6.891: Lecture 7 (September 29, 2003)
Log-Linear Models

Overview

- Anatomy of a tagger
- An alternative: Hidden markov model taggers

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 or "program" that maps new sentences to their tag sequences.

Our Goal (continued)

- **A test data sentence:**

Influential members of the House Ways and Means Committee introduced legislation that would restrict how the new savings-and-loan bailout agency can raise capital , creating another potential obstacle to the government 's sale of sick thrifts .

- **Should be mapped to underlying tags:**

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 ,/, creating/VBG another/DT potential/JJ obstacle/NN to/TO the/DT government/NN 's/POS sale/NN of/IN sick/JJ thrifts/NNS ./.

- Our goal is to minimize the number of tagging errors on sentences not seen in the training set

Some Data Structures

- A **word** is a symbol that is a member of the word set, $\mathcal{V}$
  e.g.,$\mathcal{V}$ = set of all possible ascii strings
- A **tag** is a symbol that is a member of the tag set, $\mathcal{T}$
  e.g.,$\mathcal{T} = \{$NN,NNS,JJ,IN,Vt,. . .$\}$
- A **word sequence** S is an array of words
  $S.length =$ number of words in the array
  $S(j) = j$'th word in the array
- A **tag sequence** T is an array of tags
  $T.length =$ number of tags in the array
  $T(j) = j$'th tag in the array
- The **training data** D is an array of paired word/tag sequences
  $D.length =$ number of sequences in the training data
  $D.S_i =$ the i'th word sequence in training data
  $D.T_i =$ the i'th tag sequence in training data
  Note: $D.S_i.length = D.T_i.length$ for all i

- A **parameter vector** $\mathbf{W}$ is an array of reals (doubles)
  $W.length =$ number of parameters
  $W_k =$ the k'th parameter value

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
```

- Tags = <head>, <question>, <answer>
- Words = entire sentences, e.g.,
  `is to avoid the well known serial port chip bugs.  The`

Three Functions

- TRAIN is the training algorithm. $\texttt{TRAIN}(D)$ returns a **parameter vector $\mathbf{W}$**.

- TAGGER is the function which tags a new sentence. $\texttt{TAGGER}(S, \mathbf{W})$ returns a tag sequence T.

- $\Rightarrow$ All of the information from the training set is captured in the parameter vector $\mathbf{W}$

- PROB returns a probability distribution over the possible tags. i.e.,

$$\texttt{PROB}(S, j, t_{-2}, t_{-1}, \mathbf{W}) = P(\ \cdot\ \mid S, j, t_{-2}, t_{-1}, \mathbf{W})$$

Three Functions

- `TRAIN` and `PROB` can be implemented with log-linear models.

- How do we implement `TAGGER`$(S, \mathbf{W})$, given access to `PROB`?

Log-Linear Taggers: Independence Assumptions

- The input sentence S, with length $n = S.length$, has $|\mathcal{T}|^n$ possible tag sequences.
- Each tag sequence T has a conditional probability

$$P(T \mid S) = \prod_{j=1}^{n} P(T(j) \mid S, j, T(1) \ldots T(j-1)) \quad \text{Chain rule}$$

$$= \prod_{j=1}^{n} P(T(j) \mid S, j, T(j-2), T(j-1)) \quad \text{Independence assumptions}$$

- We have a black-box PROB which can compute the $P(T(j) \mid S, j, T(j-2), T(j-1))$ terms
- How do we compute $\textsf{TAGGER}(S, \mathbf{W})$ where

$$\begin{aligned} \textsf{TAGGER}(S, \mathbf{W}) &= \text{argmax}_{T \in \mathcal{T}^n} P(T \mid S) \\ &= \text{argmax}_{T \in \mathcal{T}^n} \log P(T \mid S) \end{aligned}$$

The Viterbi Algorithm

- Define $n = S.length$, i.e., the length of the sentence
- Define a dynamic programming table

  $\pi[i, t_{-2}, t_{-1}]$ = maximum log probability of a tag sequence ending in tags t_{-2}, t_{-1} at position i

- Our goal is to calculate $\max_{t_{-2}, t_{-1} \in \mathcal{T}} \pi[n, t_{-2}, t_{-1}]$

The Viterbi Algorithm: Recursive Definitions

- **Base case:**

$$\begin{aligned} \pi[0, *, *] &= \log 1 = 0 \\ \pi[0, t_{-2}, t_{-1}] &= \log 0 = -\infty \text{ for all other } t_{-2}, t_{-1} \end{aligned}$$

here $*$ is a special tag padding the beginning of the sentence.

- **Recursive case:** for $i = 1 \ldots S.length$, for all t_{-2}, t_{-1},

$$\pi[i, t_{-2}, t_{-1}] = \max_{t \in \mathcal{T} \cup \{*\}} \{\pi[i-1, t, t_{-2}] + \log P(t_{-1} \mid S, i, t, t_{-2})\}$$

Backpointers allow us to recover the max probability sequence:

$$\text{BP}[i, t_{-2}, t_{-1}] = \text{argmax}_{t \in \mathcal{T} \cup \{*\}} \{\pi[i-1, t, t_{-2}] + \log P(t_{-1} \mid S, i, t, t_{-2})\}$$

The Viterbi Algorithm: Running Time

- $O(n|\mathcal{T}|^3)$ time to calculate $\log P(t_{-1}|S, i, t, t_{-2})$ for all i, t, t_{-2}, t_{-1}.
- $n|\mathcal{T}|^2$ entries in π to be filled in.
- $O(\mathcal{T})$ time to fill in one entry
  (assuming $O(1)$ time to look up $\log P(t_{-1} \mid S, i, t, t_{-2})$)
- $\Rightarrow O(n|\mathcal{T}|^3)$ time

Coming Next...

- How to implement `PROB` and `TRAIN`

A New Data Structure: Sparse Binary Arrays

- A **sparse array** A is an array of integers
  $A.length =$ length of the array
  $A(k) =$ the k'th integer in the array

- For example, the binary vector

  $$1000100000010010$$

  is represented as

  $A.length = 4$
  $A(1) = 1, A(2) = 5, A(3) = 11, A(4) = 14$

Why the Need for Sparse Binary Arrays?

From last lecture:

$$\phi_1(h, t) = \begin{cases} 1 & \text{if current word } w_i \text{ is } \texttt{base} \text{ and } t = \texttt{Vt} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_2(h, t) = \begin{cases} 1 & \text{if current word } w_i \text{ ends in } \texttt{ing} \text{ and } t = \texttt{VBG} \\ 0 & \text{otherwise} \end{cases}$$

$$\ldots$$

$$\phi_1(\langle \text{JJ, DT, } \langle \text{ Hispaniola, } \ldots \rangle, 6 \rangle, \text{Vt}) = 1$$
$$\phi_2(\langle \text{JJ, DT, } \langle \text{ Hispaniola, } \ldots \rangle, 6 \rangle, \text{Vt}) = 0$$
$$\ldots$$

$$\phi(\langle \text{JJ, DT, } \langle \text{ Hispaniola, } \ldots \rangle, 6 \rangle, \text{Vt}) = 10010110010011100110$$

Operations on Sparse Binary Arrays

- INSERT$(A, 12)$:

  before: $A.length = 4$
  $A(1) = 1, A(2) = 5, A(3) = 11, A(4) = 14$

  after: $A.length = 5$
  $A(1) = 1, A(2) = 5, A(3) = 11, A(4) = 14$, $A(5) = 12$

- Inner products with a parameter vector
  IP$(A, \mathbf{W})$ returns the value of the inner product

- Adding a sparse vector to a parameter vector
  ADD$(\mathbf{W}, A, \beta)$ sets $\mathbf{W} \leftarrow \mathbf{W} + \beta \times A$

- FEATURE_VECTOR$(S, j, t_{-2}, t_{-1}, t)$ returns a sparse feature array

Implementing FEATURE_VECTOR

- Intermediate step: map history/tag pair to set of **feature strings**

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

e.g., Ratnaparkhi's features:

"TAG=Vt;Word=base"
"TAG=Vt;TAG-1=JJ"
"TAG=Vt;TAG-1=JJ;TAG-2=DT"
"TAG=Vt;SUFF1=e"
"TAG=Vt;SUFF2=se"
"TAG=Vt;SUFF3=ase"
"TAG=Vt;WORD-1=important"
"TAG=Vt;WORD+1=from"

Implementing FEATURE_VECTOR

- Next step: match strings to integers through a hash table

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

e.g., Ratnaparkhi's features:

"TAG=Vt;Word=base"	$\rightarrow$ 1315
"TAG=Vt;TAG-1=JJ"	$\rightarrow$ 17
"TAG=Vt;TAG-1=JJ;TAG-2=DT"	$\rightarrow$ 32908
"TAG=Vt;SUFF1=e"	$\rightarrow$ 459
"TAG=Vt;SUFF2=se"	$\rightarrow$ 1000
"TAG=Vt;SUFF3=ase"	$\rightarrow$ 1509
"TAG=Vt;WORD-1=important"	$\rightarrow$ 1806
"TAG=Vt;WORD+1=from"	$\rightarrow$ 300

In this case, sparse array is:
$A.length = 8, A(1...8) = \{1315, 17, 32908, 459, 1000, 1509, 1806, 300\}$

Implementing FEATURE_VECTOR

FEATURE_VECTOR$(S, j, t_{-2}, t_{-1}, t)$

$A.length = 0$

$String =$ "TAG=t;WORD=$S(j)$"
$tmp = hash(String)$
INSERT(A, tmp)

$String =$ "TAG=t;TAG-1=t_{-1}"
$tmp = hash(String)$
INSERT(A, tmp)

$String =$ "TAG=t;TAG-1=t_{-1};TAG-2=t_{-2}"
$tmp = hash(String)$
INSERT(A, tmp)

Return A

Implementing PROB

PROB$(S, j, t_{-2}, t_{-1}, \mathbf{W})$

$Z = 0$

For each $t \in \mathcal{T}$

$A =$ FEATURE_VECTOR$(S, j, t_{-2}, t_{-1}, t)$

$ip =$ IP$(A, \mathbf{W})$

$Score(t) = e^{ip}$

$Z = Z + Score(t)$

For each $t \in \mathcal{T}$

$Prob(t) = Score(t)/Z$

Return $Prob$

Implementing TRAIN

- Possible training algorithms: iterative scaling, conjugate gradient methods
- Possible loss functions: log-likelihood, log-likelihood plus gaussian prior
- All of these methods were iterative
- Methods all require following functions of D and $\mathbf{W}$ at each iteration:

$$L(\mathbf{W}) = \sum_i \log P(y_i \mid x_i, \mathbf{W}) \qquad \text{Log-likelihood}$$

$$H = \sum_i \phi(x_i, y_i) \qquad \text{Empirical counts}$$

$$E(\mathbf{W}) = \sum_i \sum_y P(y \mid x, \mathbf{W}) \phi(x_i, y) \qquad \text{Expected counts}$$

- Next: implement STATISTICS$(D, \mathbf{W})$ which returns $L(\mathbf{W}), E(\mathbf{W}), H$

Implementing STATISTICS

STATISTICS$(D, \mathbf{W})$
- $H = 0, E = 0, L = 0$
- $m = D.length$
- for $i = 1 \ldots m$
  - $n = D.S_i.length$
    - for $j = 1 \ldots n$
      - $t = D.T_i(j)$
      - $t_{-1} = D.T_i(j-1)$
      - $t_{-2} = D.T_i(j-2)$
      - $A =$ FEATURE_VECTOR$(D.S_i, j, t_{-2}, t_{-1}, t)$
      - ADD$(H, A, 1)$
      - $Prob =$ PROB$(D.S_i, j, t_{-2}, t_{-1}, \mathbf{W})$
      - $L = L + \log Prob(t)$
      - for each $t \in \mathcal{T}$
        - $A =$ FEATURE_VECTOR$(D.S_i, j, t_{-2}, t_{-1}, t)$
        - $p = Prob(t)$
        - ADD(E, A, p)
- Return (L, E, H)

Implementing TRAIN with Iterative Scaling

TRAIN(D)

$\mathbf{W} = 0$

DO:

$(L, E, H) =$ STATISTICS$(D, \mathbf{W})$

for $k = 1 \ldots \mathbf{W}.length$

$\mathbf{W}_k = \mathbf{W}_k + \frac{1}{C} \log \frac{H_k}{E_k}$

UNTIL:

L does not change significantly

Summary

Implemented following functions:

- FEATURE_VECTOR$(S, j, t_{-2}, t_{-1}, t)$ maps history/tag pair to sparse array
- PROB$(S, j, t_{-2}, t_{-1}, \mathbf{W})$ returns a probability distribution over tags
- STATISTICS$(D, \mathbf{W})$ maps training set, parameter vector, to $(L, E(W), H)$.
- TRAIN(D) returns optimal parameters $\mathbf{W}$
- TAGGER$(S, \mathbf{W})$ returns most likely tag sequence under parameters $\mathbf{W}$

Case Studies

- Ratnaparkhi's part-of-speech tagger
- McCallum et. al work on FAQ segmentation

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
indented-5-to-10
more-than-one-third-space
only-punctuation
prev-is-blank
prev-begins-with-ordinal
shorter-than-30

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

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

          Pins 1, 4, and 8 must be connected together inside
        is to avoid the well known serial port chip bugs. The
```

⇒ “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”

A Third Case Study: Tagging Arabic

```
NOUN_PROP lwng
NOUN_PROP byt$
PUNC -LRB-
DET+NOUN_PROP+NSUFF_FEM_PL AlwlAyAt
DET+ADJ+NSUFF_FEM_SG AlmtHdp
...
VERB_PERFECT tgyr
PREP fy
NOUN+NSUFF_FEM_SG HyAp
DET+NOUN Almt$rd
NOUN_PROP styfn
NOUN_PROP knt
```

A Third Case Study: Tagging Arabic

```
NOUN_PROP lwng
NOUN_PROP byt$
PUNC -LRB-
DET+NOUN_PROP+NSUFF_FEM_PL AlwlAyAt

DET+NOUN_PROP+NSUFF_FEM_PL AlwlAyAt
```

$\Rightarrow$

```
tag=DET+NOUN_PROP+NSUFF_FEM_PL;word=AlwlAyAt

tag-contains=DET;word=AlwlAyAt

tag-contains=DET;pref=Al

tag-contains=DET;pref=Alw

tag-contains=NOUN;word=AlwlAyAt

tag-contains=DET;tag-2-contains=NOUN
```

Overview

- Anatomy of a tagger
- An alternative: Hidden markov model taggers

Log-Linear Taggers: Independence Assumptions

- Each tag sequence T has a conditional probability

$$P(T \mid S) = \prod_{j=1}^{n} P(T(j) \mid S, j, T(1) \ldots T(j-1)) \quad \text{Chain rule}$$

$$= \prod_{j=1}^{n} P(T(j) \mid S, j, T(j-2), T(j-1)) \quad \text{Independence assumptions}$$

- TAGGER involves search for most likely tag sequence:

$$\text{TAGGER}(S, \mathbf{W}) = \text{argmax}_{T \in \mathcal{T}^n} \log P(T \mid S)$$

Hidden Markov Models

- Model $P(T, S)$ rather than $P(T \mid S)$
- Then

$$P(T \mid S) = \frac{P(T, S)}{\sum_T P(T, S)}$$

$$\begin{aligned} \textsf{TAGGER}(S, \mathbf{W}) &= \text{argmax}_{T \in \mathcal{T}^n} \log P(T \mid S) \\ &= \text{argmax}_{T \in \mathcal{T}^n} \log P(T, S) \end{aligned}$$

How to model $P(T, S)$?

$$P(T, S) = \prod_{j=1}^{n} (\; P(T_j \mid S_1 \ldots S_{j-1}, T_1 \ldots T_{j-1}) \times P(S_j \mid S_1 \ldots S_{j-1}, T_1 \ldots T_j))$$

Chain rule

$$= \prod_{j=1}^{n} (P(T_j \mid T_{j-2}, T_{j-1}) \times P(S_j \mid T_j))$$

Independence assumptions

Why the Name?

$$P(T,S) = \underbrace{\prod_{j=1}^{n} P(T_j \mid T_{j-2}, T_{j-1})}_{\text{Hidden Markov Chain}} \times \underbrace{\prod_{j=1}^{n} P(S_j \mid T_j)}_{S_j\text{'s are observed}}$$

How to model $P(T, S)$?

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

"Score" for tag `Vt`:

$$P(\text{Vt} \mid \text{DT, JJ}) \times P(\text{base} \mid \text{Vt})$$

Smoothed Estimation

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

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

Dealing with Low-Frequency Words

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

  Frequent words = words occurring ≥ 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] **An Algorithm that Learns What's in a Name**

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
. . .

The Viterbi Algorithm: Recursive Definitions

- **Base case:**

$$\begin{aligned} \pi[0, *, *] &= \log 1 = 0 \\ \pi[0, t_{-2}, t_{-1}] &= \log 0 = -\infty \text{ for all other } t_{-2}, t_{-1} \end{aligned}$$

here $*$ is a special tag padding the beginning of the sentence.

- **Recursive case:** for $i = 1 \ldots S.length$, for all t_{-2}, t_{-1},

$$\pi[i, t_{-2}, t_{-1}] = \max_{t \in \mathcal{T} \cup \{*\}} \{\pi[i-1, t, t_{-2}] + \log \textsf{PROB}(S, i, t, t_{-2}, t_{-1})\}$$

Backpointers allow us to recover the max probability sequence:

$$\text{BP}[i, t_{-2}, t_{-1}] = \text{argmax}_{t \in \mathcal{T} \cup \{*\}} \{\pi[i-1, t, t_{-2}] + \log \textsf{PROB}(S, i, t, t_{-2}, t_{-1})\}$$

Only difference is that $\textsf{PROB}(S, i, t, t_{-2}, t_{-1})$ **returns**

$$P(t_{-1} \mid t, t_{-2}) \times P(S_i \mid t_{-1})$$

rather than

$$P(t_{-1} \mid S, i, t, t_{-2})$$

Pros and Cons

- Hidden markov model taggers are very simple to train (compile counts from the training corpus)

- Perform relatively well (over 90% performance on named entities)

- Main difficulty is modeling

$$P(word \mid tag)$$

  can be very difficult if “words” are complex

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: McCallum et. al

```
<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$) =$
$P($ 2.6$) \mid$ question$)\times$
$P(What \mid$ question$)\times$
$P(configuration \mid$ question$)\times$
$P(of \mid$ question$)\times$
$P(serial \mid$ question$)\times$
$\ldots$

- i.e. have a **language model** for each tag

FAQ Segmentation: McCallum et. al

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
indented-5-to-10
more-than-one-third-space
only-punctuation
prev-is-blank
prev-begins-with-ordinal
shorter-than-30

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-nonspac`

- Use a language model again:

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

FAQ Segmentation: McCallum et. al

Method	COAP	SegPrec	SegRec
ME-Stateless	0.520	0.038	0.362
TokenHMM	0.865	0.276	0.140
FeatureHMM	0.941	0.413	0.529
MEMM	0.965	0.867	0.681