DataScience Indy

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)

Automatic summarization

Coreference resolution

Discourse analysis

Machine translation

Morphological segmentation

Named entity recognition (NER)

Natural language generation

Natural language understanding

Optical character recognition (OCR)

Part-of-speech tagging

Parsing

Question answering

Relationship extraction

Sentence breaking (also known as sentence boundary disambiguation)

Sentiment analysis

Speech recognition

Speech segmentation

Topic segmentation and recognition

Word segmentation

Word sense disambiguation

Information extraction (IE)

Speech processing

Stemming

Text simplification

Text-to-speech

Text-proofing

Natural language search

Query expansion

Automated essay scoring

Truecasing

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

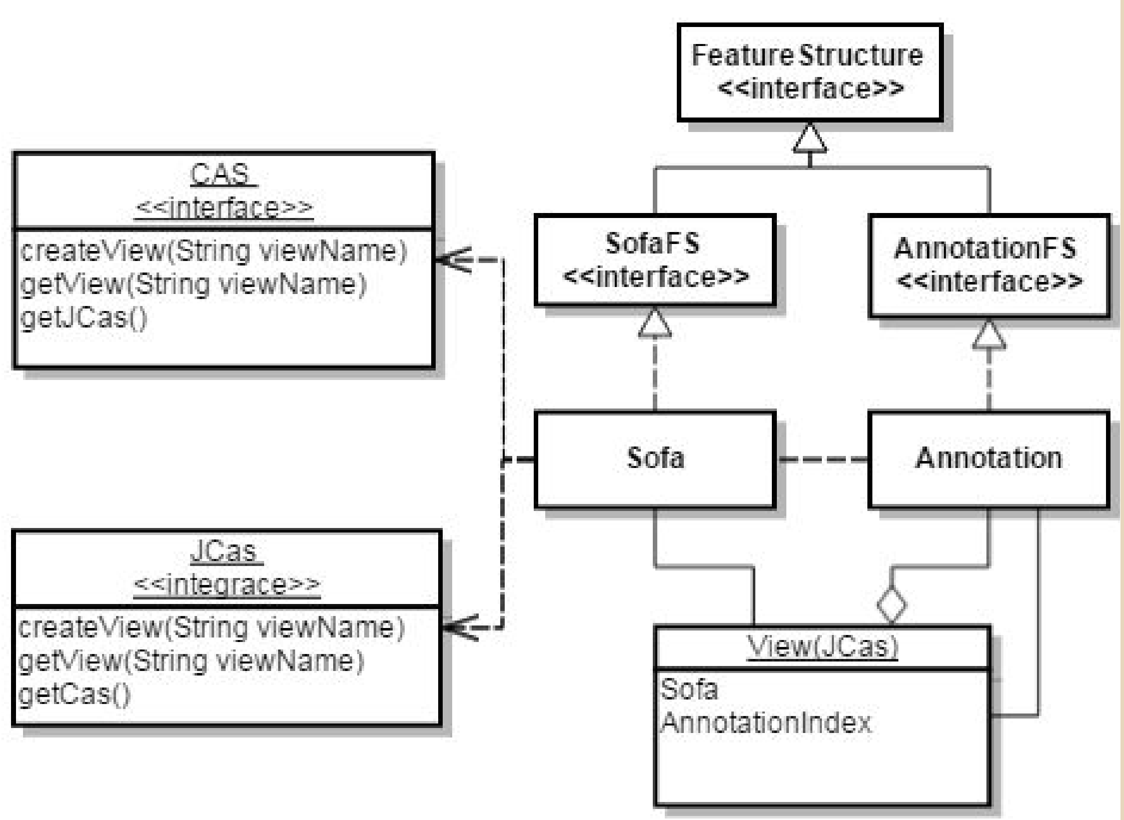
Common NLP Objects

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)

- Predict current state on only most recent history

Maximum Entropy Markov Model (MEMM)

- Predict Labels based on Conditional Probability

Perceptron

- Linear Model, non-binary Output, Linear Classifier

Support Vector Machines (SVM)

- Vector space based large-margin classifier

Conditional Random Fields

- Uni-directed Graph
- Predicts Sequences of Labels for Input Samples

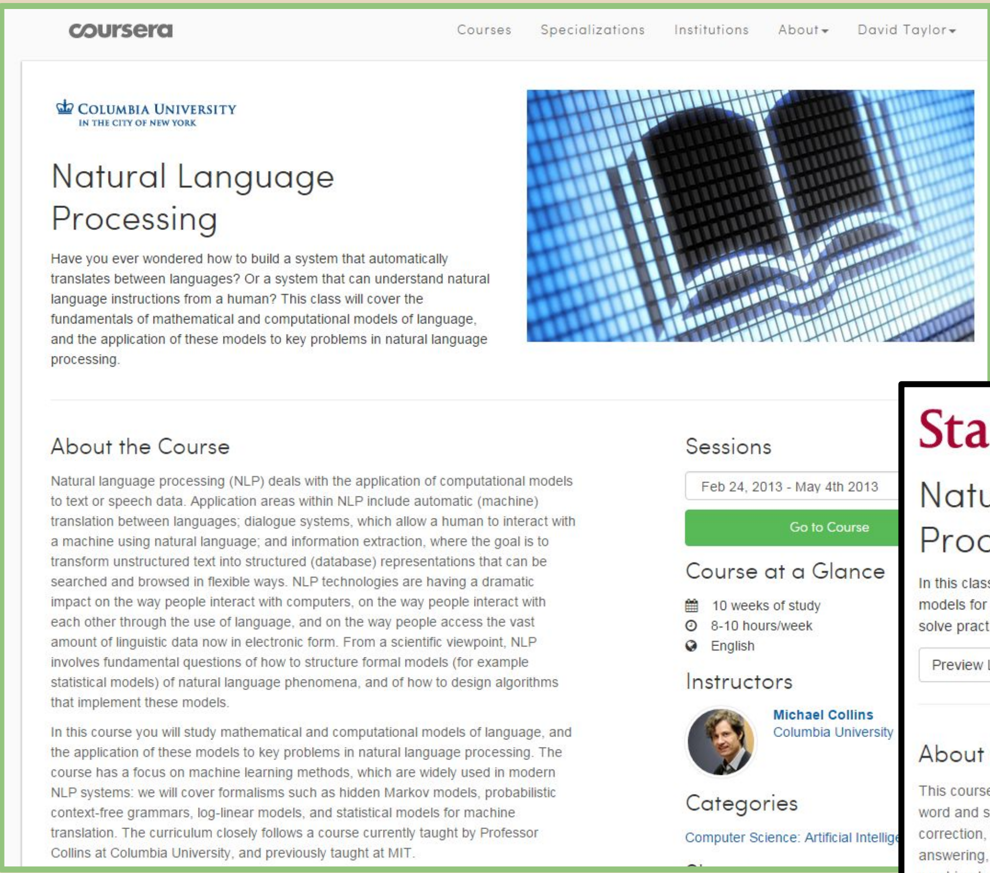
Coreference

- Occurrence of terms together

Probabilistic Context Free Grammar (PCFG)

- Directed Graph from One State to other possible states
- 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

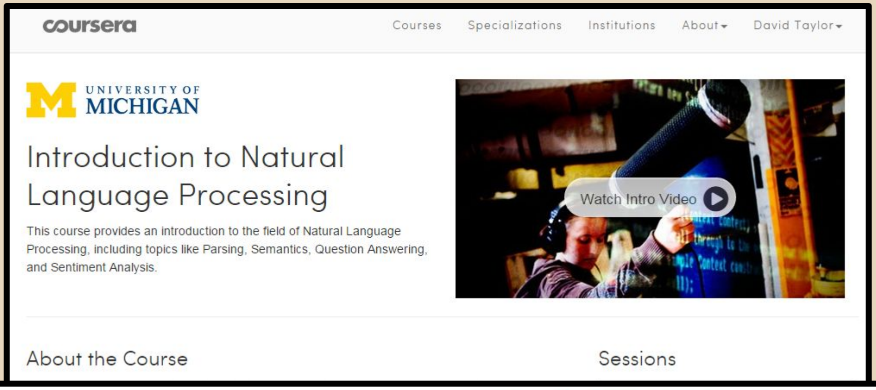
- 10 weeks of study
- 8-10 hours/week
- English

Instructors

Michael Collins
Columbia University

Categories

Computer Science: Artificial Intellige



coursera Courses Specializations Institutions About David Taylor

UNIVERSITY OF MICHIGAN

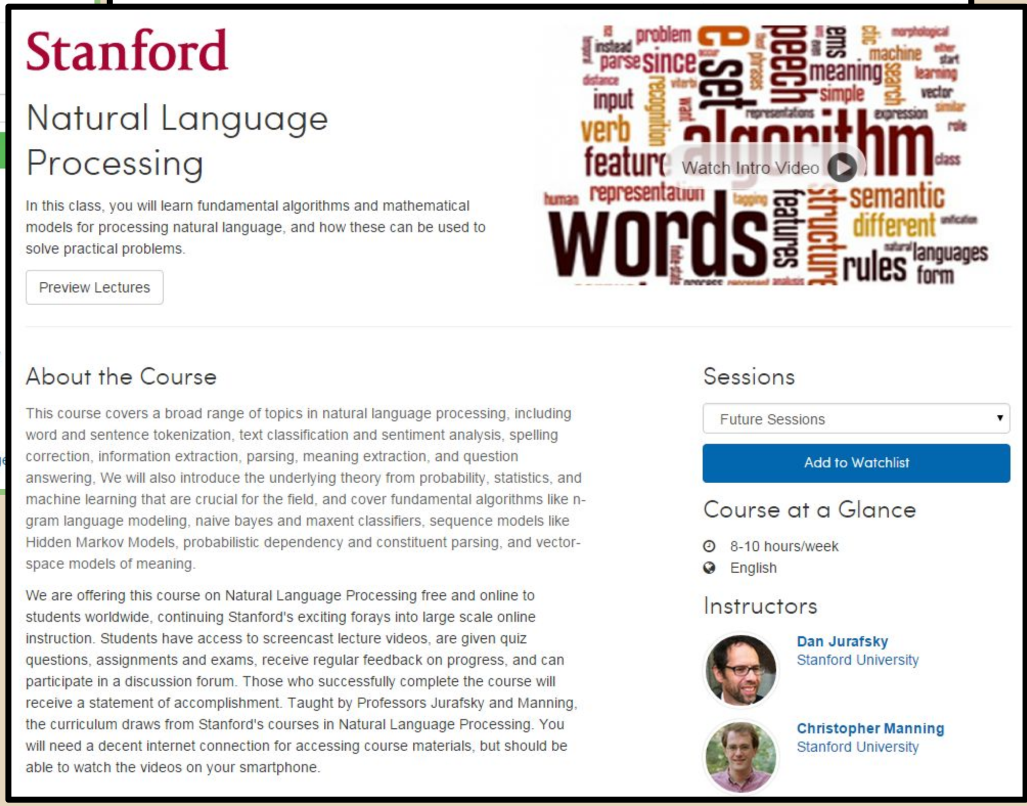
Introduction to Natural Language Processing

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

Watch Intro Video

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

Watch Intro Video

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.

Sessions

Future Sessions

Add to Watchlist

Course at a Glance

- 8-10 hours/week
- English

Instructors

Dan Jurafsky
Stanford University

Christopher Manning
Stanford University

Hidden Markov Model (Trigram HMM)

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

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

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

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

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

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

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

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

- 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

$$p(x_1 \dots x_n, y_1 \dots y_n) = \overbrace{\underbrace{q(\text{STOP}|y_{n-1}, y_n) \prod_{j=1}^{n} q(y_j \mid y_{j-2}, y_{j-1})}_{\text{Markov Chain}}}^{p(y_1, y_2 \dots y_{n+1})}$$

$$\times \boxed{\underbrace{\prod_{j=1}^{n} e(x_j \mid y_j)}_{x_j\text{'s are observed}}}$$

$p(x_1 \dots x_n | y_1 \dots y_{n+1})$

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

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

Hidden Markov Models (HMM) cont.

d Estimation

TRIGRAM ML ESTIMATE

$$q(\text{Vt} \mid \text{DT}, \text{JJ}) = \lambda_1 \times \boxed{\frac{\text{Count}(\text{Dt}, \text{JJ}, \text{Vt})}{\text{Count}(\text{Dt}, \text{JJ})}}$$
$$+\lambda_2 \times \boxed{\frac{\text{Count}(\text{JJ}, \text{Vt})}{\text{Count}(\text{JJ})}}$$
$$+\lambda_3 \times \boxed{\frac{\text{Count}(\text{Vt})}{\text{Count}()}}$$

BIGRAM ML ESTIMATE

UNIGRAM ML ESTIMATE

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

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

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

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

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

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

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

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

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 $p(y \mid x)$ 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 $y \in \mathcal{Y}$.
- We also have a **parameter vector** $v \in \mathbb{R}^m$
- We define

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

$m = 3$

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

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

$v \cdot f(x,y) =$

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

$= 4$

Maximum Entropy Markov Model (MEMM)

Recap: Feature Vector Representations in Log-Linear Models

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

$\{D, N, V\}$

- 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** $f : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}$).

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

- 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 $y \in \mathcal{Y}$.

Maximum Entropy Markov Model (MEMM)

Feature Vectors given limited History + token(label)

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

For example:

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

$e(\text{base}|\text{Vt})$

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

DIFFICULT FOR HMMS

$\ldots$

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

Maximum Entropy Markov Model (MEMM)

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

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

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

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

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

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

Maximum Entropy Markov Model (MEMM)

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

- Contextual Features, e.g.,

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

⟸ BIGRAM

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

⟸ UNIGRAM

$\square$

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

— W_i —

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

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

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 $p(y \mid x)$ 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 $\underline{f(x,y)} \in \mathbb{R}^m$ 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

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

$m = 3$

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

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

$v \cdot f(x,y) =$

$1\times1 + 0\times2 + 1\times3$

$= 4$

Maximum Entropy Markov Model (MEMM)

Training the Log-Linear Model <DT, JJ, the red dog, 3>

NN

- To train a log-linear model, we need a training set (x_i, y_i) for $i = 1 \ldots n$. Then search for

> 0

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

(see last lecture on log-linear models)

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

Maximum Entropy Markov Model (MEMM)

Summary

- Key ideas in log-linear taggers:
  - Decompose

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

  - Estimate

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

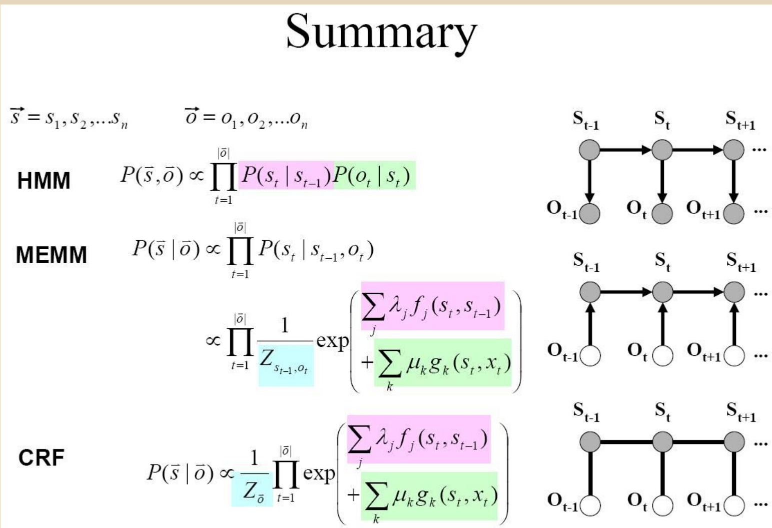
    using a log-linear model
  - For a test sentence $w_1 \dots w_n$, use the Viterbi algorithm to find

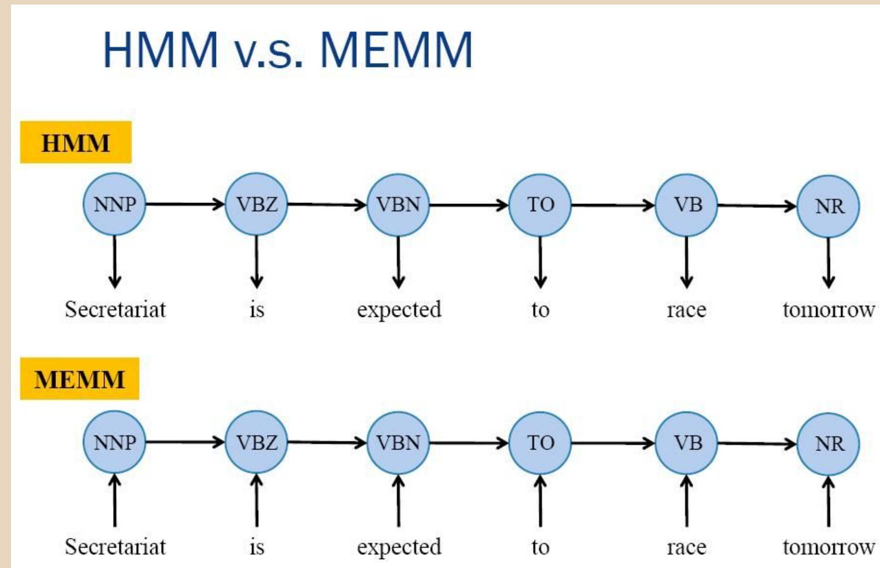
$$\arg\max_{t_1 \dots t_n} \left(\prod_{i=1}^{n} p(t_i | t_{i-2}, t_{i-1}, w_1 \dots w_n) \right)$$

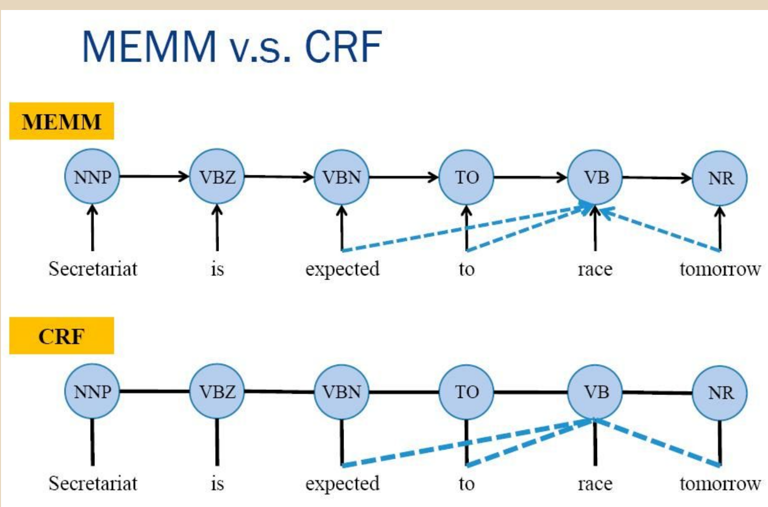
- Key advantage over HMM taggers: flexibility in the features they can use

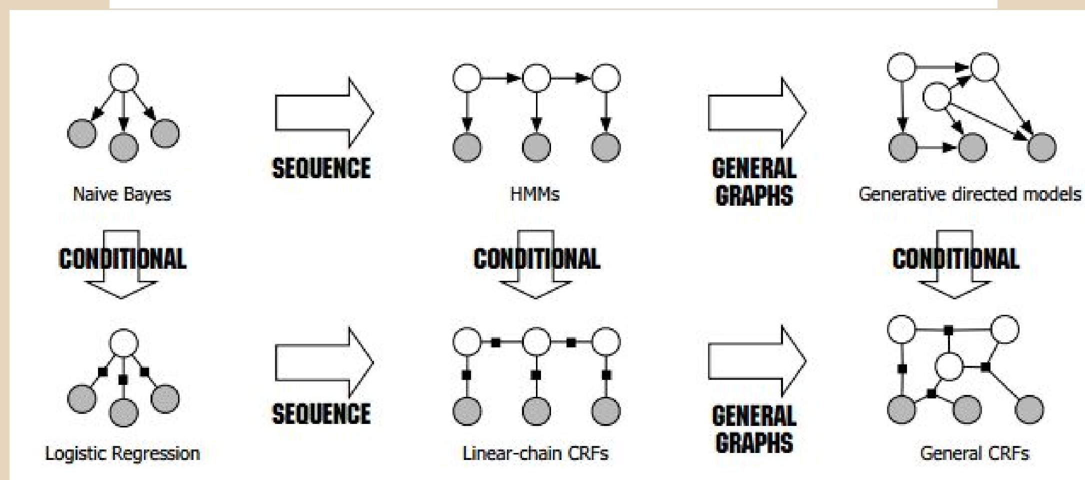
Conditional Random Fields

Comparison of HMM, MEMM and CRF (JCarafe)









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.

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

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

- $R =$

S	$\rightarrow$	NP	VP
VP	$\rightarrow$	Vi	
VP	$\rightarrow$	Vt	NP
VP	$\rightarrow$	VP	PP
NP	$\rightarrow$	DT	NN
NP	$\rightarrow$	NP	PP
PP	$\rightarrow$	IN	NP

Vi	$\rightarrow$	sleeps
Vt	$\rightarrow$	saw
NN	$\rightarrow$	man
NN	$\rightarrow$	woman
NN	$\rightarrow$	telescope
DT	$\rightarrow$	the
IN	$\rightarrow$	with
IN	$\rightarrow$	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

Left-Most Derivations

Σ^* $\Sigma = \{$the, dog, a$\}$

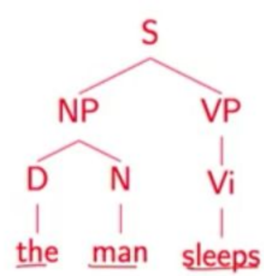
A left-most derivation is a sequence of strings $s_1 \ldots s_n$, where Σ^*

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

$\boxed{\in}$
a
dog
the
a dog
the dog

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:



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

WordNet - Large Lexical English Database

SentiWordNet - Sentiment Lexical Database

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

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/Maximum-entropy_Markov_model

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

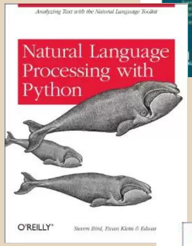
http://nlp.stanford.edu/projects/coref.shtml

http://en.wikipedia.org/wiki/Stochastic_context-free_grammar

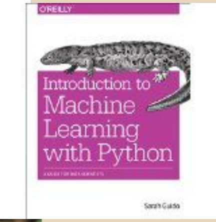
http://en.wikipedia.org/wiki/Machine_learning

Books

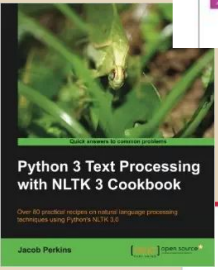
http://www.amazon.com/Taming-Text-Find-Organize-Manipulate/dp/193398838X



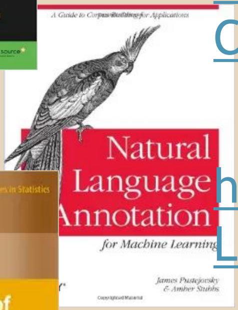
http://www.amazon.com/Natural-Language-Processing-Python-Steven/dp/0596516495



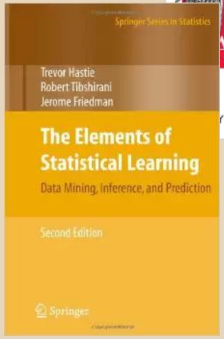
http://www.amazon.com/Introduction-Machine-Learning-Python-Sarah/dp/1449369413



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

$x_1 \quad x_2 \quad x_3 \; \ldots \; x_n$

$* \quad * \quad D \quad N \quad V$

$y_{-1} \quad y_0 \quad y_1 \quad y_2 \quad y_3$

$k=3$

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

$$\boxed{\begin{array}{c} S_{-1} = S_0 = \{*\} \\ S_k = S \quad \text{for } k \in \{1 \ldots n\} \end{array}}$$

$-1 \quad 0 \quad x_1 \quad x_2 \quad x_3 \; \ldots \; x_n$

$* \quad * \quad D,N,V,P \quad D,N,V,P \quad D,N,V,P$

- Define

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

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

- Define a dynamic programming table

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

k: $\{1, 2, \ldots n\}$ $\quad u$: S_{k-1} $\quad v$: S_k

that is,

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

$* \quad *$

An Example

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

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

The man saw the dog with the telescope

1 2 3 4 5 6 7 8

$q(\ \mid\ )$

$e(\ \mid\ )$

$\pi(7, P, D)$

A Recursive Definition

$* \quad *$

Base case:

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

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

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

$\{1, \ldots, n\}$ $\quad \in S_{k-1}$ $\quad \in S_k$

Justification for the Recursive Definition

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

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

$S_5 = S = \{D, N, V, P\}$

$\boxed{\pi(6, N, P) \times q(D|N, P) \times e(\text{the}|D)}$

D
$\boxed{N}$
V
P P D

The man saw the dog with the telescope
1 2 3 4 5 6 7 8

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

The Viterbi Algorithm

INPUT $= x_1 \, x_2 \, \ldots \, x_n$

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

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

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

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

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

Algorithm:

1, 2, 3, n

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

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

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

The Viterbi Algorithm with Backpointers

INPUT = $x_1, \ldots, x_n$

OUTPUT =

arg max $p(x_1 .. x_n, y_1 .. y_{n+1})$
$y_1 .. y_{n+1}$

Input: a sentence $x_1 \ldots x_n$, 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 $k \in \{1 \ldots n\}$

Algorithm:

- For $k = 1 \ldots n$,
  - 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 $(y_{n-1}, y_n) = \arg\max_{(u,v)} (\pi(n, u, v) \times q(\mathsf{STOP}|u, v))$
- For $k = (n-2) \ldots 1$, $y_k = bp(k+2, y_{k+1}, y_{k+2})$
- **Return** the tag sequence $y_1 \ldots y_n$

The Viterbi Algorithm with Backpointers

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

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

$O(n|S|^3)$

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

Algorithm:

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

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

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

- Set $(y_{n-1}, y_n) = \arg\max_{(u,v)} \left(\pi(n,u,v) \times q(\text{STOP}|u,v)\right)$
- For $k = (n-2) \ldots 1$, $y_k = bp(k+2, y_{k+1}, y_{k+2})$
- **Return** the tag sequence $y_1 \ldots y_n$

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

$\Downarrow$

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
...	

$e(\text{firstword} \mid NA)$

$e(\text{initCap} \mid SC)$