Tema 7

Inferência de Gramáticas Livres de Contexto

Professora:
Ariane Machado Lima

EACH USP

Inferência gramatical (revisão)

- Identificação no limite
  - Objetivo: identificar uma linguagem alvo $\mathbb{L}$
  - G_i = gramática inferida a partir de uma amostra $S^+ = \{x_1, x_2, \ldots, x_i\}$
  - Para j >= t exemplos, G_j não muda e $L(G_j) = \mathbb{L}$

Inferência gramatical (revisão)

- Identificação no limite
- Alguns resultados:
  - **Nenhuma** classe de linguagens super-finita (que contém todas as finitas e pelo menos uma infinita) pode ser identificada no limite apenas a partir de ***texto*** (ie, apenas exemplos positivos)
    - INCLUI A CLASSE DE LING. LIVRES DE CONT.

Inferência gramatical

- Objetivo: aprender uma gramática G^t alvo
- **Uso de oráculos**
  - **Perguntas de pertencimento**: a cadeia x é gerada por G^t?
  - **Perguntas de equivalência**: uma dada gramática G^h é equivalente à G^t? Se não, o oráculo retorna um contra-exemplo
    - (lembrando que o problema de equivalência entre duas GLCs é indecidível)

EACH USP

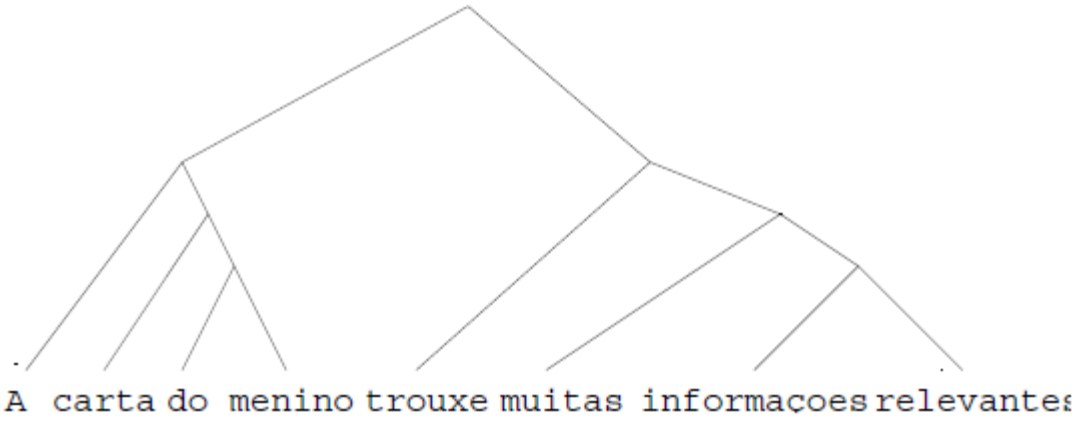
Inferência de GLCs

- A classe de GLCs **não** pode ser identificada (no limite) em tempo polinomial a partir apenas de texto e de um oráculo (perguntas de pertencimento e de equivalência) (Sakakibara, 1995)

EACH USP

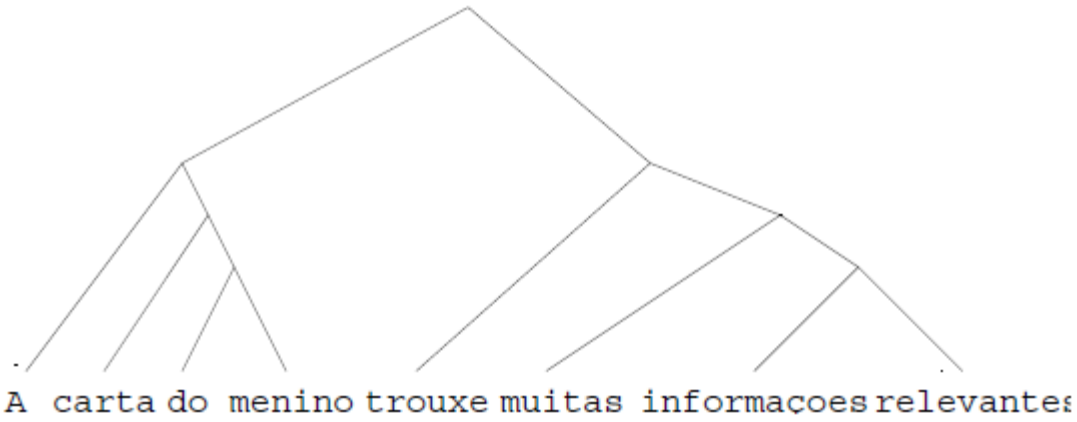
Inferência gramatical **de GLCs**

- Objetivo: aprender uma gramática G^t alvo
- **Uso de informação estrutural (esqueleto)**
  - Cadeia x seguida de sua árvore sintática (segundo G^t) mas sem rotulação nos nós internos. Ex:



Inferência gramatical **de GLCs**

- Objetivo: aprender uma gramática G^t alvo
- **Uso de informação estrutural (esqueleto)**
  - Cadeia x seguida de sua árvore sintática (segundo G^t) mas sem rotulação nos nós internos. Ex:



((A (carta (do menino))) (trouxe (muitas (informações relevantes))))

Inferência gramatical **de GLCs**

- Estratégias para inferência de GLCs:
  - Uso de informação estrutural → permite a identificação de GLCs
  - Uso de oráculos com informação estrutural (perguntas de pertencimento de uma cadeia estruturada, perguntas de equivalência estrutural)
  - Restrição do espaço de busca para subclasses de GLCs

EACH USP

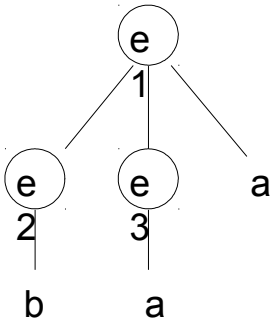
Inferência de gramáticas reversíveis

- É possível identificar no limite a classe de gramáticas ***reversíveis*** (uma subclasse das GLCs) utilizando exemplos estruturados (Sakakibara, 1992)
- Uma gramática é reversível se:
  - Se $A \rightarrow \alpha$ e $B \rightarrow \alpha$ então $A = B$
  - Se $A \rightarrow \alpha B \beta$ e $A \rightarrow \alpha C \beta$ então $B = C$
- Detalhe importante: essa classe de *gramáticas* é capaz de gerar toda a classe de *linguagens* livres de contexto.

EACH USP

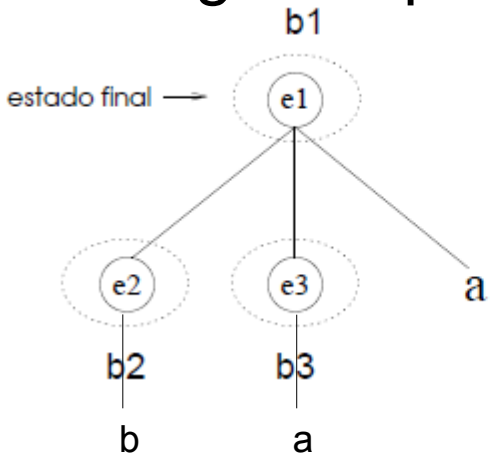
Inferência de gramáticas reversíveis

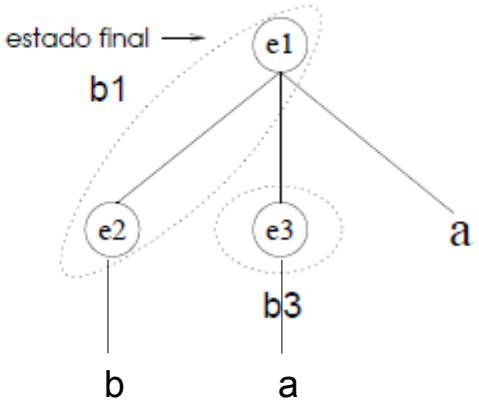
- Uso de uma estrutura chamada *Tree Automaton* (autômato árvore), similar uma árvore sintática
- Tree automaton = (Q, V, δ, F)
  - Nós internos: estados (Q)
  - Nós folhas: símbolos terminais (V)
  - Relação nó pai – filhos: transições (δ)
  - Nó raiz: estado final (F)



Tree Automaton

- Estados particionados em blocos
- Cada bloco corresponde a um símbolo não terminal da gramática
- Relação pai – filhos: regra de produção
- Exemplos:





<b1> → <b2><b3>a
<b2> → b
<b3> → a

<b1> → <b1><b3>a
<b1> → b
<b3> → a

Inferência de gramáticas reversíveis

- Ideia do algoritmo:
  - Monta um autômato a árvore para cada cadeia da amostra de treinamento (com informação estrutural), cada estado formando um bloco unitário (partição trivial)
  - Une todos os estados finais (raízes) em um único bloco → um único autômato a árvore que reconhece exatamente a amostra de treinamento
  - Sucessivos testes e uniões de blocos para garantir as duas propriedades de reversibilidade

EACH USP

Procedimento:

Crie o autômato a árvore $A = (Q, V, \delta, F)$ formado a partir da amostra S

Crie uma partição trivial π_0 sobre Q

$q \leftarrow$ algum estado pertencente a F

$lista \leftarrow$ todos os pares (q, q') onde $q' \in F - \{q\}$

$i \leftarrow 0$

enquanto $(lista \neq \emptyset)$ faça

    remova o primeiro par (q_1, q_2) de $lista$

    %$B(q, \pi_i)$ retorna o bloco do estado q dentro da partição π_i

    $B_1 \leftarrow B(q_1, \pi_0)$ e $B_2 \leftarrow B(q_2, \pi_0)$

    se $B_1 \neq B_2$

      $\pi_{i+1} \leftarrow \pi_i$ com B_1 e B_2 agrupados

      VerificaPropriedade1(π_{i+1})

      VerificaPropriedade2(π_{i+1}, B_1, B_2)

      $i \leftarrow i + 1$

fim

Exercício: simular o algoritmo sobre essa amostra S:

((bb(ab)a)(ba))
((ab)a)
(bab(ab))
(bb(ab)a)
((ba)(ba))

Procedimento:

Crie o autômato a árvore $A = (Q, V, \delta, F)$ formado a partir da amostra S

Crie uma partição trivial π_0 sobre Q

$q \leftarrow$ algum estado pertencente a F

$lista \leftarrow$ todos os pares (q, q') onde $q' \in F - \{q\}$

$i \leftarrow 0$

enquanto $(lista \neq \emptyset)$ faça

    remova o primeiro par (q_1, q_2) de $lista$

    $\%B(q, \pi_i)$ retorna o bloco do estado q dentro da partição π_i

    $B_1 \leftarrow B(q_1, \pi_0)$ e $B_2 \leftarrow B(q_2, \pi_0)$

    se $B_1 \neq B_2$

        $\pi_{i+1} \leftarrow \pi_i$ com B_1 e B_2 agrupados

        VerificaPropriedade1(π_{i+1})

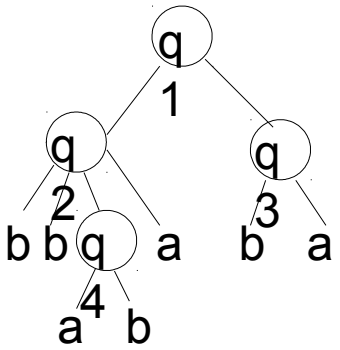
        VerificaPropriedade2(π_{i+1}, B_1, B_2)

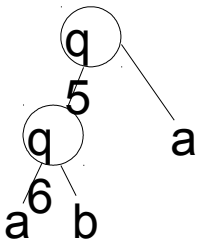
        $i \leftarrow i + 1$

fim

Exercício: simular o algoritmo sobre essa amostra S:

((bb(ab)a)(ba))
((ab)a)
(bab(ab))
(bb(ab)a)
((ba)(ba))





..

.

F = {q1, q5, ...}

Procedimento:

Crie o autômato a árvore $A = (Q, V, \delta, F)$ formado a partir da amostra S

Crie uma partição trivial π_0 sobre Q 

$q \leftarrow$ algum estado pertencente a F

$lista \leftarrow$ todos os pares (q, q') onde $q' \in F - \{q\}$

$i \leftarrow 0$

enquanto $(lista \neq \emptyset)$ faça

    remova o primeiro par (q_1, q_2) de $lista$

    $\%B(q, \pi_i)$ retorna o bloco do estado q dentro da partição π_i

    $B_1 \leftarrow B(q_1, \pi_0)$ e $B_2 \leftarrow B(q_2, \pi_0)$

    se $B_1 \neq B_2$

      $\pi_{i+1} \leftarrow \pi_i$ com B_1 e B_2 agrupados

      VerificaPropriedade1(π_{i+1})

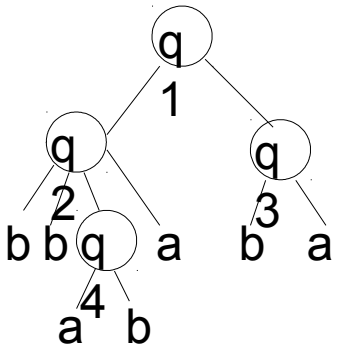
      VerificaPropriedade2(π_{i+1}, B_1, B_2)

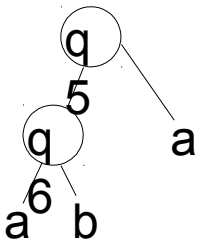
      $i \leftarrow i + 1$

fim

Exercício: simular o algoritmo sobre essa amostra S:

((bb(ab)a)(ba))
((ab)a)
(bab(ab))
(bb(ab)a)
((ba)(ba))





..

.

F = {q1, q5, ...}

π_0 =
B1 = {q1}
B2 = {q2}
B3 = {q3}
B4 = {q4}
B5 = {q5}
B6 = {q6}
...

Procedimento:

Crie o autômato a árvore $A = (Q, V, \delta, F)$ formado a partir da amostra S

Crie uma partição trivial π_0 sobre Q

$q \leftarrow$ algum estado pertencente a F

E assim por diante...

$lista \leftarrow$ todos os pares (q, q') onde $q' \in F - \{q\}$

$i \leftarrow 0$

enquanto $(lista \neq \emptyset)$ faça

    remova o primeiro par (q_1, q_2) de $lista$

    $\%B(q, \pi_i)$ retorna o bloco do estado q dentro da partição π_i

    $B_1 \leftarrow B(q_1, \pi_0)$ e $B_2 \leftarrow B(q_2, \pi_0)$

    se $B_1 \neq B_2$

        $\pi_{i+1} \leftarrow \pi_i$ com B_1 e B_2 agrupados

        VerificaPropriedade1(π_{i+1})

        VerificaPropriedade2(π_{i+1}, B_1, B_2)

        $i \leftarrow i + 1$

fim

EACH USP

Propriedade 1:

Se A→α e B→ α então A = B

VerificaPropriedade1(π_{i+1}):

para todos os pares de estados $(q, q') \in Q \times Q$

se q e q' têm o mesmo número de descendentes

q = pai$(u_1, u_2, ..., u_k)$ e q' = pai$(u'_1, u'_2, ..., u'_k)$

se $((u_j, u'_j \in Q$ e $B(u_j, \pi_{i+1}) = B(u'_j, \pi_{i+1}))$ ou $u_j = u'_j \in V)$ para 1 $\leq$ j $\leq$ k

e $B(q, \pi_{i+1}) \neq B(q', \pi_{i+1})$

adicione o par (q, q') à *lista*

Propriedade 2:

Se $A \rightarrow \alpha B \beta$ e $A \rightarrow \alpha C \beta$ então $B = C$

VerificaPropriedade2(π_{i+1}, B_1, B_2)**:**

para todos os pares de estados $(q, q') \in B_1 \times B_2$

se q e q' têm o mesmo número de descendentes

q = pai$(u_1, u_2, ..., u_k)$ e q' = pai$(u'_1, u'_2, ..., u'_k)$

se $((u_l, u'_l \in Q$ e $B(u_l, \pi_{i+1}) \neq B(u'_l, \pi_{i+1}))$ para algum l $(1 \leq l \leq k)$

e $B(u_j, \pi_{i+1}) = B(u'_j, \pi_{i+1})$ ou $u_j = u'_j \in V$, para $1 \leq j \leq k$, $j \neq l$

adicione o par (u_l, u'_l) à *lista*

EACH USP

Estimação de probabilidades de GLCEs

Gramáticas Estocásticas

- Definição: uma gramática estocática G é uma quíntupla (V, Σ, S, P, ρ), onde
  - V é o conjunto de símbolos não-terminais (variáveis)
  - Σ é o conjunto de símbolos terminais
  - S é o símbolo inicial
  - P é o conjunto de produções da forma

    $(\Sigma \cup V)^* V (\Sigma \cup V)^* \rightarrow (\Sigma \cup V)^*$
  - ρ é o conjunto de distribuições de probabilidades sobre as produções de mesmo lado esquerdo

    $\sum_i \rho(\alpha \rightarrow \beta_i) = 1$

Estimação por máxima verossimilhança ou máxima a posteriori

- Utiliza-se um contador para cada regra de produção da gramática

- Utiliza-se um analisador sintático para analisar a amostra de treinamento: cada vez que uma regra é utilizada incrementa-se seu contador
- Ao final, normaliza-se os contadores para que a soma dos “contadores” das produções com o mesmo lado esquedo some 1 (isto é, vira probabilidade)

EACH USP

Estimação por máxima verossimilhança ou máxima a posteriori

- Utiliza-se um contador para cada regra de produção da gramática
  - contador inicializado com 0: **máxima verossimilhança**
  - contador inicializados com um valor > 0 (pseudocontador): **máxima a posteriori**
- Utiliza-se um analisador sintático para analisar a amostra de treinamento: cada vez que uma regra é utilizada incrementa-se seu contador
- Ao final, normaliza-se os contadores para que a soma dos "contadores" das produções com o mesmo lado esquedo some 1 (isto é, vira probabilidade)

EACH USP

Estimação de probabilidades de GLCEs

- Pergunta relevante: sua gramática é ambígua? Se for, você quer estimar as probabilidades com base em todas as árvores sintáticas (da amostra de treinamento) ou apenas na "correta"?
- Se for baseada na "correta": você possui uma amostra de treinamento com informação estrutural ?

EACH USP

Estimação por máxima verossimilhança ou máxima a posteriori

- Se a amostra possui informação estrutural, a estrutura ajuda a guiar a análise sintática (principalmente se a cadeia sem estrutura é gerada ambiguamente pela gramática)
- Se não, e a gramática for ambígua, você pode estimar as probabilidades com base em todas as árvores sintáticas da amostra de treinamento, ou usar uma técnica EM...

Algoritmos EM de estimação de probabilidades de GLCE

- EM (Expectation-Maximization)
- Algoritmo base:
  - Inicialização das probabilidades da gramática
  - Loop:
    - Encontra a(s) árvore(s) sintática(s) de cada cadeia da amostra de treinamento
    - Reestima as probabilidades da gramática
  - Até convergir
- Algoritmos:
  - **Inside-outside (para gramáticas na forma normal de Chomsky)**
  - Tree Grammar EM (para gramáticas em qualquer formato)

EACH USP

Inside-Outside

- Gramáticas precisam estar na forma normal de Chomsky (baseia-se no CYK)
- É o equivalente para GLCE do algoritmo Baum-Welch de HMMs (forward-backward)
- Probabilidade forward (f) → inner (I)
- Probabilidade backward (b) → outer (O)

EACH USP

Forma Normal de Chomsky

Uma GLC está na Forma Normal de Chomsky se:

a) Toda regra de produção é da forma

$$A \rightarrow BC \quad \text{ou } A \rightarrow a$$

sendo B,C variáveis, a um símbolo terminal;

b) A variável inicial S não pode aparecer no lado direito de nenhuma regra;

c) Somente a variável inicial pode ter a regra

$$S \rightarrow \varepsilon \ .$$

Inside-Outside

- Matrizes $A_{|V|x|V|x|V|}$ e $B_{|V|x|\Sigma|}$:
  - A: $a[\upsilon,y,z] = P(V[\upsilon] \rightarrow V[y]V[z])$
  - B: $b[\upsilon,m] = P(V[\upsilon] \rightarrow \Sigma[m])$
  - $c : \Sigma \rightarrow \mathbb{N}$ (mapeia um símbolo em seu índice)

$$\Sigma_{y,z}\, a[\upsilon,y,z] + \Sigma_{m}\, b[\upsilon,m] = 1 \text{ para todo } \upsilon$$

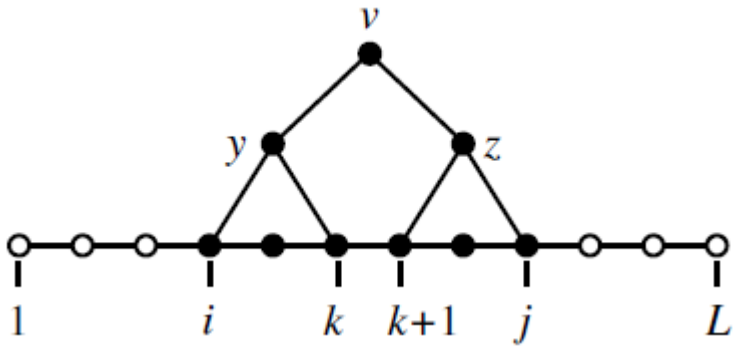
Inside

- Dada uma sequência x
- Inner I(i,j,υ): probabilidade da subsequência $x_i...x_j$ ser gerada pelo não terminal υ
- Caso 1: i = j

$$I(i,i,\upsilon) = P(V[\upsilon] \rightarrow x_i)$$

Inside

- Dada uma sequência x
- Inner I(i,j,υ): probabilidade da subsequência $x_i...x_j$ ser gerada pelo não terminal υ
- Caso 2: i ≠ j



$$I(i,j,\upsilon) = \Sigma_{y,z} \Sigma_{k=i}^{j-1} I(i,k,y)\, I(k+1,j,z)\, P(V[\upsilon] \rightarrow V[y]V[z])$$

Inside

- Algoritmo Inside: calcula $P(x \mid G)$, $x = x_1 \ldots x_L$
- Matriz de programação dinâmica $I_{L \times L \times |V|}$

Inicialização: para i = 1 até L, υ = 1 até |V|

$$I(i,i,\upsilon) = P(V[\upsilon] \rightarrow x_i)$$

Iteração: para i = 1 até L-1, j = i+1 até L, υ = 1 até |V|

$$I(i,j,\upsilon) = \Sigma_{y,z} \Sigma_{k=i}^{j-1} I(i,k,y)\, I(k+1,j,z)\, P(V[\upsilon] \rightarrow V[y]V[z])$$

Término: $P(x|G) = I(1, L, 1)$

EACH USP

Inside

- Algoritmo Inside: calcula $P(x \mid G)$, $x = x_1 \ldots x_L$
- Matriz de programação dinâmica $I_{L \times L \times |V|}$

Inicialização: para i = 1 até L, υ = 1 até |V|

$$I(i,i,\upsilon) = b[\upsilon, c(x_i)]$$

Iteração: para i = 1 até L-1, j = i+1 até L, υ = 1 até |V|

$$I(i,j,\upsilon) = \Sigma_{y,z} \Sigma_{k=i}^{j-1} I(i,k,y)\, I(k+1,j,z)\, a[\upsilon, y, z]$$

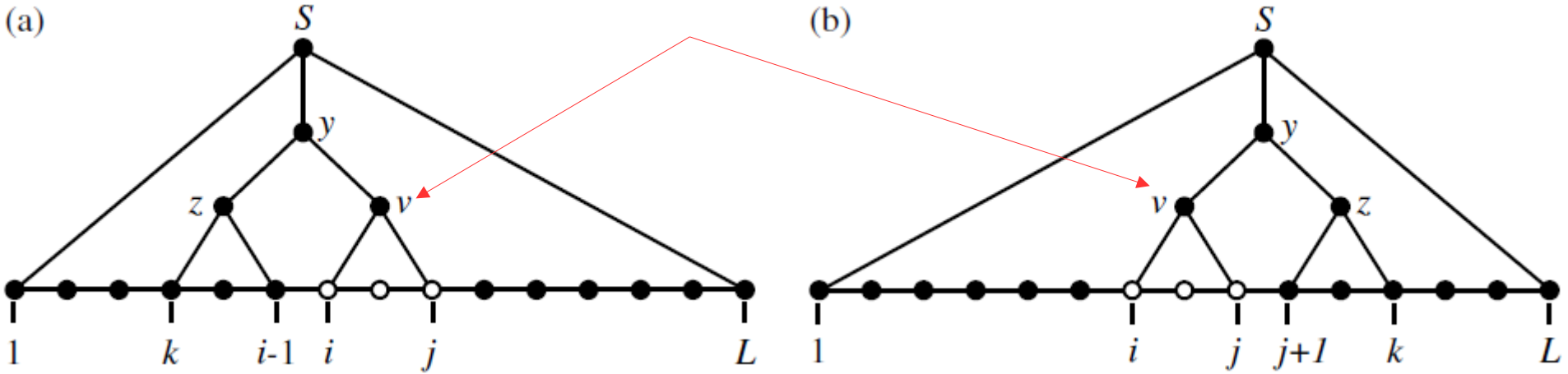
Término: $P(x|G) = I(1, L, 1)$

EACH USP

Outside

- Dada uma sequência x
- Outer O(i,j,υ): probabilidade da sequência x ser gerada por S, excluindo a subárvore com raiz em υ gerando $x_i...x_j$
- $O(i,j,\upsilon) = P(S =>^* x_1...x_{i-1}\ V[\upsilon]\ x_{j+1}...x_L|G)$

Outside



- $O(i,j,\upsilon) = P(S \Rightarrow^* x_1 \ldots x_{i-1}\ V[\upsilon]\ x_{j+1} \ldots x_L | G)$

$$= \Sigma_{y,z} \Sigma_{k=1}^{i-1} I(k,i-1,z)\, I(k,j,y)\, P(V[y] \rightarrow V[z]V[\upsilon])$$

$$+ \Sigma_{y,z} \Sigma_{k=j+1}^{L} I(j+1,k,z)\, I(i,k,y)\, P(V[y] \rightarrow V[z]V[\upsilon])$$

Outside

- Algoritmo Outside: calcula $P(x \mid G)$, $x = x_1 \ldots x_L$
- Matriz de programação dinâmica $O_{L \times L \times |V|}$

Inicialização: $O(1,L,1) = 1$

$$O(1,L,\upsilon) = 0 \text{ para todo } \upsilon \neq 1$$

Iteração: para i = 1 até L, j = L até i, υ = 1 até |V|

$$O(i,j,\upsilon) = \Sigma_{y,z}^{i-1} \Sigma_{k=1} I(k,i-1,z)\, I(k,j,y)\, P(V[y] \rightarrow V[z]V[\upsilon])$$

$$+ \Sigma_{y,z} \Sigma_{k=j+1}^{L} I(j+1,k,z)\, I(i,k,y)\, P(V[y] \rightarrow V[z]V[\upsilon])$$

Término: $P(x|G) = \Sigma_{\upsilon} O(i, i, \upsilon)\, P(V[\upsilon] \rightarrow x_i)$ para *qualquer* i

EACH USP

Outside

- Algoritmo Outside: calcula $P(x \mid G)$, $x = x_1 \ldots x_L$
- Matriz de programação dinâmica $O_{L \times L \times |V|}$

Inicialização: $O(1,L,1) = 1$

$O(1,L,\upsilon) = 0$ para todo υ

Iteração: para i = 1 até L, j = L até i, υ = 1 até |V|

$$O(i,j,\upsilon) = \sum_{y,z}^{j-1} \sum_{k=1} I(k,i-1,z)\, I(k,j,y)\, a[y,z,\upsilon]$$

$$+ \sum_{y,z} \sum_{k=j+1}^{L} I(j+1,k,z)\, I(i,k,y)\, a[y,z,\upsilon]$$

Término: $P(x|G) = \sum_{\upsilon} O(i, i, \upsilon)\, b[\upsilon, c(x_i)]$ para *qualquer* i

EACH USP

Inside-Outside

- n(υ): número esperado de vezes que υ é usado em uma derivação
  - $n(\upsilon) = \frac{1}{P(x|G)} \sum_{i=1}^{L} \sum_{j=1}^{L} I(i,j,\upsilon)\, O(i,j,\upsilon)$
- n($\upsilon \rightarrow yz$): número esperado de vezes que a produção $\upsilon \rightarrow yz$ é usada
- $n(\upsilon \rightarrow yz) = \frac{1}{P(x|G)} \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} \sum_{k=i}^{j-1} O(i,j,\upsilon)\, I(i,k,y)\, I(k+1,j,z)\, P(V[\upsilon] \rightarrow V[y]V[z])$
- $\hat{P}(V[\upsilon] \rightarrow V[y]V[z]) = n(\upsilon \rightarrow yz) / n(\upsilon)$
- $\hat{P}(V[\upsilon] \rightarrow \alpha) = n(\upsilon \rightarrow \alpha) / n(\upsilon)$ para $\alpha \in \Sigma$

EACH USP

Algoritmos EM de estimação de probabilidades de GLCE

- EM (Expectation-Maximization)
- Algoritmo base:
  - Inicialização das probabilidades da gramática
  - Loop:
    - Encontra a(s) árvore(s) sintática(s) de cada cadeia da amostra de treinamento
    - Reestima as probabilidades da gramática
  - Até convergir
- Algoritmos:
  - Inside-outside (para gramáticas na forma normal de Chomsky)
  - Tree Grammar EM (para gramáticas em qualquer formato)

EACH USP

Inside-Outside

EXPECTATION

- $n(\upsilon)$: número esperado de vezes que υ é usado em uma derivação
  - $n(\upsilon) = \frac{1}{P(x|G)} \Sigma_{i=1}^{L}\Sigma_{j=1}^{L} I(i,j,\upsilon)\, O(i,j,\upsilon)$
- $n(\upsilon \rightarrow yz)$: número esperado de vezes que a produção $\upsilon \rightarrow yz$ é usada
- $n(\upsilon \rightarrow yz) = \frac{1}{P(x|G)} \Sigma_{i=1}^{L-1}\Sigma_{j=i+1}^{L} \Sigma_{k=i}^{j-1} O(i,j,\upsilon)\, I(i,k,y)\, I(k+1,j,z)\, P(V[\upsilon] \rightarrow V[y]V[z])$

Soma para as várias sequências da amostra de treinamento

- $\hat{P}(V[\upsilon] \rightarrow V[y]V[z]) = n(\upsilon \rightarrow yz) / n(\upsilon)$
- $\hat{P}(V[\upsilon] \rightarrow \alpha) = n(\upsilon \rightarrow \alpha) / n(\upsilon)$ para $\alpha \in \Sigma$

MAXIMIZATION

EACH USP

Referências

- Durbin, R.; Eddy, S. R.; Krogh, A. Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press, 2002. Cap 9
- Lari, K.; Young, S. J. (1990) The estimation of stochastic context-free grammars using the Inside-Outside algorithm. Computer Speech and Language 4:35-56.
- Sakakibara Y. (1992) Efficient learning of context-free grammars from positive structural examples. Information and Computation 97(1):23-60.
- Sakakibara Y. (1995) Grammatical inference: An old and new paradigm. In: Jantke K.P., Shinohara T., Zeugmann T. (eds) Algorithmic Learning Theory. ALT 1995. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol 997. Springer, Berlin, Heidelberg.

EACH USP