$  to labels  $f(x)$ .

# Generative Models

- ▶ We have training examples  $x^{(i)}, y^{(i)}$  for  $i = 1 \dots m$ . Task is to learn a function  $f$  mapping inputs  $x$  to labels  $f(x)$ .
- ▶ Generative models:
  - ▶ Learn a distribution  $p(x, y)$  from training examples
  - ▶ Often we have  $p(x, y) = p(y)p(x|y)$

# Generative Models

- ▶ We have training examples  $x^{(i)}, y^{(i)}$  for  $i = 1 \dots m$ . Task is to learn a function  $f$  mapping inputs  $x$  to labels  $f(x)$ .
- ▶ Generative models:
  - ▶ Learn a distribution  $p(x, y)$  from training examples
  - ▶ Often we have  $p(x, y) = p(y)p(x|y)$
- ▶ Note: we then have

$$p(y|x) = \frac{p(y)p(x|y)}{p(x)}$$

where  $p(x) = \sum_y p(y)p(x|y)$

# Decoding with Generative Models

- ▶ We have training examples  $x^{(i)}, y^{(i)}$  for  $i = 1 \dots m$ . Task is to learn a function  $f$  mapping inputs  $x$  to labels  $f(x)$ .

# Decoding with Generative Models

- ▶ We have training examples  $x^{(i)}, y^{(i)}$  for  $i = 1 \dots m$ . Task is to learn a function  $f$  mapping inputs  $x$  to labels  $f(x)$ .
- ▶ Generative models:
  - ▶ Learn a distribution  $p(x, y)$  from training examples
  - ▶ Often we have  $p(x, y) = p(y)p(x|y)$

# Decoding with Generative Models

- ▶ We have training examples  $x^{(i)}, y^{(i)}$  for  $i = 1 \dots m$ . Task is to learn a function  $f$  mapping inputs  $x$  to labels  $f(x)$ .
- ▶ Generative models:
  - ▶ Learn a distribution  $p(x, y)$  from training examples
  - ▶ Often we have  $p(x, y) = p(y)p(x|y)$
- ▶ Output from the model:

$$\begin{aligned} f(x) &= \arg \max_y p(y|x) \\ &= \arg \max_y \frac{p(y)p(x|y)}{p(x)} \\ &= \arg \max_y p(y)p(x|y) \end{aligned}$$



# Overview

- ▶ The Tagging Problem
- ▶ Generative models, and the noisy-channel model, for supervised learning
- ▶ Hidden Markov Model (HMM) taggers
  - ▶ Basic definitions
  - ▶ Parameter estimation
  - ▶ The Viterbi algorithm

# Hidden Markov Models

- ▶ We have an input sentence  $x = x_1, x_2, \dots, x_n$   
( $x_i$  is the  $i$ 'th word in the sentence)
- ▶ We have a tag sequence  $y = y_1, y_2, \dots, y_n$   
( $y_i$  is the  $i$ 'th tag in the sentence)
- ▶ We'll use an HMM to define

$$p(x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n)$$

for any sentence  $x_1 \dots x_n$  and tag sequence  $y_1 \dots y_n$  of the same length.

- ▶ Then the most likely tag sequence for  $x$  is

$$\arg \max_{y_1 \dots y_n} p(x_1 \dots x_n, y_1, y_2, \dots, y_n)$$

# Trigram Hidden Markov Models (Trigram HMMs)

For any sentence  $x_1 \dots x_n$  where  $x_i \in \mathcal{V}$  for  $i = 1 \dots n$ , and any tag sequence  $y_1 \dots y_{n+1}$  where  $y_i \in \mathcal{S}$  for  $i = 1 \dots n$ , and  $y_{n+1} = \text{STOP}$ , the joint probability of the sentence and tag sequence is

$$p(x_1 \dots x_n, y_1 \dots y_{n+1}) = \prod_{i=1}^{n+1} q(y_i | y_{i-2}, y_{i-1}) \prod_{i=1}^n e(x_i | y_i)$$

where we have assumed that  $x_0 = x_{-1} = *$ .

Parameters of the model:

- ▶  $q(s|u, v)$  for any  $s \in \mathcal{S} \cup \{\text{STOP}\}$ ,  $u, v \in \mathcal{S} \cup \{*\}$
- ▶  $e(x|s)$  for any  $s \in \mathcal{S}$ ,  $x \in \mathcal{V}$

## An Example

If we have  $n = 3$ ,  $x_1 \dots x_3$  equal to the sentence *the dog laughs*, and  $y_1 \dots y_4$  equal to the tag sequence D N V STOP, then

$$\begin{aligned} & p(x_1 \dots x_n, y_1 \dots y_{n+1}) \\ = & q(D|*, *) \times q(N|*, D) \times q(V|D, N) \times q(\text{STOP}|N, V) \\ & \times e(\textit{the}|D) \times e(\textit{dog}|N) \times e(\textit{laughs}|V) \end{aligned}$$

- ▶ STOP is a special tag that terminates the sequence
- ▶ We take  $y_0 = y_{-1} = *$ , where  $*$  is a special “padding” symbol

## Why the Name?

$$p(x_1 \dots x_n, y_1 \dots y_n) = \underbrace{q(\text{STOP} | y_{n-1}, y_n) \prod_{j=1}^n q(y_j | y_{j-2}, y_{j-1})}_{\text{Markov Chain}} \\ \times \underbrace{\prod_{j=1}^n e(x_j | y_j)}_{x_j \text{'s are observed}}$$

# Overview

- ▶ The Tagging Problem
- ▶ Generative models, and the noisy-channel model, for supervised learning
- ▶ Hidden Markov Model (HMM) taggers
  - ▶ Basic definitions
  - ▶ Parameter estimation
  - ▶ The Viterbi algorithm

## Smoothed Estimation

$$\begin{aligned} q(\mathbf{Vt} \mid \mathbf{DT}, \mathbf{JJ}) = & \lambda_1 \times \frac{\text{Count}(\mathbf{Dt}, \mathbf{JJ}, \mathbf{Vt})}{\text{Count}(\mathbf{Dt}, \mathbf{JJ})} \\ & + \lambda_2 \times \frac{\text{Count}(\mathbf{JJ}, \mathbf{Vt})}{\text{Count}(\mathbf{JJ})} \\ & + \lambda_3 \times \frac{\text{Count}(\mathbf{Vt})}{\text{Count}()} \end{aligned}$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1, \quad \text{and for all } i, \lambda_i \geq 0$$

$$e(\text{base} \mid \mathbf{Vt}) = \frac{\text{Count}(\mathbf{Vt}, \text{base})}{\text{Count}(\mathbf{Vt})}$$

## Dealing with Low-Frequency Words: An Example

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



# Dealing with Low-Frequency Words

A common method is as follows:

- ▶ **Step 1:** Split vocabulary into two sets

Frequent words = words occurring  $\geq 5$  times in training

Low frequency words = all other words

- ▶ **Step 2:** Map low frequency words into a small, finite set, depending on prefixes, suffixes etc.

# Dealing with Low-Frequency Words: An Example

[[Bikel et. al 1999](#)] (**named-entity recognition**)

Word class	Example	Intuition
twoDigitNum	90	Two digit year
fourDigitNum	1990	Four digit year
containsDigitAndAlpha	A8956-67	Product code
containsDigitAndDash	09-96	Date
containsDigitAndSlash	11/9/89	Date
containsDigitAndComma	23,000.00	Monetary amount
containsDigitAndPeriod	1.00	Monetary amount, percentage
othernum	456789	Other number
allCaps	BBN	Organization
capPeriod	M.	Person name initial
firstWord	first word of sentence	no useful capitalization information
initCap	Sally	Capitalized word
lowercase	can	Uncapitalized word
other	,	Punctuation marks, all other words

# Dealing with Low-Frequency Words: An Example

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



firstword/NA soared/NA at/NA initCap/SC Co./CC ,/NA easily/NA  
lowercase/NA forecasts/NA on/NA initCap/SL Street/CL ,/NA as/NA  
their/NA CEO/NA Alan/SP initCap/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

...

# Overview

- ▶ The Tagging Problem
- ▶ Generative models, and the noisy-channel model, for supervised learning
- ▶ Hidden Markov Model (HMM) taggers
  - ▶ Basic definitions
  - ▶ Parameter estimation
  - ▶ The Viterbi algorithm

# The Viterbi Algorithm

Problem: for an input  $x_1 \dots x_n$ , find

$$\arg \max_{y_1 \dots y_{n+1}} p(x_1 \dots x_n, y_1 \dots y_{n+1})$$

where the  $\arg \max$  is taken over all sequences  $y_1 \dots y_{n+1}$  such that  $y_i \in \mathcal{S}$  for  $i = 1 \dots n$ , and  $y_{n+1} = \text{STOP}$ .

We assume that  $p$  again takes the form

$$p(x_1 \dots x_n, y_1 \dots y_{n+1}) = \prod_{i=1}^{n+1} q(y_i | y_{i-2}, y_{i-1}) \prod_{i=1}^n e(x_i | y_i)$$

Recall that we have assumed in this definition that  $y_0 = y_{-1} = *$ , and  $y_{n+1} = \text{STOP}$ .

# Brute Force Search is Hopelessly Inefficient

Problem: for an input  $x_1 \dots x_n$ , find

$$\arg \max_{y_1 \dots y_{n+1}} p(x_1 \dots x_n, y_1 \dots y_{n+1})$$

where the  $\arg \max$  is taken over all sequences  $y_1 \dots y_{n+1}$  such that  $y_i \in \mathcal{S}$  for  $i = 1 \dots n$ , and  $y_{n+1} = \text{STOP}$ .

# The Viterbi Algorithm

- ▶ Define  $n$  to be the length of the sentence
- ▶ Define  $S_k$  for  $k = -1 \dots n$  to be the set of possible tags at position  $k$ :

$$S_{-1} = S_0 = \{*\}$$
$$S_k = S \quad \text{for } k \in \{1 \dots n\}$$

- ▶ Define

$$r(y_{-1}, y_0, y_1, \dots, y_k) = \prod_{i=1}^k q(y_i | y_{i-2}, y_{i-1}) \prod_{i=1}^k e(x_i | y_i)$$

- ▶ Define a dynamic programming table

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

that is,

$$\pi(k, u, v) = \max_{\langle y_{-1}, y_0, y_1, \dots, y_k \rangle : y_{k-1}=u, y_k=v} r(y_{-1}, y_0, y_1 \dots y_k)$$

## An Example

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

The man saw the dog with the telescope



# 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  $v \in \mathcal{S}_k$ :

$$\pi(k, u, v) = \max_{w \in \mathcal{S}_{k-2}} (\pi(k-1, w, u) \times q(v|w, u) \times e(x_k|v))$$

## Justification for the Recursive Definition

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

$$\pi(k, u, v) = \max_{w \in \mathcal{S}_{k-2}} (\pi(k-1, w, u) \times q(v|w, u) \times e(x_k|v))$$

The man saw the dog with the telescope

# The Viterbi Algorithm

**Input:** a sentence  $x_1 \dots x_n$ , parameters  $q(s|u, v)$  and  $e(x|s)$ .

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

**Definition:**  $\mathcal{S}_{-1} = \mathcal{S}_0 = \{*\}$ ,  $\mathcal{S}_k = \mathcal{S}$  for  $k \in \{1 \dots n\}$

**Algorithm:**

► For  $k = 1 \dots n$ ,

► For  $u \in \mathcal{S}_{k-1}$ ,  $v \in \mathcal{S}_k$ ,

$$\pi(k, u, v) = \max_{w \in \mathcal{S}_{k-2}} (\pi(k-1, w, u) \times q(v|w, u) \times e(x_k|v))$$

► **Return**  $\max_{u \in \mathcal{S}_{n-1}, v \in \mathcal{S}_n} (\pi(n, u, v) \times q(\text{STOP}|u, v))$

# The Viterbi Algorithm with Backpointers

**Input:** a sentence  $x_1 \dots x_n$ , parameters  $q(s|u, v)$  and  $e(x|s)$ .

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

**Definition:**  $\mathcal{S}_{-1} = \mathcal{S}_0 = \{*\}$ ,  $\mathcal{S}_k = \mathcal{S}$  for  $k \in \{1 \dots n\}$

**Algorithm:**

► For  $k = 1 \dots n$ ,

► For  $u \in \mathcal{S}_{k-1}$ ,  $v \in \mathcal{S}_k$ ,

$$\pi(k, u, v) = \max_{w \in \mathcal{S}_{k-2}} (\pi(k-1, w, u) \times q(v|w, u) \times e(x_k|v))$$

$$bp(k, u, v) = \arg \max_{w \in \mathcal{S}_{k-2}} (\pi(k-1, w, u) \times q(v|w, u) \times e(x_k|v))$$

► Set  $(y_{n-1}, y_n) = \arg \max_{(u,v)} (\pi(n, u, v) \times q(\text{STOP}|u, v))$

► For  $k = (n-2) \dots 1$ ,  $y_k = bp(k+2, y_{k+1}, y_{k+2})$

► **Return** the tag sequence  $y_1 \dots y_n$

# The Viterbi Algorithm: Running Time

- ▶  $O(n|\mathcal{S}|^3)$  time to calculate  $q(s|u, v) \times e(x_k|s)$  for all  $k, s, u, v$ .
- ▶  $n|\mathcal{S}|^2$  entries in  $\pi$  to be filled in.
- ▶  $O(|\mathcal{S}|)$  time to fill in one entry
- ▶  $\Rightarrow O(n|\mathcal{S}|^3)$  time in total

## Pros and Cons

- ▶ Hidden markov model taggers are very simple to train (just need to compile counts from the training corpus)
- ▶ Perform relatively well (over 90% performance on named entity recognition)
- ▶ Main difficulty is modeling

$$e(\textit{word} \mid \textit{tag})$$

can be very difficult if “words” are complex