ECE8813
Statistical Natural Language Processing

Lecture 5: Linguistics Fundamentals

Chin-Hui Lee
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA
chl@ece.gatech.edu

Entropy of English (Shannon, 1951)

Model	Cross Entropy (bits)	Comments
Zeroth order	4.76	uniform letter log(27)
First order	4.03	unigram
Second order	2.8	bigram
Shannon's 2nd Experiment	1.34	human prediction

C. E. Shannon, "Prediction and Entropy of Printed English", *Bell System Technical Journal*, Vol. 30, pp. 50-64, 1951.

Lab1 : Probabilities of Letters

- Markov Approximation to Probability of Letters

$$P(L) = P(l_1)P(l_2 \mid l_1)\cdots P(l_{|L|} \mid l_1,\ldots,l_{|L|-1}) \quad k-gram$$

$$\approx P(l_1)P(l_2 \mid l_1)\cdots P(l_k \mid l_1,\ldots,l_{k-1})\prod_{i=k+1}^{|L|} P(l_i \mid l_{i-1}, l_{i-2},\ldots,l_k)$$

- Cross Entropy between true *p(x)* and model *q(x)*

$$H(X,q) \equiv H(X) + D(p(x) \| q(x)) = -\sum_{x \in \mathrm{X}} p(x)\log_2 q(x) = \mathrm{E}_{\mathrm{p}}[\log_2 \frac{1}{q(X)}]$$

- Perplexity – $H(X,q) \approx \log_2(\mathrm{Perp}(X))$

- Lab1: simulate Shannon's study on English letters
  - Do it for 1000 and 10000 sentences, any difference?

Linguistic Units

- Fundamental Units
  - Alphabet, letter
  - Characters (e.g. Chinese)
- Word: dictionary, lexicon
  - Stem (lexeme): morphology, inflection form (prefix/suffix)
  - Part-of-speech (PoS): eight major groups
  - Word sense disambiguation: words with multiple senses
- Phrase
- Sentence and Grammar
- Paragraph
- Articles (documents): topics and stories
- Syntax, semantics, and pragmatics
- Language-specific properties: Multilingual issues

Part of Speech and Morphology

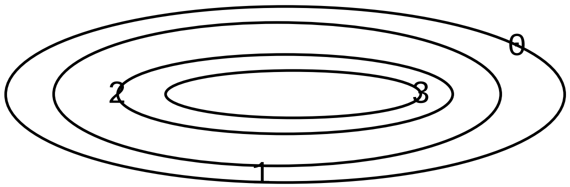
- Syntactic and Semantic Categories
  - Words that show similar syntactic behavior (semantic type)
  - Often known as PoS (noun, adjective, verb, etc.)
- Open vs. closed lexical categories
  - Class with new words added: open
  - Class with often fixed vocabulary: functional words, closed
- Part-of-speech
  - Brown Corpus (a useful corpus with PoS tags)
  - Noun, pronoun, determiner, adjective, adverbs, particles, propositions, conjuction, complementizer, and others
- Language-specific properties: Multilingual issues

Phrase Structure

- Syntax and word order
  - "I want to go to a movie tomorrow." (English vs. Chinese)
- Constituents and phrases: equivalent classes
  - Noun phrases
  - Verb phrases
  - Prepositional phrases
  - Adjective phrases
- Phrase structure grammars
  - Start symbols and derivation (rewrite) rules
  - Terminal vs. non-terminal nodes
  - Local vs. global parse trees
  - Dependency: arguments and adjuncts
- Semantics (meaning) and pragmatics
- Language-specific properties: Multilingual issues

Formal Grammar Specification

- Grammar *G={A, I, S, D}* and Language *L(G)*
  - G is defined by an alphabet set *A*, an intermediate set *I*, a root symbol *S*, and a set of derivation (production) rules *D*
  - *L(G)* is the language of the set of sentences generated by *G*
- Type of String Grammars



  - *Type 0: free* or *unrestricted*
  - *Type 1: context-sensitive*

$$D = \{\alpha\theta\beta \to \alpha\psi\beta\} \quad \theta \in I \quad \psi \in I \cup A \quad \alpha, \beta : \text{string}$$

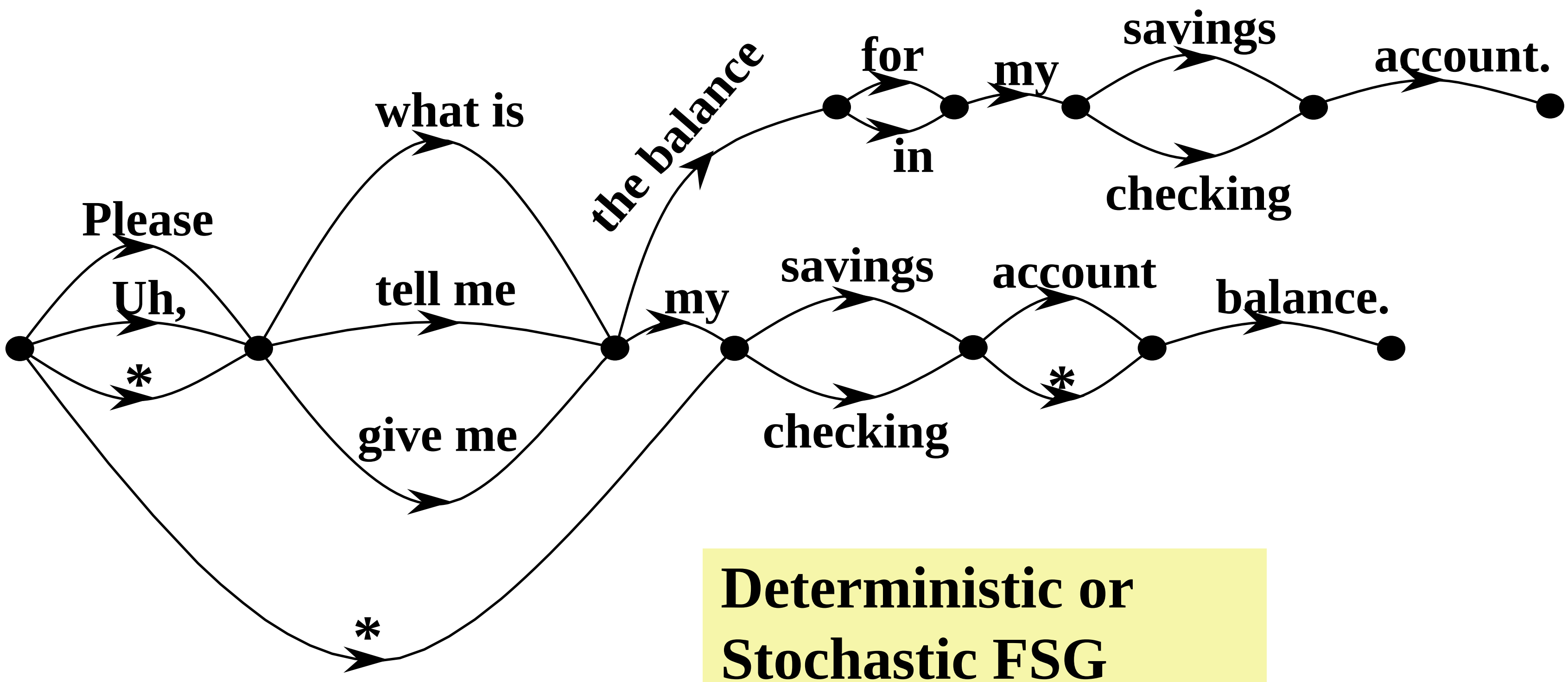
  - *Type 2: context-free*

$$D = \{\theta \to \psi\} \quad \theta \in I \quad \psi \in I \cup A$$

  - *Type 3: finite state* or *regular* $D = \{\alpha \to z\beta, \alpha \to z\} \quad \alpha, \beta \in I \quad z \in A$
- *Chomsky Normal Form* (CNF)
  - a context-free language can be replaced by another language in CNF

$$D = \{\alpha \to \beta\gamma, \alpha \to z\} \quad \alpha, \beta, \gamma \in I \quad z \in A$$

An Example of FSG
(Specified by Terminal Symbols)



Deterministic or Stochastic FSG

An Example of FSG (Specified by Non-Terminal Symbols)

- How to pronounce a six digit sequence?
  - Alphabet terminal set: *A* = {*one, two, … , ten, eleven, … , twenty, … , ninety, hundred, thousand*}
  - Non-terminal (intermediate) set: *I* = {*digit6, digit3, digit2, digit1, teens, tys*}

FSG Derivation Rules with Non-Terminals)

- 8 Rewrite (Derivation) rules with root *S* = *digit6*

$$D = \left\{ \begin{array}{c} digit\,6 \rightarrow digit\,3 \quad thousand \quad digit\,3 \\ digit\,6 \rightarrow digit\,3 \quad thousand \quad OR \quad digit\,3 \\ digit\,3 \rightarrow digit\,1 \quad hundred \quad digit\,2 \\ digit\,3 \rightarrow digit\,1 \quad hundred \quad OR \quad digit\,2 \\ digit\,2 \rightarrow teens \quad OR \quad tys \quad OR \quad tys \quad digit\,1 \quad OR \quad digit\,1 \\ digit\,1 \rightarrow one \quad OR \quad two \quad OR \quad \ldots \quad OR \quad nine \\ teens \rightarrow ten \quad OR \quad eleven \quad OR \quad \ldots \quad OR \quad nineteen \\ tys \rightarrow twenty \quad OR \quad thirty \quad OR \quad \ldots \quad OR \quad ninety \end{array} \right\}$$

Language, Grammar and Parsing

- Language generation rules
- Parsing: Given a test string *x* in a language, find a sequence of derivation rules that leads to *x*
  - *Recognition:* testing if *x* is in *L(G)* by parsing (debuggling)
  - *Generation*: forming a derivation from the root to a sentence
  - Parsing is a way to derive ***structures*** of a language
- Cocke-Younger-Kasami (CYK) Algorithm
  - Starting with the testing sentence *x*, find rewrite rules whose right-hand side matches with part of the current string
  - Replace the string with a segment that could have produced it
  - Generate a *parse table* from the bottom up (*bottom-up parsing*)
  - Continue the process until reaching the root symbol
  - Express the grammar in CNF before parsing

Other Parsing Strategies

- Top-down parsing
  - with some constraints, e.g. the left symbol)

- Specific strategies for specific grammars
  - e.g. Viterbi algorithm for FSG (left-to-right parsing)
  - e.g. inside-outside parsing for CSG
  - Forward-backward algorithm for computing probabilities

- Statistical Parsing without grammatical rules
  - the Lancaster housewife example (recent revolution)
  - statistical translation (from IBM to Aachen to Google)

Bottom-Up Parsing

- An Illustration Example
  - *A={a,b}, I={X,Y,Z}, S, D={d1,d2,d3,d4}={d1: S=>XY OR YZ; d2: X=>YX OR a; d3: Y=>ZZ OR b; d4: Z=>XY OR a}*
  - *x="baaba"="x1, x2, … , xn"*
  - Dividing the candidates into smaller substrings
  - Three search loops, complexity O(n*n*n)
  - Rule sequence: {d1,d2,d3,d4,d3,d2,d2,d3,d4}

Parse Tree

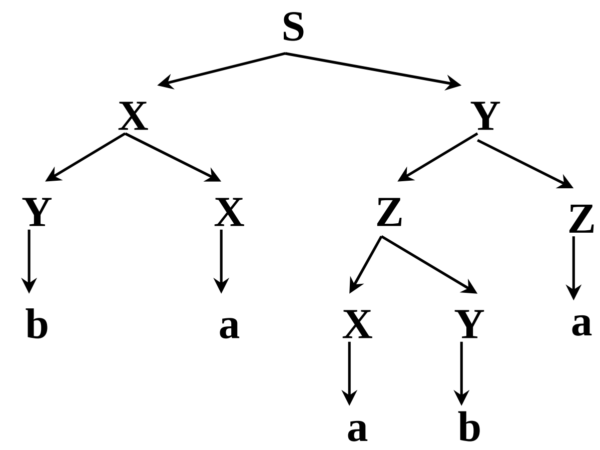
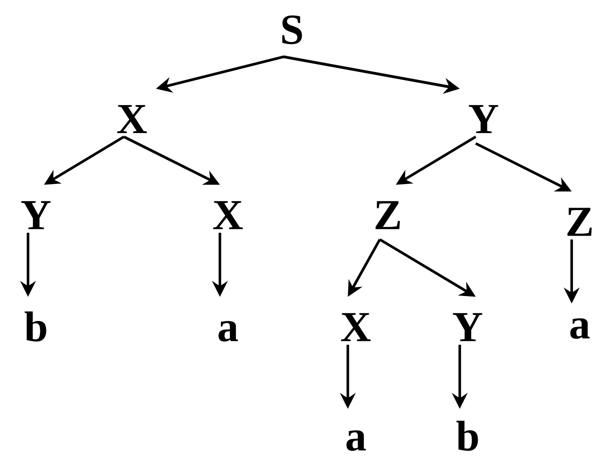
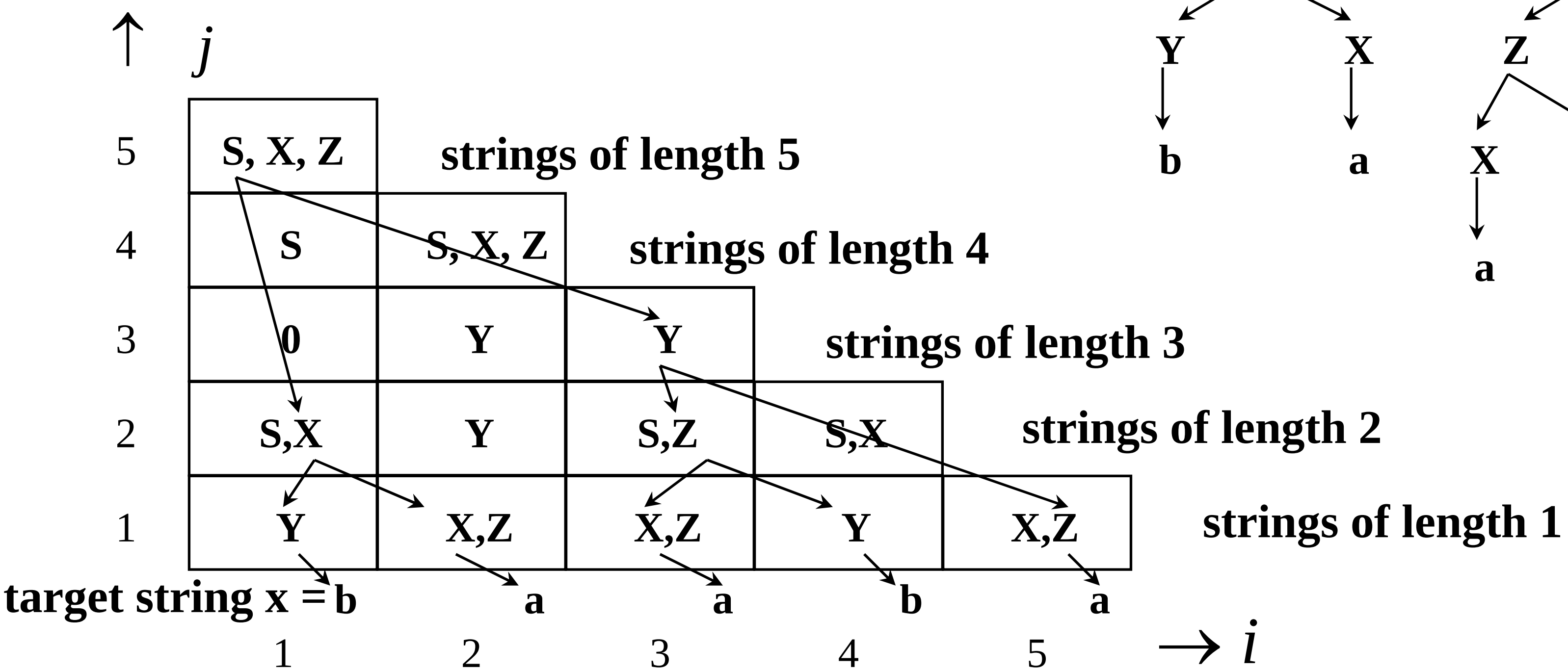


Chart Parsing

Parse Tree





HMM Definition and Parameters

- HMM is a probabilistic regular grammar (PRG)

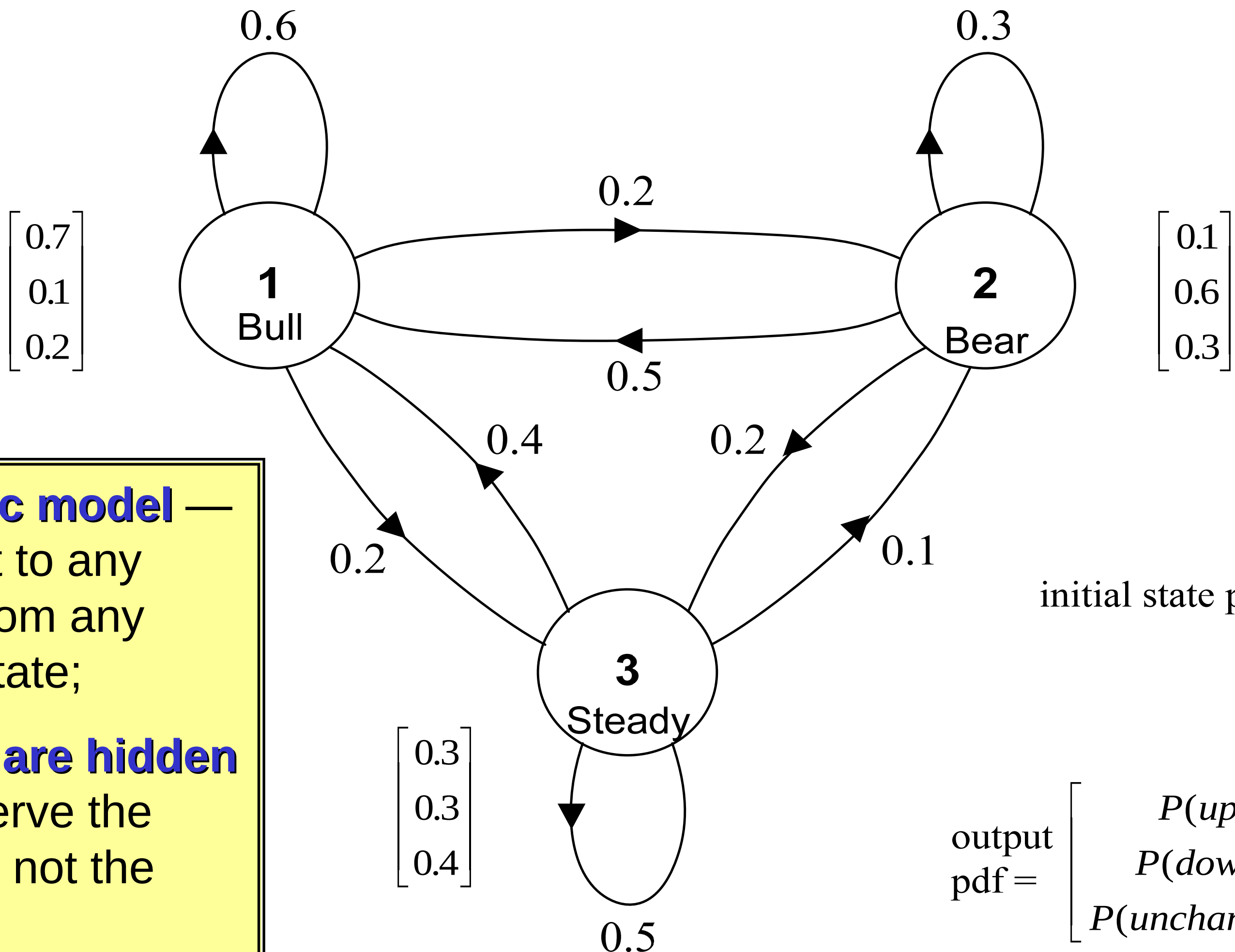
$$P(W \mid G) = \sum_t P(w_1, \ldots, w_Q \mid t) P(t) = \sum_t \pi_{X_0} \prod_{t=1}^{T} a_{X_{t-1} X_t} b_{X_t s_t}$$

Initial prob. : $\pi = (\pi_1, \ldots, \pi_N)$

Transition prob. : $A = (a_{ij}) \quad 1 \le i \le N \quad 1 \le j \le N$

State prob. : $B = (b_{jk}) \quad 1 \le j \le N \quad 1 \le k \le K$

Hidden Markov Models



$$\text{initial state prob.} = \begin{bmatrix} 0.5 \\ 0.2 \\ 0.3 \end{bmatrix}$$

$$\begin{array}{l}\text{output} \\ \text{pdf} = \end{array} \begin{bmatrix} P(up) \\ P(down) \\ P(unchanged) \end{bmatrix}$$

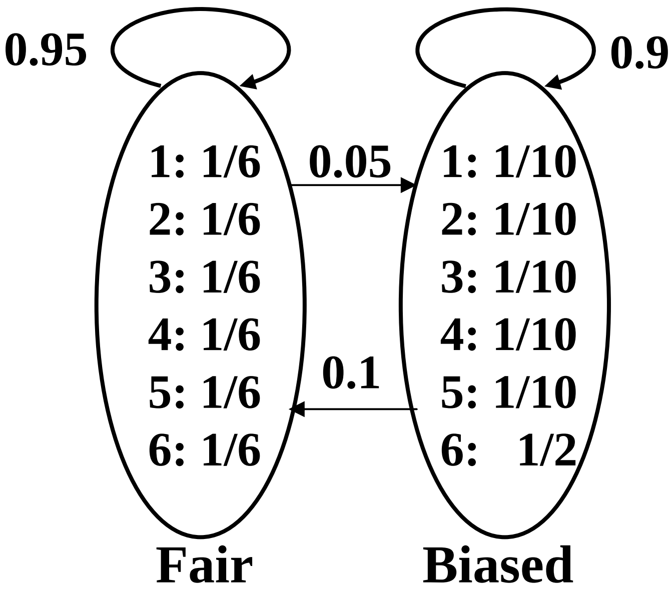
Ergodic model — can get to any state from any other state;

States are hidden — observe the effects, not the states

HMM Computation and Inference

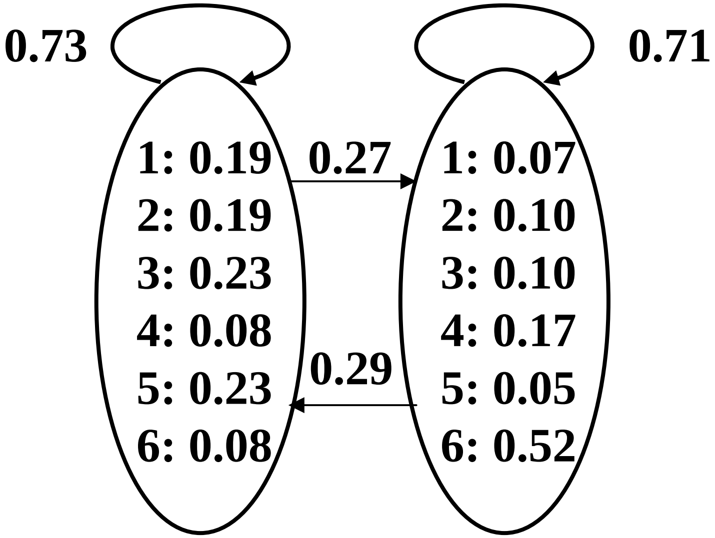
- Problem 1: Evaluation
  - How to compute *P(W|G)* efficiently?
  - Computing forward and backward probabilities over strings of certain length according to RPG derivation rules
- Problem 2: Decoding
  - *Viterbi*: finding the most likely derivation sequence
  - Derivation is always left to right (first to the last word)
- Problem 3: Parameter Estimation (Learning)
  - Given a set of observations *W*, determine the unknown values of the set of parameters

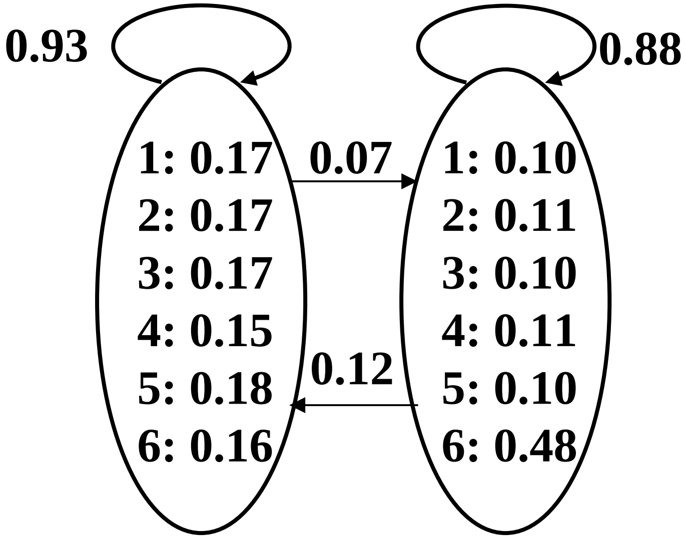
HMM: An Occasionally Dishonest Casino



- **Assume: A casino switches occasionally to a biased dice to increase winning odds !!**
- **Can we model it with HMM ?**
- **How do we prove it cheats ?**
- **Can we estimate the HMM ?**
- **How many samples needed ?**
- **Which dice used at what time?**

Estimation: More vs. Less Data





Estimates with 300 rolls

Estimates with 30000 rolls

Properties of PCFG (for Reference)

- Place Invariance
  - Probability of a subtree does not depend on where in the sentence it dominates (spanning from *p* to *q*)

$$P(I_j(w_k,\ldots,w_{k+c}) = I_j(k,k+c) \to H_i) \quad \text{same} \quad \forall k,i,j$$

  - Same as in HMM for time invariance
- Context-Free
  - Probability of a subtree does not depend on words it does not dominates (spanning from *p* to *q*)

$$P(I_j(k,l) \to H_i \,|\, \text{outside}-\text{words}) = P(I_j(k,l) \to H_i)$$

- Ancestor-Free
  - Probability of a subtree does not depend on any derivation outside the subtree (spanning from *p* to *q*)

$$P(I_j(k,l) \to H_i \,|\, \text{outside}-\text{subtrees}) = P(I_j(k,l) \to H_i)$$

Probabilistic Context Free Grammar (PCFG)

***G* = {*A*, *I*, *S*, *D*, *P*(*D*)}**

$$A = \{w_1, \ldots, w_V\}$$

$$I = \{I_1, \ldots, I_Q\} \quad \text{with} \quad S = I_1$$

$$D = \{I_i \rightarrow H_j\} \quad \text{with} \quad j = 1, \ldots, J_i$$

$$\forall i \quad \sum_{j=1}^{J_i} P(I_i \rightarrow H_j) = 1 \quad H_j = \text{symbol-sequence}$$

- Probability of a word sequence *W* according to *G*

$$P(W \mid G) = P(w_1^M \mid G) = \sum_t P(w_1^M \mid t) P(t) \quad t: \text{parse-tree}$$

- Probability of a parse tree (score and compare)

$$P(t) = P(d_1^L) = \prod_{j=1}^{L} P(d_j) \quad d_j: \text{parse-tree-rule}$$

PCFG Computation and Inference

- Problem 1: Evaluation
  - How to compute *P(W|G)* efficiently?
  - Computing inside and outside probabilities
    - *inside-outside* algorithm for re-estimation
- Problem 2: Decoding
  - *Viterbi* algorithm: finding the most likely parse tree which also implies the most likely derivation sequence
  - Bayes Theorem: $\hat{t} = \arg\max_t P(t|W) = \arg\max_t P(W|t)P(t)$
- Problem 3: Parameter Estimation (Learning)
  - Given a set of observations *W*, determine the unknown values of the set of parameters (much more involved)

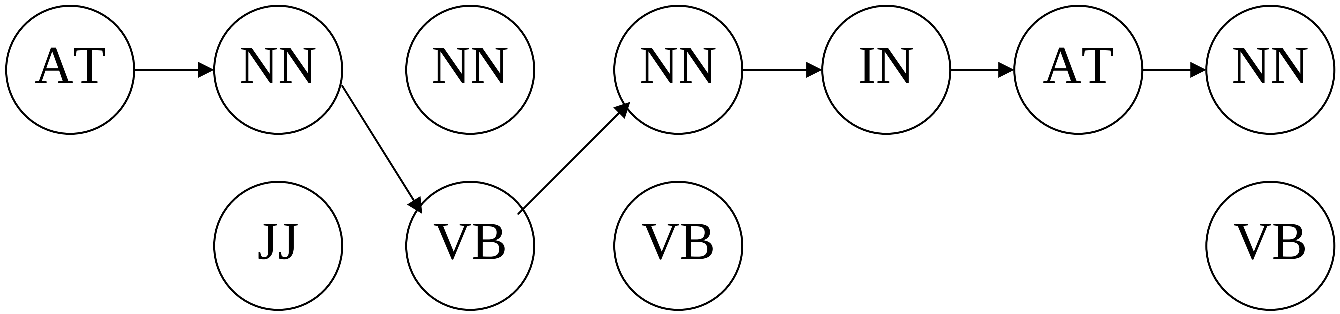
$$\theta = \{a_{ji} = P(I_j \to H_i):\ \ 1 \le i \le J_i,\ \ 1 \le j \le Q\}$$

- Countable State HMM (instead of finite state HMM)

Problem Mapping of POS Tagging

- Finite state network (FSN) representation
  - State (node) space: the set of tags
  - Arc: tag transition (probabilities)
  - State output: tag-specific word probabilities
  - State-sequence: tag sequence
- An example:

The representative put chairs on the table.



Statistical POS Tagging

$$\hat{T} = \arg\max_{T \in \Psi} P(T \mid W)$$
$$= \arg\max_{T \in \Psi} P(W \mid T) P(T)$$

P(W|T): tag-specific word LM

P(T): tag language model

- Bigram tag language model approximation

$$P(T) = P(t_1^Q) \approx \prod_{q=1}^{Q} P(t_q \mid t_{q-1}) \quad P(t_1 \mid t_0) = 1$$

- Localized tag-specific language model

$$P(W \mid T) = P(w_1^Q \mid t_1^Q) \approx \prod_{q=1}^{Q} P(w_q \mid t_1^n) \approx \prod_{q=1}^{Q} P(w_q \mid t_q)$$

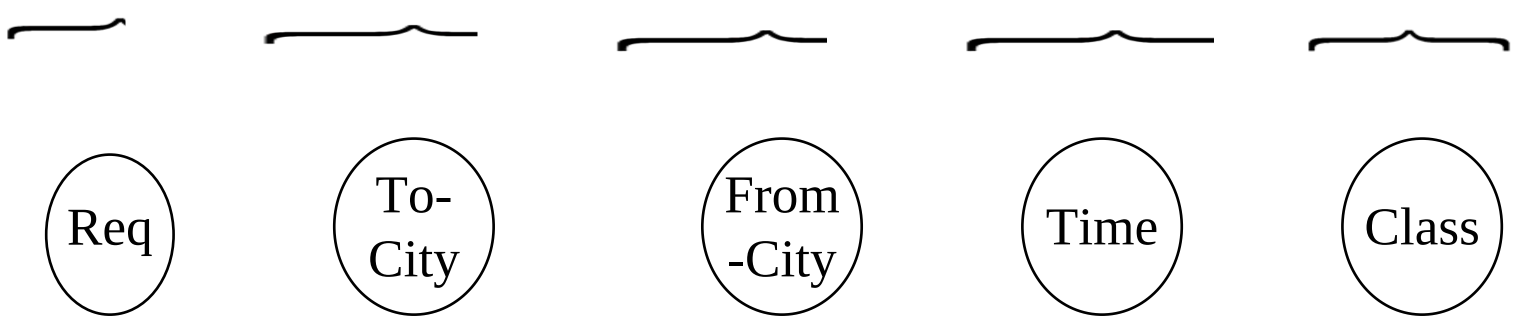
- Overall approximation

$$\hat{t}_1^Q = \arg\max_T P(W \mid T) P(T) \approx \arg\max_{t_1^Q} \prod_{q=1}^{Q} P(w_q \mid t_q) P(t_q \mid t_{q-1})$$

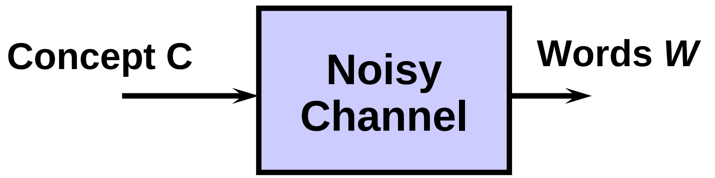
Problem Mapping for Text Understanding

- Finite state network (FSN) representation
  - State (node) space: the set of concepts
  - Arc: concept transition (probabilities)
  - State output: concept-specific word sequences
  - State-sequence: concept sequence (meaning expressed in sequence of semantic attributes)
- An example:

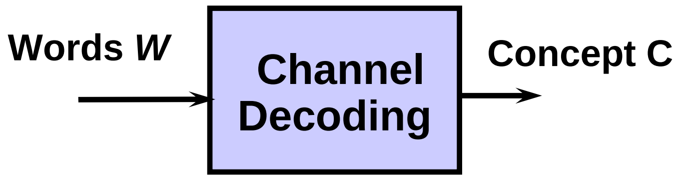
I want to fly to Boston from Dallas Friday noon on coach.



Statistical Concept Decoding



$$\hat{C} = \mathrm{argmax}_{C \in \Psi} P(C \mid W)$$
$$= \mathrm{argmax}_{C \in \Psi} P(W \mid C) P(C)$$



P(W|C): concept-specific word LM

P(C): concept language model

- Bigram concept language model approximation

$$P(C) = P(c_1^Q) \approx \prod_{q=1}^{Q} P(c_q \mid c_{q-1}) \quad P(c_1 \mid c_0) = 1$$

- Localized concept-specific bigram or trigram LM

$$P(W \mid C) = P(w_1^Q \mid c_1^Q) \approx \prod_{q=1}^{Q} P(w_1^Q \mid c_q) \approx \prod_{q=1}^{Q} P(w_{q-2}^q \mid c_q)$$

- Overall approximation

$$\hat{c}_1^Q = \arg\max_C P(W \mid C) P(C) \approx \arg\max_{c_1^Q} \prod_{q=1}^{Q} P(w_{q-2}^q \mid c_q) P(c_q \mid c_{q-1})$$

Grammatical Inference

- There are some techniques but the general notion of *grammatical inference* is not easily tractable
  - Usually designed by hand with human experts
  - The number of rules and language coverage are key issues
  - Corpus-based learning approaches are now being explored
  - Probabilistic approaches offer a good way to score parse trees and are capable of handling flexible grammars (*robust parsing* even with ill-formed sentences, a highly desirable property)

Language Acquisition & Inference

- Problem Statement
  - Given a set of sentence samples, find $G = \{A, I, S, D\}$
  - Usually underspecified (few samples but too many solutions)
- A Rule-Based Strategy (Generalization?)
  - Divide sentences into positive (x+) and negative (x-) examples
  - Start with a guessing grammar G0 (e.g. from known rules)
  - Test G0 on the x+ sentences one by one, add rules if needed and make sure new rules do not part x- examples, update G0

A Grammatical Inference Example

U+={a,aaa,aaab,aab}, U-={ab,abc,abb,aabb}, A={a,b}, I={X},

iter	u+	*D*	*D => U-?*
1	a	S -> X X -> a	No
2	aaa	S -> X X -> a X -> aX	No
3	aaab	S -> X X -> a X -> aX X -> ab (-) X -> aab (+)	Yes for “ab” in X-, the 4th rule needs to be removed and the 5th rule is added
4	aab	No new rules (Done !!)	No

initialize D={S -> X}

Some Issues before Moving on

- Problems with PCFG estimation
  - Many unsolved research issues: *less studied, more rewards*
  - Sizes of *A* and *I* often unknown: O(M*M*M*Q*Q*Q)
  - Too little data to estimate too many parameters
  - But we can not ignore unobserved events
  - Greater *A* and *I* imply more estimation & storage problem
  - Techniques in search, N-gram and HMM can be extended
- Parsing for disambiguation and understanding?
  - Probabilities for determining the sentence
  - Probabilities for speedier parsing (pruning efficiency)
  - Probabilities for choosing between parses (ranking/scoring)
- Labeled corpra for learning – treebank and others
  - Chunking (bracketing): the first step to studying parsing
  - Penn Treebank: widely used, large size; other languages

More Issues before Moving on

- Other probabilistic grammars
  - Probabilistic left-corner grammar
  - Probabilistic dependency grammar
  - Probabilistic history-based grammar
  - Probabilistic tree-adjoining grammar
- Other learning approaches
  - Knowledge-based detailed refinement (learning from ASR)
  - Unsupervised learning – how much labeling is needed?
  - Transformation-based learning (decision-feedback)
- Other search algorithms
  - stack decoding, A* search, beam search
- Other notions on parsing
  - Data-driven (non-lexicalized, non-grammatical approaches)

Summary

- Today's Class
  - Linguistics Foundations
    - formal grammars and Chomsky normal form
    - grammatical inference & language acquisition
    - probabilistic finite state grammar (PFSG)
    - probabilistic context-free grammar (PCFG)
  - Lab1 due on Jan. 23
- Next Classes
  - Class project list and corpus-based study
- Reading Assignments
  - Manning and Schutze, Chapters 2 & 3