

# Natural Language Processing

Creating Computerized Structure  
from Human Language

# Tonight's Topics

## What I will cover

Typical Analysis Pipeline

Common Algorithms

Sample Frameworks

UIMA (Java)

Sample Libraries

OpenNLP

Stanford

cTakes

Gate (Java)

NLTK (Python)

Resources

MOOCs, Links, Books

## What I won't cover

Search Tasks

Search Techniques

Search Technologies

Advanced Algorithms

Hopefully in future  
presentation

Details about specific  
Use Cases

# NLP Tasks

([http://en.wikipedia.org/wiki/Natural\\_language\\_processing](http://en.wikipedia.org/wiki/Natural_language_processing))

Automatic summarization	Sentiment analysis
Coreference resolution	Speech recognition
Discourse analysis	Speech segmentation
Machine translation	Topic segmentation and recognition
Morphological segmentation	Word segmentation
Named entity recognition (NER)	Word sense disambiguation
Natural language generation	Information extraction (IE)
Natural language understanding	Speech processing
Optical character recognition (OCR)	Stemming
Part-of-speech tagging	Text simplification
Parsing	Text-to-speech
Question answering	Text-proofing
Relationship extraction	Natural language search
Sentence breaking (also known as sentence boundary disambiguation)	Query expansion
	Automated essay scoring
	<u>Truecasing</u>

# Sample Analysis Pipeline

## Separating the Text

Tokenizer - Identifies Individual Words, Punctuation, Tags, etc.

Sentence Detector - Identifies Sentences based on ML, RegEx, Rules, etc.

## Identifying Grammatical Purpose of the Text

Lemmatizer - Set of all forms of the words

Stemmer - root of words (runs -> run)

POS Tagger - Parts Of Speech (Noun, Verb, Adj, etc.)

## Assign Meaning to Words

Named Entity Recognizer(NER) - (MEMM)

Gazzetter - Word / Phrase Match

Rules (e.g., UIMA RUTA) - RegEx / Rule based Annotation

# Sample Analysis Pipeline pt. 2

## Identify Phrases or Context Based Annotation

Chunker - Shallow (non-recursive) Identity of Phrases

(S (NP The/DT cat/NN) sat/VBD on/IN (NP the/DT mat/NN the/DT dog/NN) chewed/VBD ./.)

Coreference - Identifies Coreference of entities

“The patient didn’t feel pain until after their game was over.” Their game -> Patient’s game.

Sentiment Analysis - Determines Emotional Context

“I was very happy with this product.” (positive)

“I hate this and would never buy from this company again.” (negative)

Negation (e.g., NegEx) - Determines Affirmed / Negated

“The patient did not have a fever.”

# Common Frameworks

## Java

[Apache UIMA](#) - Unstructured Information Management

[GATE](#) - General Architecture for Text Engineering

[Apache cTAKES](#) - Clinical Text Analysis and Knowledge Extraction System - UIMA + Custom Components

## Libraries

[OpenNLP](#)

[Stanford Parser](#)

[DKPro](#)

## Python

[NLTK](#) - Natural Language Toolkit

# UIMA NLP Concepts

## Type System

Defines Class Model

Includes Annotation Types and Analysis Engines

## Collection Producer / Reader

Creates CAS objects with associated Sofa+Default View

## Pipeline / Aggregate

Collection of Analysis Engines that Enhance CAS with Annotations associated with Views of Sofas

## Collection Consumers / XMI Writer

# Common NLP Objects

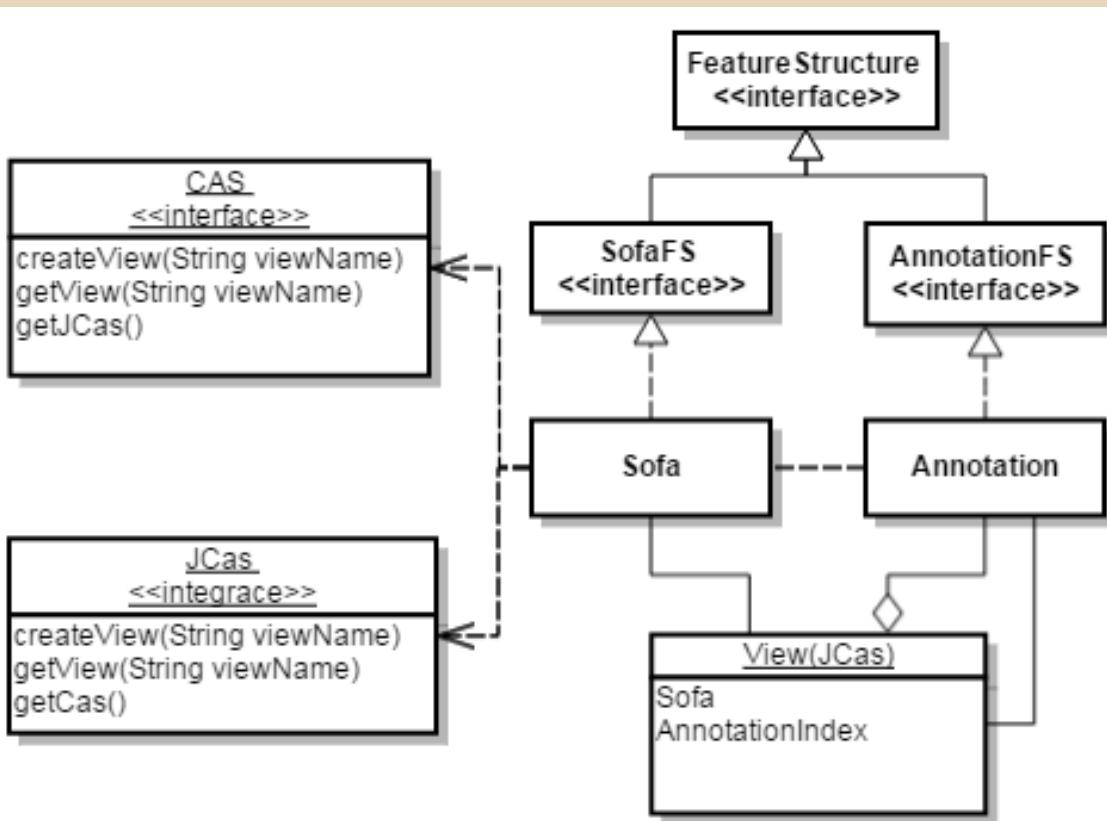
## Type System

Class model

## Analysis Model

(e.g., CAS - Common Analysis System)

## Object model



## Text Document

(e.g., Sofa - Subject of Analysis)

Indexed Character Position

Collection of Views

## Typed Annotations

(e.g., AnnotationFS)

Spans Text Range (begin,end)

Features (Typed Attributes)

## Views of Document

Collection of Annotations

## Annotator

(e.g., AnalysisEngine)

Analysis Component

Creates / Modifies

## Annotations

Adds Features to Annotations

# Some Algorithms Common w/ NLP

## Hidden Markov Model (HMM) Conditional Random Fields

Predict current state on  
only most recent history

Uni-directed Graph

## Maximum Entropy Markov Model (MEMM)

Predict Labels based on  
Conditional Probability

Predicts Sequences of Labels  
for Input Samples

## Perceptron

Linear Model, non-binary  
Output, Linear Classifier

## Coreference

Occurrence of terms together

## Probabilistic Context Free Grammar (PCFG)

Directed Graph from One  
State to other possible states

## Support Vector Machines (SVM)

Vector space based large-  
margin classifier

Each transition weighted with  
a probability

# Coursera NLP Courses

coursera

Courses Specializations Institutions About David Taylor

COLUMBIA UNIVERSITY  
IN THE CITY OF NEW YORK

## Natural Language Processing

Have you ever wondered how to build a system that automatically translates between languages? Or a system that can understand natural language instructions from a human? This class will cover the fundamentals of mathematical and computational models of language, and the application of these models to key problems in natural language processing.



### About the Course

Natural language processing (NLP) deals with the application of computational models to text or speech data. Application areas within NLP include automatic (machine) translation between languages; dialogue systems, which allow a human to interact with a machine using natural language; and information extraction, where the goal is to transform unstructured text into structured (database) representations that can be searched and browsed in flexible ways. NLP technologies are having a dramatic impact on the way people interact with computers, on the way people interact with each other through the use of language, and on the way people access the vast amount of linguistic data now in electronic form. From a scientific viewpoint, NLP involves fundamental questions of how to structure formal models (for example statistical models) of natural language phenomena, and of how to design algorithms that implement these models.

In this course you will study mathematical and computational models of language, and the application of these models to key problems in natural language processing. The course has a focus on machine learning methods, which are widely used in modern NLP systems: we will cover formalisms such as hidden Markov models, probabilistic context-free grammars, log-linear models, and statistical models for machine translation. The curriculum closely follows a course currently taught by Professor Collins at Columbia University, and previously taught at MIT.

### Sessions

Feb 24, 2013 - May 4th 2013

[Go to Course](#)

### Course at a Glance

- 10 weeks of study
- 8-10 hours/week
- English

### Instructors



Michael Collins  
Columbia University

### Categories

Computer Science: Artificial Intelligence

coursera

Courses Specializations Institutions About David Taylor

M UNIVERSITY OF MICHIGAN

## Introduction to Natural Language Processing

This course provides an introduction to the field of Natural Language Processing, including topics like Parsing, Semantics, Question Answering, and Sentiment Analysis.



### About the Course

### Sessions

## Stanford

## Natural Language Processing

In this class, you will learn fundamental algorithms and mathematical models for processing natural language, and how these can be used to solve practical problems.

[Preview Lectures](#)



### Sessions

Future Sessions

[Add to Watchlist](#)

### Course at a Glance

- 8-10 hours/week
- English

### Instructors



Dan Jurafsky  
Stanford University



Christopher Manning  
Stanford University

### About the Course

This course covers a broad range of topics in natural language processing, including word and sentence tokenization, text classification and sentiment analysis, spelling correction, information extraction, parsing, meaning extraction, and question answering. We will also introduce the underlying theory from probability, statistics, and machine learning that are crucial for the field, and cover fundamental algorithms like n-gram language modeling, naive bayes and maxent classifiers, sequence models like Hidden Markov Models, probabilistic dependency and constituent parsing, and vector-space models of meaning.

We are offering this course on Natural Language Processing free and online to students worldwide, continuing Stanford's exciting forays into large scale online instruction. Students have access to screencast lecture videos, are given quiz questions, assignments and exams, receive regular feedback on progress, and can participate in a discussion forum. Those who successfully complete the course will receive a statement of accomplishment. Taught by Professors Jurafsky and Manning, the curriculum draws from Stanford's courses in Natural Language Processing. You will need a decent internet connection for accessing course materials, but should be able to watch the videos on your smartphone.

# Hidden Markov Model (Trigram HMM)

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  $s \in \mathcal{S}$ ,  $x \in \mathcal{V}$

Screen Shots from Coursera NLP Course:  
<https://www.coursera.org/course/nlangp>

$p(w_1, w_2, w_3, t_1, t_2, t_3, t_4)$ . Joint probability

$$t_{[1:n]}^* = \arg \max t_{[1:n]} p(w_1, w_2, w_3, t_1, t_2, t_3, t_4)$$

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

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

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

# Hidden Markov Model (HMM) cont.

HIDDEN MARKOV MODELS

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

$p(x_1, \dots, x_n | y_1, \dots, y_{n+r})$

$p(x, y) = p(y) \times p(x|y)$

Hidden because words are observed but Markov Chain is evaluating the derived tags (states) are hidden.

# Hidden Markov Models (HMM) cont.

## Parameter Estimation

$$q(V_t | D_t, J_t) = \lambda_1 \times \frac{\text{Count}(D_t, J_t, V_t)}{\text{Count}(D_t, J_t)} + \lambda_2 \times \frac{\text{Count}(J_t, V_t)}{\text{Count}(J_t)} + \lambda_3 \times \frac{\text{Count}(V_t)}{\text{Count}()}$$

TRIGRAM  
ML ESTIMATE

BIGRAM  
ML ESTIMATE

UNIGRAM  
ML ESTIMATE

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

$$e(\text{base} | V_t) = \frac{\text{Count}(V_t, \text{base})}{\text{Count}(V_t)}$$

$e(x|y) = 0$  for all  $y$ ,

if  $x$  vs never seen  
in the training data

# Low Frequency Words Strategy

{ 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

# Maximum Entropy Markov Model (MEMM)

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

Conditional Probability (HMM Joint Probability) of Tag sequence given Word Sequence

Most likely Tag Sequence  $t_{[1:n]}^*$  given  $w_{[1:n]}$  is =  $\text{argmax } t_{[1:n]} p(t_{[1:n]} | w_{[1:n]})$

## Log-Linear Models for Tagging

- ▶ We have an input sentence  $w_{[1:n]} = w_1, w_2, \dots, w_n$   
( $w_i$  is the  $i$ 'th word in the sentence)
- ▶ We have a tag sequence  $t_{[1:n]} = t_1, t_2, \dots, t_n$   
( $t_i$  is the  $i$ 'th tag in the sentence)
- ▶ We'll use an log-linear model to define

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

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

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

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

# Maximum Entropy Markov Model (MEMM)

## Log-Linear Model Overview

### Log-Linear Models

- ▶ We have some input domain  $\mathcal{X}$ , and a finite label set  $\mathcal{Y}$ . Aim is to provide a conditional probability  $p(y | x)$  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  $k = 1 \dots m$   
⇒ A feature vector  $\underline{f}(x, y) \in \mathbb{R}^m$  for any  $x \in \mathcal{X}$  and  $y \in \mathcal{Y}$ .

- ▶ We also have a **parameter vector**  $\underline{v} \in \mathbb{R}^m$

- ▶ We define

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

$$m = 3$$

$$\underline{f}(x, y) = \langle 1, 0, 1 \rangle$$

$$\underline{v} = \langle 1, 2, 3 \rangle$$

$$\underline{v} \cdot \underline{f}(x, y) =$$

$$1 \times 1 + 0 \times 2 + 1 \times 3$$

$$= 4$$

# Maximum Entropy Markov Model (MEMM)

## Recap: Feature Vector Representations in Log-Linear Models

$\langle t_{-2}, t_{-1}, w_1, \dots, w_n, i \rangle$

$\{D, N, V\}$



- We have some input domain  $\underline{\mathcal{X}}$ , and a finite label set  $\underline{\mathcal{Y}}$ . Aim is to provide a conditional probability  $p(y | x)$  for any  $x \in \underline{\mathcal{X}}$  and  $y \in \underline{\mathcal{Y}}$ .
- 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  $k = 1 \dots m$   
⇒ A **feature vector**  $f(x, y) \in \mathbb{R}^m$  for any  $x \in \mathcal{X}$  and  $y \in \mathcal{Y}$ .

$\langle 1, 0, 1, 0, 0, 0, 1, 1 \rangle$

# Maximum Entropy Markov Model (MEMM)

Feature Vectors given limited History + token(label)

- $\mathcal{X}$  is the set of all possible histories of form  $\langle t_{-2}, t_{-1}, w_{[1:n]}, i \rangle$
- $\mathcal{Y} = \{\text{NN, NNS, Vt, Vi, IN, DT, ...}\}$
- We have  $m$  features  $f_k : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$  for  $k = 1 \dots m$

For example:

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

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

$$f_1(\langle \text{JJ, DT, ( Hispaniola, ...), 6 }, \text{Vt} \rangle) = 1$$
$$f_2(\langle \text{JJ, DT, ( Hispaniola, ...), 6 }, \text{Vt} \rangle) = 0$$

e(base | Vt)

DIFFICULT  
FOR  
HMMs

# Maximum Entropy Markov Model (MEMM)

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

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

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

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

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

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

# Maximum Entropy Markov Model (MEMM)

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

- Contextual Features, e.g.,

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

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

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

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

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

# Maximum Entropy Markov Model (MEMM)

$v^*f(x, y)$      $v$  = parameter vector     $y'$  all possible labels

$e^{v^*f(x, y)}$  = prob given  $y$  label     $\sum_{y' \in Y} e^{v^*f(x, y')}$  = Normalization

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

$$m = 3$$

$$f(x, y) = \langle 1, 0, 1 \rangle$$

$$v = \langle 1, 2, 3 \rangle$$

- 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  $k = 1 \dots m$   
⇒ A feature vector  $\underline{f}(x, y) \in \mathbb{R}^m$  for any  $x \in \mathcal{X}$  and  $y \in \mathcal{Y}$ .

- We also have a **parameter vector**  $\underline{v} \in \mathbb{R}^m$

- We define

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

$$\begin{aligned} v \cdot f(x, y) &= \\ 1 \times 1 + 0 \times 2 + 1 \times 3 &= 4 \end{aligned}$$

# Maximum Entropy Markov Model (MEMM)

## Training the Log-Linear Model

<DT, JS, the red dog, 3>

NN

- To train a log-linear model, we need a training set  $(\underline{x}_i, \underline{y}_i)$  for  $i = 1 \dots n$ . Then search for

$$\underline{v}^* = \operatorname{argmax}_v \left( \underbrace{\sum_i \log p(y_i | x_i; v)}_{\text{Log-Likelihood}} - \frac{\lambda}{2} \underbrace{\sum_k v_k^2}_{\text{Regularizer}} \right) > 0$$

(see last lecture on log-linear models)

- Training set is simply all history/tag pairs seen in the training data

# Maximum Entropy Markov Model (MEMM)

## 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**

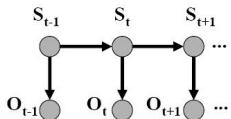
# Conditional Random Fields

## Comparison of HMM, MEMM and CRF (JCarafe)

### Summary

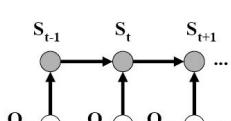
$$\vec{s} = s_1, s_2, \dots, s_n \quad \vec{o} = o_1, o_2, \dots, o_n$$

**HMM**  $P(\vec{s}, \vec{o}) \propto \prod_{t=1}^{|\vec{o}|} P(s_t | s_{t-1}) P(o_t | s_t)$



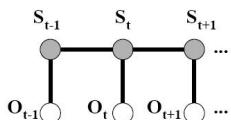
**MEMM**  $P(\vec{s} | \vec{o}) \propto \prod_{t=1}^{|\vec{o}|} P(s_t | s_{t-1}, o_t)$

$$\propto \prod_{t=1}^{|\vec{o}|} \frac{1}{Z_{s_{t-1}, o_t}} \exp \left( \sum_j \lambda_j f_j(s_t, s_{t-1}) + \sum_k \mu_k g_k(s_t, x_t) \right)$$

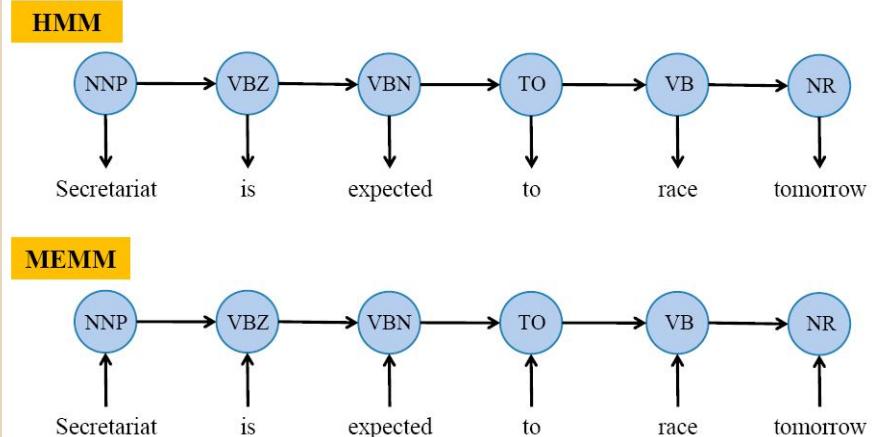


**CRF**

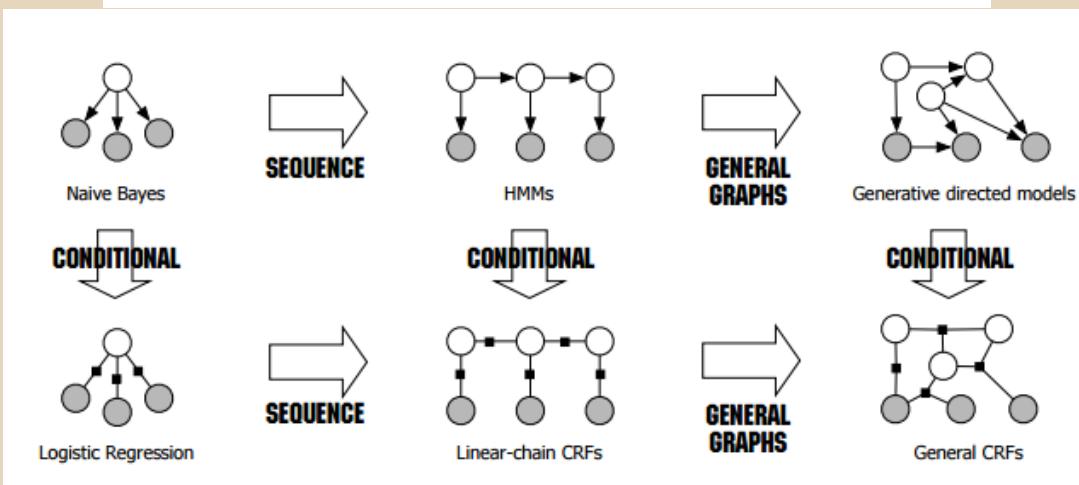
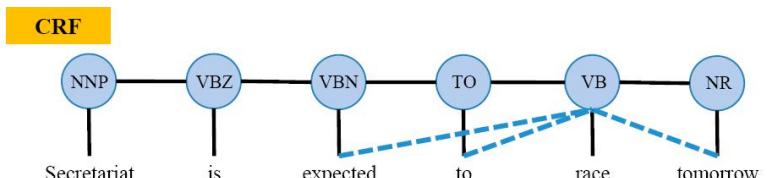
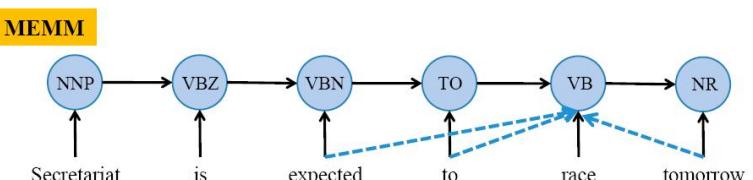
$$P(\vec{s} | \vec{o}) \propto \frac{1}{Z_{\vec{o}}} \prod_{t=1}^{|\vec{o}|} \exp \left( \sum_j \lambda_j f_j(s_t, s_{t-1}) + \sum_k \mu_k g_k(s_t, x_t) \right)$$



### HMM v.s. MEMM



### MEMM v.s. CRF



# HMM -> MEMM -> CRF

## HMM

Simpler but limited features

Joint Probability words with tags

Select Maximize probability

## MEMM

Supports more complex vector of features

Conditional Probability not Joint

Still prone to bias

## CRF (more on this next time)

Able to better capture context

Less Bias

Best Performance

# Coreference

Connecting references such as:

“**I** voted for **Nadar** because **he** was most aligned with **my** values,” **she** said

Wiki: <http://en.wikipedia.org/wiki/Coreference>

Stanford:

<http://nlp.stanford.edu/projects/coref.shtml>

<http://nlp.stanford.edu/software/dcoref.shtml>

Implemented using Multi-pass Siev

Clinical Note Usage:

<http://www.ncbi.nlm.nih.gov>

# Probabilistic Context Free Grammars

Hopcroft and Ullman, 1979

A context free grammar  $G = (N, \Sigma, R, S)$  where:

- ▶  $N$  is a set of non-terminal symbols
- ▶  $\Sigma$  is a set of terminal symbols
- ▶  $R$  is a set of rules of the form  $X \rightarrow Y_1 Y_2 \dots Y_n$  for  $n \geq 0$ ,  $X \in N$ ,  $Y_i \in (N \cup \Sigma)$
- ▶  $S \in N$  is a distinguished start symbol

## Left-Most Derivations

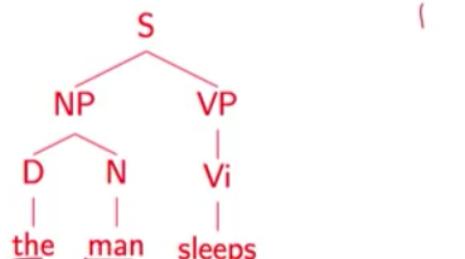
$$\Sigma^* \quad \Sigma = \{\text{the, dog, a, }\}$$

A left-most derivation is a sequence of strings  $s_1 \dots s_n$ , where  $\Sigma^*$

- ▶  $s_1 = S$ , the start symbol
- ▶  $s_n \in \Sigma^*$ , i.e.  $s_n$  is made up of terminal symbols only
- ▶ Each  $s_i$  for  $i = 2 \dots n$  is derived from  $s_{i-1}$  by picking the left-most non-terminal  $X$  in  $s_{i-1}$  and replacing it by some  $\beta$  where  $X \rightarrow \beta$  is a rule in  $R$

For example: [S], [NP VP], [D N VP], [the N VP], [the man VP], [the man Vi], [the man sleeps]

Representation of a derivation as a tree:



- $N = \{S, NP, VP, PP, DT, Vi, Vt, NN, IN\}$
- $S = S$
- $\Sigma = \{\text{sleeps, saw, man, woman, telescope, the, with, in}\}$

\* $R =$

<u>S</u>	$\rightarrow$	<u>NP</u>	<u>VP</u>
<u>VP</u>	$\rightarrow$	<u>Vi</u>	
<u>VP</u>	$\rightarrow$	<u>Vt</u>	<u>NP</u>
<u>VP</u>	$\rightarrow$	<u>VP</u>	<u>PP</u>
<u>NP</u>	$\rightarrow$	<u>DT</u>	<u>NN</u>
<u>NP</u>	$\rightarrow$	<u>NP</u>	<u>PP</u>
<u>PP</u>	$\rightarrow$	<u>IN</u>	<u>NP</u>

<u>Vi</u>	$\rightarrow$	sleeps
<u>Vt</u>	$\rightarrow$	saw
<u>NN</u>	$\rightarrow$	man
<u>NN</u>	$\rightarrow$	woman
<u>NN</u>	$\rightarrow$	telescope
<u>DT</u>	$\rightarrow$	the
<u>IN</u>	$\rightarrow$	with
<u>IN</u>	$\rightarrow$	in

Note: S=sentence, VP=verb phrase, NP=noun phrase, PP=prepositional phrase, DT=determiner, Vi=intransitive verb, Vt=transitive verb, NN=noun, IN=preposition

E  
a  
dog  
the  
a dog  
the dog  
!

# UIMAFit Example

```
JCas jCas = JCasFactory.createJCas();

AnalysisEngine analysisEngine = AnalysisEngineFactory.createEngine(
    GetStartedQuickAE.class,
    GetStartedQuickAE.PARAM_STRING, "uimaFIT");

analysisEngine.process(jCas);
```

<http://uima.apache.org/d/uimafit-current/tools.uimafit.book.html>

# Gate Example

*SORRY - NO TIME TONIGHT - WILL NEXT PRES TBD*

Getting Started

<https://gate.ac.uk/2mins.html>

Tutorials:

<https://gate.ac.uk/demos/movies.html>

# Python NLTK

Natural Language Processing with Python online book:

<http://www.nltk.org/book/>

<http://www.nltk.org/>

How to's:

<http://www.nltk.org/howto/>

# Sentiment Analysis Example

Determine if a statement is Positive or Negative

[http://en.wikipedia.org/wiki/Sentiment\\_analysis](http://en.wikipedia.org/wiki/Sentiment_analysis)

[WordNet](#) - Large Lexical English Database

SentiWordNet - Sentiment Lexical Database

<http://nlp.stanford.edu/sentiment/>

[http://nlp.stanford.edu:  
8080/sentiment/rntnDemo.html](http://nlp.stanford.edu:8080/sentiment/rntnDemo.html)

# Data Extraction

## Annotation Based

Use POS + NER taggers to identify domain and related tokens and extract those that match

## Rule Based

In addition to Token tagging, build rules around complex patterns for extraction including values versus entities

## Regular Expression

Match RegEx with Pattern Groups

## Proximity Search

Find phrases with words within specified distance

# Language Conversion

Annotation Token conversion of individual words  
POS + NER, etc. Replace with translated word(s)

Grammar tree transformation for realignment of words

e.g., Spanish: Adjective follows the noun.  
red apple -> manzana roja

# MOOCs

## Coursera:

<https://www.coursera.org/course/nlpintro>

<https://www.coursera.org/course/nlangp>

<https://www.coursera.org/course/nlp>

<https://www.coursera.org/course/textretrieval>

<https://www.coursera.org/course/textanalytics>

<https://open.hpi.de/courses/semanticweb>

<https://open.hpi.de/courses/semanticweb2014>

<http://mt-class.org/>

multiple university course outlines some w/  
lecture slides, etc.

# Frameworks / Libraries

Java

<http://uima.apache.org/>

<https://gate.ac.uk/>

<http://ctakes.apache.org/>

<http://opennlp.apache.org>

<https://code.google.com/p/cleartk/>

<http://nlp.stanford.edu/>

<http://alias-i.com/lingpipe/>

Python

<http://nltk.org>

# Reference Materials

[http://en.wikipedia.org/wiki/Hidden\\_Markov\\_model](http://en.wikipedia.org/wiki/Hidden_Markov_model)

[http://en.wikipedia.org/wiki/Maximum-entropy\\_Markov\\_model](http://en.wikipedia.org/wiki/Maximum-entropy_Markov_model)

[http://en.wikipedia.org/wiki/Conditional\\_random\\_field](http://en.wikipedia.org/wiki/Conditional_random_field)

<http://people.cs.umass.edu/~mccallum/papers/crf-tutorial.pdf>

<http://nlp.stanford.edu/IR-book/html/htmledition/support-vector-machines-and-machine-learning-on-documents-1.html>

<http://nlp.stanford.edu/projects/coref.shtml>

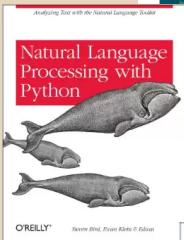
[http://en.wikipedia.org/wiki/Stochastic\\_context-free\\_grammar](http://en.wikipedia.org/wiki/Stochastic_context-free_grammar)

[http://en.wikipedia.org/wiki/Machine\\_learning](http://en.wikipedia.org/wiki/Machine_learning)

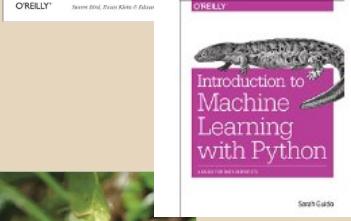
# Books



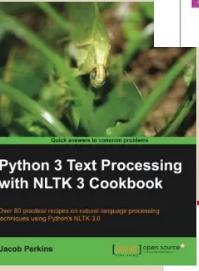
<http://www.amazon.com/Taming-Text-Find-Organize-Manipulate/dp/193398838X>



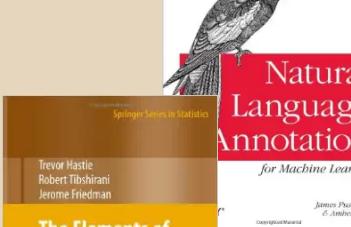
<http://www.amazon.com/Natural-Language-Processing-Python-Steven/dp/0596516495>



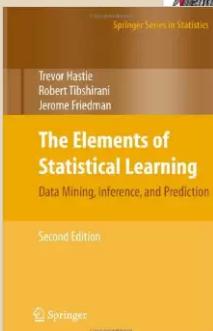
<http://www.amazon.com/Introduction-Machine-Learning-Python-Sarah/dp/1449369413>



<http://www.amazon.com/Python-Text-Processing-NLTK-Cookbook/dp/1782167854>



<http://www.amazon.com/Natural-Language-Annotation-Machine-Learning/dp/1449306667>



<http://www.amazon.com/The-Elements-Statistical-Learning-Prediction/dp/0387848576>

# Thank you for time

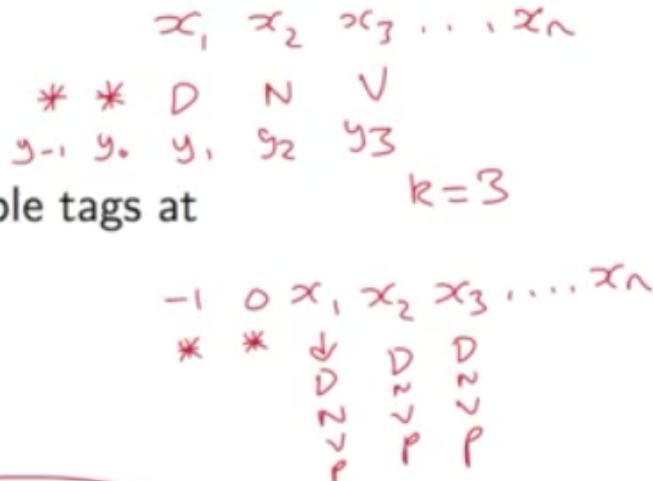
Hopefully this was helpful for at least some of you.

# Extra Slides

## 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  $k$ :

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



- ▶ Define

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

$$S = \{D, N, V, P\}$$

- ▶ Define a dynamic programming table

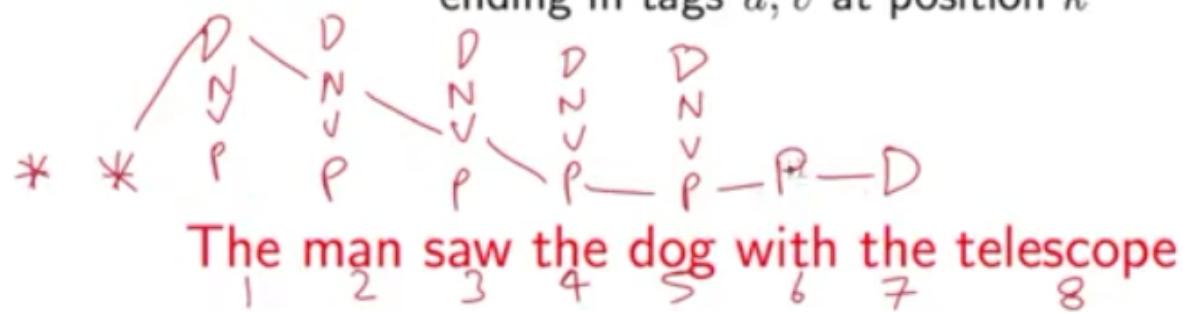
$\pi(k, u, v)$  = maximum probability of a tag sequence  
ending in tags  $u, v$  at position  $k$   
that is,

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

## An Example

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

$$S = \{D, N, V, P\}$$



$$q(1)$$
$$e(1)$$

$$\pi(7, e, D)$$

# A Recursive Definition

Base case:

$$\pi(\underline{0}, \underline{*}, \underline{*}) = \underline{1}$$

\* \*

## Recursive definition:

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

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

$\{1 \dots n\}$   
 $\in \mathcal{S}_{k-1} \in \mathcal{S}_k$

## Justification for the Recursive Definition

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

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

$$\begin{aligned} \mathcal{S}_5 \\ = \mathcal{S} = \{D, N, V, P\} \end{aligned}$$

$$\begin{aligned} & \pi(b, N, P) \\ & + q(D|N, P) \\ & + e(\text{the}|D) \end{aligned}$$



The man saw the dog with the telescope

$$\pi(7, P, D) = \max_{w \in \{D, N, V, P\}} (\pi(6, w, P) \times q(D|w, P) \times e(\text{the}|D))$$

# The Viterbi Algorithm

INPUT =  $x_1 \ x_2 \ \dots \ x_n$

OUTPUT = ~~max~~  $\max_{y_1 \dots y_{n+1}} p(x_1 \dots x_n, y_1 \dots y_{n+1})$

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

**Initialization:** Set  $\pi(0, *, *) = 1$   $\leftarrow \{D, N, V, P\}$

**Definition:**  $S_{-1} = S_0 = \{*\}$ ,  $S_k = S$  for  $k \in \{1 \dots n\}$

**Algorithm:**

1.2, 3,  $\sim$

► For  $k = 1 \dots n$ ,

► For  $u \in S_{k-1}$ ,  $v \in S_k$ ,

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

► **Return**  $\max_{u \in S_{n-1}, v \in S_n} (\pi(n, u, v) \times q(\text{STOP}|u, v))$   $\ddagger$

## 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  $u \in \mathcal{S}_{k-1}$ ,  $v \in \mathcal{S}_k$ ,
$$\begin{aligned}\pi(k, u, v) &= \max_{w \in \mathcal{S}_{k-2}} (\pi(k-1, w, u) \times q(v|w, u) \times e(x_k|v)) \\ bp(k, u, v) &= \arg \max_{w \in \mathcal{S}_{k-2}} (\pi(k-1, w, u) \times q(v|w, u) \times e(x_k|v))\end{aligned}$$
  - ▶ 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  $y_1 \dots y_n$

INPUT =  $x_1 \dots x_n$

OUTPUT =

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

## The Viterbi Algorithm with Backpointers

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

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

$O(n|\mathcal{S}|^3)$

**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  $y_1 \dots y_n$

## 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

e(firstword|NA)

e(initCap|SC)\*