Lecture 3:
Structures and Decoding
Outline

1. Structures in NLP
2. HMMs as BNs
   - Viterbi algorithm as variable elimination
3. Linear models
4. Five views of decoding
Two Meanings of “Structure”

• Yesterday: structure of a graph for modeling a collection of random variables together.

• Today: linguistic structure.
  – Sequence labelings (POS, IOB chunkings, …)
  – Parse trees (phrase-structure, dependency, …)
  – Alignments (word, phrase, tree, …)
  – Predicate-argument structures
  – Text-to-text (translation, paraphrase, answers, …)
A Useful Abstraction?

• We think so.
• Brings out commonalities:
  – Modeling formalisms (e.g., linear models with features)
  – Learning algorithms (lectures 4-6)
  – Generic inference algorithms
• Permits sharing across a wider space of problems.
• Disadvantage: hides engineering details.
Familiar Example:
Hidden Markov Models
Hidden Markov Model

• $X$ and $Y$ are both sequences of symbols
  – $X$ is a sequence from the vocabulary $\Sigma$
  – $Y$ is a sequence from the state space $\Lambda$

\[
p(X = x, Y = y) = \left( \prod_{i=1}^{n} p(x_i \mid y_i)p(y_i \mid y_{i-1}) \right) p(\text{stop} \mid y_n)\]

• Parameters:
  – Transitions $p(y' \mid y)$
    • including $p(\text{stop} \mid y)$, $p(y \mid \text{start})$
  – Emissions $p(x \mid y)$
Hidden Markov Model

• The joint model’s independence assumptions are easy to capture with a Bayesian network.

\[ p(X = x, Y = y) = \left( \prod_{i=1}^{n} p(x_i | y_i)p(y_i | y_{i-1}) \right) p(stop | y_n) \]
Hidden Markov Model

- The joint model instantiates **dynamic Bayesian networks**.

\[
p(X = x, Y = y) = \left( \prod_{i=1}^{n} p(x_i | y_i)p(y_i | y_{i-1}) \right) p(stop | y_n)
\]
Hidden Markov Model

- Given $X$'s value as evidence, the dynamic part becomes unnecessary, since we know $n$.

$$p(X = x, Y = y) = \left( \prod_{i=1}^{n} p(x_i | y_i)p(y_i | y_{i-1}) \right) p(stop | y_n)$$
Hidden Markov Model

• The usual inference problem is to find the most probable value of $Y$ given $X = x$. 

\[
\begin{align*}
Y_0 & \rightarrow Y_1 \rightarrow Y_2 \rightarrow Y_3 \rightarrow \ldots \rightarrow Y_n \\
X_1 &= x_1 \\
X_2 &= x_2 \\
X_3 &= x_3 \\
X_n &= x_n \\
\text{stop} \\
\end{align*}
\]
Hidden Markov Model

• The usual inference problem is to find the most probable value of $Y$ given $X = x$.

• Factor graph:
Hidden Markov Model

• The usual inference problem is to find the most probable value of $Y$ given $X = x$.

• Factor graph after reducing factors to respect evidence:
Hidden Markov Model

• The usual inference problem is to find the most probable value of $Y$ given $X = x$.

• Clever ordering should be apparent!
Hidden Markov Model

• When we eliminate $Y_1$, we take a product of three relevant factors.
  • $p(Y_1 \mid \text{start})$
  • $\eta(Y_1) = \text{reduced } p(x_1 \mid Y_1)$
  • $p(Y_2 \mid Y_1)$
Hidden Markov Model

- When we eliminate $Y_1$, we first take a product of two factors that only involve $Y_1$.

$p(Y_1 \mid \text{start})$

$\eta(Y_1) = \text{reduced } p(x_1 \mid Y_1)$
Hidden Markov Model

• When we eliminate $Y_1$, we first take a product of two factors that only involve $Y_1$.

• This is the Viterbi probability vector for $Y_1$. 

\[
\begin{align*}
\phi_1(Y_1) \\
Y_1 \\
Y_2 \\
\vdots \\
Y_{|\Lambda|} \\
\end{align*}
\]
Hidden Markov Model

- When we eliminate $Y_1$, we first take a product of two factors that only involve $Y_1$.
- This is the Viterbi probability vector for $Y_1$.
- Eliminating $Y_1$ equates to solving the Viterbi probabilities for $Y_2$. 

$y_1 \quad y_2 \quad \cdots \quad y_n$

$\phi_1(Y_1) \quad \frac{\Lambda}{\Lambda} \quad \frac{p(Y_2 \mid Y_1)}{p(Y_2 \mid Y_1)}$
Hidden Markov Model

• Product of all factors involving $Y_1$, then reduce.
  • $\phi_2(Y_2) = \max_{y \in \text{Val}(Y_1)} (\phi_1(y) \times p(Y_2 \mid y))$
  • This factor holds Viterbi probabilities for $Y_2$. 
Hidden Markov Model

• When we eliminate $Y_2$, we take a product of the analogous two relevant factors.
• Then reduce.
  • $\phi_3(Y_3) = \max_{y \in \text{Val}(Y_2)} (\phi_2(y) \times p(Y_3 | y))$
Hidden Markov Model

- At the end, we have one final factor with one row, $\phi_{n+1}$.
- This is the score of the best sequence.
- Use backtrace to recover values.
Why Think This Way?

• Easy to see how to generalize HMMs.
  – More evidence
  – More factors
  – More hidden structure
  – More dependencies

• Probabilistic interpretation of factors is not central to finding the “best” $Y$ ...
  – Many factors are not conditional probability tables.
• Each word also depends on previous state.
Generalization Example 2

• “Trigram” HMM
Generalization Example 3

- Aggregate bigram model (Saul and Pereira, 1997)
General Decoding Problem

• Two structured random variables, $X$ and $Y$.
  – Sometimes described as collections of random variables.

• “Decode” observed value $X = x$ into some value of $Y$.

• Usually, we seek to maximize some score.
  – E.g., MAP inference from yesterday.
Linear Models

• Define a feature vector function $g$ that maps $(x, y)$ pairs into $d$-dimensional real space.

• Score is linear in $g(x, y)$.

$$score(x, y) = \mathbf{w}^\top g(x, y)$$

$$y^* = \arg \max_{y \in \mathcal{Y}_x} \mathbf{w}^\top g(x, y)$$

• Results:
  – decoding seeks $y$ to maximize the score.
  – learning seeks $\mathbf{w}$ to ... do something we’ll talk about later.

• Extremely general!
Generic Noisy Channel as Linear Model

\[
\hat{y} = \arg \max_y \log (p(y) \cdot p(x | y)) \\
= \arg \max_y \log p(y) + \log p(x | y) \\
= \arg \max_y w_y + w_{x|y} \\
= \arg \max_y w^\top g(x, y)
\]

• Of course, the two probability terms are typically composed of “smaller” factors; each can be understood as an exponentiated weight.
Max Ent Models as Linear Models

\[
\hat{y} = \arg\max_y \log p(y \mid x)
\]

\[
= \arg\max_y \log \frac{\exp w^\top g(x, y)}{z(x)}
\]

\[
= \arg\max_y w^\top g(x, y) - \log z(x)
\]

\[
= \arg\max_y w^\top g(x, y)
\]
HMMs as Linear Models

\[ \hat{y} = \arg \max_y \log p(x, y) \]

\[ = \arg \max_y \left( \sum_{i=1}^n \log p(x_i \mid y_i) + \log p(y_i \mid y_{i-1}) \right) + \log p(\text{stop} \mid y_n) \]

\[ = \arg \max_y \left( \sum_{i=1}^n w_{y_i \downarrow x_i} + w_{y_{i-1} \rightarrow y_i} \right) + w_{y_n \rightarrow \text{stop}} \]

\[ = \arg \max_y \sum_{y,x} w_{y \downarrow x} \text{freq}(y \downarrow x; y, x) + \sum_{y,y^\prime} w_{y \rightarrow y^\prime} \text{freq}(y \rightarrow y^\prime; y) \]

\[ = \arg \max_y \mathbf{w}^\top \mathbf{g}(x, y) \]
Running Example

- IOB sequence labeling, here applied to NER
- Often solved with HMMs, CRFs, $M^3Ns$ ...
<table>
<thead>
<tr>
<th>feature function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$</th>
<th>$g(x, y)$</th>
<th>$g(x, y')$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bias:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>count of $i$ s.t. $y_i = B$</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>count of $i$ s.t. $y_i = I$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>count of $i$ s.t. $y_i = O$</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td><strong>lexical:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>count of $i$ s.t. $x_i = Britain$ and $y_i = B$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>count of $i$ s.t. $x_i = Britain$ and $y_i = I$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>count of $i$ s.t. $x_i = Britain$ and $y_i = O$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>downcased:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>count of $i$ s.t. $lc(x_i) = britain$ and $y_i = B$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>count of $i$ s.t. $lc(x_i) = britain$ and $y_i = I$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>count of $i$ s.t. $lc(x_i) = britain$ and $y_i = O$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>count of $i$ s.t. $lc(x_i) = sent$ and $y_i = O$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>count of $i$ s.t. $lc(x_i) = warships$ and $y_i = O$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>shape:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>count of $i$ s.t. $shape(x_i) = Aaaaaa$ and $y_i = B$</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>count of $i$ s.t. $shape(x_i) = Aaaaaa$ and $y_i = I$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>count of $i$ s.t. $shape(x_i) = Aaaaaa$ and $y_i = O$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>prefix:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>count of $i$ s.t. $pre_1(x_i) = B$ and $y_i = B$</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>count of $i$ s.t. $pre_1(x_i) = B$ and $y_i = I$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>count of $i$ s.t. $pre_1(x_i) = B$ and $y_i = O$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>count of $i$ s.t. $pre_1(x_i) = s$ and $y_i = O$</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>count of $i$ s.t. $shape(pre_1(x_i)) = A$ and $y_i = B$</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>count of $i$ s.t. $shape(pre_1(x_i)) = A$ and $y_i = I$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>count of $i$ s.t. $shape(pre_1(x_i)) = A$ and $y_i = O$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$[shape(pre_1(x_i)) = A \land y_1 = B]$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$[shape(pre_1(x_i)) = A \land y_1 = O]$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>gazetteer:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>count of $i$ s.t. $x_i$ is in the gazetteer and $y_i = B$</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>count of $i$ s.t. $x_i$ is in the gazetteer and $y_i = I$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>count of $i$ s.t. $x_i$ is in the gazetteer and $y_i = O$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>count of $i$ s.t. $x_i = sent$ and $y_i = O$</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
(What is Not A Linear Model?)

- Models with hidden variables

\[
\arg\max_y p(y \mid x) = \arg\max_y \sum_z p(y, z \mid x)
\]

- Models based on non-linear kernels

\[
\arg\max_y w^\top g(x, y) = \arg\max_y \sum_{i=1}^N \alpha_i K (\langle x_i, y_i \rangle, \langle x, y \rangle)
\]
Decoding

• For HMMs, the decoding algorithm we usually think of first is the Viterbi algorithm.
  – This is just one example.

• We will view decoding in five different ways.
  – Sequence models as a running example.
  – These views are not just for HMMs.
  – Sometimes they will lead us back to Viterbi!
Five Views of Decoding
1. Probabilistic Graphical Models

- View the linguistic structure as a collection of random variables that are interdependent.
- Represent interdependencies as a directed or undirected graphical model.
- Conditional probability tables (BNs) or factors (MNs) encode the probability distribution.
Inference in Graphical Models

• General algorithm for exact MAP inference: variable elimination.
  – Iteratively solve for the best values of each variable conditioned on values of “preceding” neighbors.
  – Then trace back.

The Viterbi algorithm is an instance of max-product variable elimination!
MAP is Linear Decoding

- Bayesian network:

\[ \sum_i \log p(x_i | \text{parents}(X_i)) \]

\[ + \sum_j \log p(y_j | \text{parents}(Y_j)) \]

- Markov network:

\[ \sum_C \log \phi_C (\{x_i\}_{i \in C}, \{y_j\}_{j \in C}) \]

- This only works if every variable is in $X$ or $Y$. 
Inference in Graphical Models

• Remember: more edges make inference more expensive.
  – Fewer edges means stronger independence.

• Really pleasant:
Inference in Graphical Models

• Remember: more edges make inference more expensive.
  – Fewer edges means stronger independence.

• Really unpleasant:
2. Polytopes
“Parts”

• Assume that feature function $g$ breaks down into local parts.

$$g(x, y) = \sum_{i=1}^{\#parts(x)} f(\Pi_i(x, y))$$

• Each part has an alphabet of possible values.
  – Decoding is choosing values for all parts, with consistency constraints.
  – (In the graphical models view, a part is a clique.)
Example

- One part per word, each is in \{B, I, O\}
- No features look at multiple parts
  - Fast inference
  - Not very expressive
Example

- One part per bigram, each is in \{BB, BI, BO, IB, II, IO, OB, OO\}
- Features and constraints can look at pairs
  - Slower inference
  - A bit more expressive
Let $z_{i,\pi}$ be 1 if part $i$ takes value $\pi$ and 0 otherwise.

- $z$ is a vector in $\{0, 1\}^N$
  - $N = $ total number of localized part values
  - Each $z$ is a vertex of the unit cube
Score is Linear in $z$

$$\arg\max_y w^\top g(x, y) = \arg\max_y w^\top \sum_{i=1}^{\#\text{parts}(x)} f(\Pi_i(x, y))$$

$$= \arg\max_y w^\top \sum_{i=1}^{\#\text{parts}(x)} \sum_{\pi \in \text{Values}(\Pi_i)} f(\pi) 1\{\Pi_i(x, y) = \pi\}$$

$$= \arg\max_{z \in Z_x} w^\top \sum_{i=1}^{\#\text{parts}(x)} \sum_{\pi \in \text{Values}(\Pi_i)} f(\pi) z_{i, \pi}$$

$$= \arg\max_{z \in Z_x} w^\top F_x z$$

$$= \arg\max_{z \in Z_x} (w^\top F_x) z$$
Polyhedra

• Not all vertices of the $N$-dimensional unit cube satisfy the constraints.
  – E.g., can’t have $z_{1,Bl} = 1$ and $z_{2,Bl} = 1$

• Sometimes we can write down a small (polynomial number) of linear constraints on $z$.

• Result: linear objective, linear constraints, integer constraints ...
Integer Linear Programming

• Very easy to add new constraints and non-local features.

• Many decoding problems have been mapped to ILP (sequence labeling, parsing, ...), but it’s not always trivial.

• NP-hard in general.
  – But there are packages that often work well in practice (e.g., CPLEX)
  – Specialized algorithms in some cases
  – LP relaxation for approximate solutions
Remark

• Graphical models assumed a probabilistic interpretation
  – Though they are not always learned using a probabilistic interpretation!

• The polytope view is agnostic about how you interpret the weights.
  – It only says that the decoding problem is an ILP.
3. Weighted Parsing
Grammars

• Grammars are often associated with natural language parsing, but they are extremely powerful for imposing constraints.

• We can add weights to them.
  – HMMs are a kind of weighted regular grammar (closely connected to WFSAs)
  – PCFGs are a kind of weighted CFG
  – Many, many more.

• Weighted parsing: find the maximum-weighted derivation for a string $x$. 
Decoding as Weighted Parsing

• Every valid $y$ is a grammatical derivation (parse) for $x$.
  – HMM: sequence of “grammatical” states is one allowed by the transition table.

• Augment parsing algorithms with weights and find the best parse.

The Viterbi algorithm is an instance of recognition by a weighted grammar!
BIO Tagging as a CFG

\[
\begin{align*}
N \rightarrow & \quad B \ R_B \\
N \rightarrow & \quad O \ R_O \\
R_B \rightarrow & \quad B \ R_B \\
R_B \rightarrow & \quad O \ R_O \\
R_B \rightarrow & \quad I \ R_I \\
R_B \rightarrow & \quad \epsilon \\
R_I \rightarrow & \quad B \ R_B \\
R_I \rightarrow & \quad O \ R_O \\
R_I \rightarrow & \quad I \ R_I \\
R_I \rightarrow & \quad \epsilon \\
R_O \rightarrow & \quad B \ R_B \\
R_O \rightarrow & \quad O \ R_O \\
R_O \rightarrow & \quad \epsilon \\
\forall x \in \Sigma, \quad B \rightarrow & \quad x \\
I \rightarrow & \quad x \\
O \rightarrow & \quad x
\end{align*}
\]

- Weighted (or probabilistic) CKY is a dynamic programming algorithm very similar in structure to classical CKY.
4. Paths and Hyperpaths
Best Path

• General idea: take $x$ and build a graph.
• Score of a path factors into the edges.

$$\arg \max_y w^T g(x, y) = \arg \max_y w^T \sum_{e \in \text{Edges}} f(e) \mathbf{1}\{e \text{ is crossed by } y \text{'s path}\}$$

• Decoding is finding the best path.

The Viterbi algorithm is an instance of finding a best path!
“Lattice” View of Viterbi
Minimum Cost Hyperpath

• General idea: take $x$ and build a hypergraph.
• Score of a hyperpath factors into the hyperedges.
• Decoding is finding the best hyperpath.

• This connection was elucidated by Klein and Manning (2002).
Parsing as a Hypergraph
Parsing as a Hypergraph

cf. “Dean for democracy”
Forced to work on his thesis, sunshine streaming in the window, Mike experienced a ...
Forced to work on his thesis, sunshine streaming in the window, Mike began to ...
Why Hypergraphs?

• Useful, compact encoding of the hypothesis space.
  – Build hypothesis space using local features, maybe do some filtering.
  – Pass it off to another module for more fine-grained scoring with richer or more expensive features.
5. Weighted Logic Programming
Logic Programming

- Start with a set of axioms and a set of inference rules.

\[
\forall A, C, \quad \text{ancestor}(A, C) \iff \text{parent}(A, C)
\]

\[
\forall A, C, \quad \text{ancestor}(A, C) \iff \bigvee_{B} \text{ancestor}(A, B) \land \text{parent}(B, C)
\]

- The goal is to prove a specific theorem, goal.
- Many approaches, but we assume a deductive approach.
  - Start with axioms, iteratively produce more theorems.
\begin{align*}
\forall \ell \in \Lambda, \quad & v(\ell, 1) = \text{labeled-word}(x_1, \ell) \\
\forall \ell \in \Lambda, \quad & v(\ell, i) = \bigvee_{\ell' \in \Lambda} v(\ell', i - 1) \land \text{label-bigram}(\ell', \ell) \land \text{labeled-word}(x_i, \ell) \\
\text{goal} &= \bigvee_{\ell \in \Lambda} v(\ell, n)
\end{align*}
Weighted Logic Programming

• Twist: axioms have weights.

• Want the proof of goal with the best score:

\[
\arg \max_y \mathbf{w}^\top \mathbf{g}(\mathbf{x}, \mathbf{y}) = \arg \max_y \mathbf{w}^\top \sum_{a \in \text{Axioms}} f(a) \text{freq}(a; \mathbf{y})
\]

• Note that axioms can be used more than once in a proof (y).
Whence WLP?

• Shieber, Schabes, and Pereira (1995): many parsing algorithms can be understood in the same deductive logic framework.
• Goodman (1999): add weights, get many useful NLP algorithms.
Dynamic Programming

• Most views (exception is polytopes) can be understood as DP algorithms.
  – The low-level procedures we use are often DP.
  – Even DP is too high-level to know the best way to implement.

• DP does not imply polynomial time and space!
  – Most common approximations when the desired state space is too big: beam search, cube pruning, agendas with early stopping, ...
  – Other views suggest others.
Summary

- Decoding is the general problem of choosing a complex structure.
  - Linguistic analysis, machine translation, speech recognition, ...
  - Statistical models are usually involved (not necessarily probabilistic).
- No perfect general view, but much can be gained through a combination of views.