Universidade de Brasília
Departamento de Estatística

Estudo de Estimadores de Árvores de Contexto com Aplicação em Linguística

Alex Barros Azevedo Bomfim

Monografia apresentada para obtenção do título de Bacharel em Estatística.

Brasília
2015

Alex Barros Azevedo Bomfim

Estudo de Estimadores de Árvores de Contexto com Aplicação em Linguística

Orientador:
Prof. Dr. **LUCAS MOREIRA**

Monografia apresentada para obtenção do título de Bacharel em Estatística.

Brasília
2015

SUMÁRIO

1 INTRODUÇÃO

O objetivo principal deste trabalho é analisar diferenças entre o Português Brasileiro e o Português Europeu quanto ao ritmo em textos escritos. Para tanto, os textos de ambas as línguas foram codificados em sequências de símbolos finitos que tiveram como base características rítmicas. A classe de modelos que foi utilizada no estudo foram as árvores probabilísticas de contextos.

Analisar diferenças linguísticas entre o Português Brasileiro e o Português Europeu é, em particular, interessante, pois ambos apresentam o mesmo conjunto de palavras em sua estrutura (lexico). No entanto, essas línguas apresentam sintaxes e prosódias distintas, isto é, as palavras são ordenadas de forma diferente e as sentenças têm ritmo diferenciado. Este trabalho visa identificar tais diferenças por meio das árvores probabilísticas de contexto.

Árvores probabilísticas de contextos foram introduzidos em Rissanen (1983) como uma generalização dos modelos de Markov. Nesses modelos, o comprimento da porção relevante do passado é função do próprio passado. A ideia é que para cada passado, somente uma porção dele, denominada contexto, é suficiente para predizer o próximo símbolo. Como nenhum contexto pode ser representado por um sufixo de outro contexto, é possível representar o conjunto de contextos por uma árvore probabilística. Esses modelos também são conhecidos na literatura por Cadeias de Ordem Variável.

Esses modelos são frequentemente usadas para descrever processos estocásticos de forma mais eficiente que um modelo de Markov. São muito mais flexíveis e econômicas, porque consideram as dependências estruturais do processo, incluindo memória somente onde necessário. Além disso, o número de parâmetros de uma Cadeia de Markov cresce exponencialmente conforme sua ordem aumenta, enquanto uma árvore probabilística de contextos pode reduzir de forma significativa o número de parâmetros do modelo.

Árvores probabilísticas de contextos também se estendem a processos não-Markovianos, isto é, quando a árvore de contextos é não limitada.

Rissanen (1983) também introduziu o Algoritmo Contexto para estimar a árvore de contextos de um processo estocástico. A decisão de podar a árvore de contextos está relacionada com uma função ganho que envolve a distância de Kullback-Leibler. Uma versão do Algoritmo Contexto com uma diferente função ganho foi introduzida em Galves e Leonardi

(2008), considerando as diferenças entre sucessivas probabilidades de transição empíricas. Uma abordagem diferente foi considerada em Csiszár e Talata (2006), que utilizou o *Bayesian Information Criterion* (BIC) e o *Minimum Description Length Principle* (MDL) na estimação de árvores de contextos.

Rissanen (1983) provou a consistência fraca do Algoritmo Contexto para o caso em que a árvore é limitada, quando é conhecida uma cota superior para seu tamanho. Bühlmann e Wyner (1999) provaram a consistência fraca do Algoritmo Contexto para o caso limitado fazendo a profundidade da árvore crescer de acordo com o tamanho da amostra. O caso em que a árvore é ilimitada foi considerado por Ferrari and Wyner (2003), que provaram a consistência fraca do Algoritmo Contexto nesse cenário.

Além da importância em Linguística, árvores probabilísticas de contextos são ferramentas importantes na Teoria da Informação, Bioinformática e Codificação Universal.

Neste trabalho, a teoria, metodologia e resultados foram organizados do seguinte modo: na Seção 2, apresentamos notações e conceitos básicos. Na mesma seção, descrevemos o Algoritmo Contexto, a versão do Algoritmo Contexto introduzida em Galves e Leonardi (2008) e o critério BIC. Enunciamos também teoremas que tratam da consistência desses estimadores. No final da seção, definimos o regime de contaminação Zero Inflado, que aplicamos posteriormente em simulações. Na Seção 3, apresentamos a metodologia deste trabalho. Os resultados do trabalho foram divididos em duas partes: uma parte computacional e outra aplicada com o objetivo de distinguir características rítmicas em textos de autores brasileiros e portugueses. Na parte computacional, comparamos o desempenho dos estimadores apresentados por meio de simulações. Na parte aplicada, codificamos textos brasileiros e portugueses em sequências de símbolos no alfabeto $\mathcal{A} = \{0, 1, 2, 3, 4\}$ e aplicamos o estimador BIC. Apresentamos os resultados e as discussões do trabalho na Seção 4. Por fim, reservamos a Seção 5 para as considerações finais do trabalho.

2 REVISÃO BIBLIOGRÁFICA

2.1 Notações e Conceitos Básicos

Considere um alfabeto $\mathcal{A} = \{0, 1, 2, \ldots, N-1\}$ finito com cardinalidade $|\mathcal{A}| = N$. Dados dois inteiros $m \leq n$, denotamos por a_m^n a sequência $a_m a_{m+1} \ldots a_n$ e definimos seu comprimento por $l(a_m^n) = n - m + 1$, em que $a_i \in \mathcal{A}$, $m \leq i \leq n$. A sequência vazia é denotada por $\emptyset$, com comprimento $l(\emptyset) = 0$. Se $n < m$, convencionamos que $a_m^n = \emptyset$. Denotamos por $a_{-\infty}^n$ a sequência semi-infinita $\ldots a_{n-1} a_n$ de simbolos em $\mathcal{A}$ com comprimento $l(a_{-\infty}^n) = +\infty$. $\mathcal{A}^j$ denota o conjunto de todas as sequências de tamanho j com símbolos em $\mathcal{A}$. Em particular, $\mathcal{A}^0$ contém somente a sequência vazia. Denotamos por $\mathcal{A}^* = \bigcup_{k=0}^{\infty} \mathcal{A}^k$ o conjunto de todas as sequências finitas com símbolos em $\mathcal{A}$ e por $\mathcal{A}^\infty$ o conjunto de todas as sequências semi-infinitas com símbolos em $\mathcal{A}$. Dadas duas sequências $u \in \mathcal{A}^* \cup \mathcal{A}^\infty$ e $v \in \mathcal{A}^*$, denotamos por uv a sequência obtida pela concatenação entre u e v. Por exemplo, se $u = u_1^m$ e $v = v_1^n$, com $m \geq 1$ e $n \geq 1$ inteiros, então a concatenação entre u e v é dada por $uv = u_1 \ldots u_m v_1 \ldots v_n$. Dizemos que a sequência $s \in \mathcal{A}^*$ é um *sufixo* da sequência $w \in \mathcal{A}^* \cup \mathcal{A}^\infty$ se existir alguma sequência $u \in \mathcal{A}^* \cup \mathcal{A}^\infty$, com $l(u) \geq 1$, tal que $w = us$. Neste caso, escrevemos $s \prec w$. Quando $s \prec w$ ou $s = w$, escrevemos $s \preceq w$. O maior sufixo de uma sequência $\omega \in \bigcup_{k=1}^{\infty} \mathcal{A}^k$ é denotado por $\text{suf}(\omega)$.

Um conjunto $\mathcal{T} \subset \mathcal{A}^* \cup \mathcal{A}^\infty$ de sequências é uma *árvore* se nenhum $s_1 \in \mathcal{T}$ for sufixo de algum $s_2 \in \mathcal{T}$. Essa propriedade é chamada *propriedade do sufixo*. Os elementos de $\mathcal{T}$ são chamados *folhas* de $\mathcal{T}$. Um *nó interno* é um sufixo de uma folha e os *nós* representam o conjunto de todos os nós internos e folhas. Os *descendentes* de um nó interno s são todas as sequências as, $a \in \mathcal{A}$, que são nós. Uma árvore $\mathcal{T}$ é *completa* se cada nó interno tem exatamente $|\mathcal{A}|$ descendentes e é *irredutível* se nenhum $s \in \mathcal{T}$ puder ser substituído por um sufixo de s sem violar a propriedade do sufixo.

Denotamos por $|\mathcal{T}|$ a cardinalidade de $\mathcal{T}$. A profundidade de uma árvore $\mathcal{T}$ é definida como

$$h(\mathcal{T}) = \max\{l(s), s \in \mathcal{T}\}.$$

Se $h(\mathcal{T}) < \infty$, dizemos que $\mathcal{T}$ é *limitada*. Caso contrario, dizemos que $\mathcal{T}$ é

ilimitada. Dado um inteiro K, $\mathcal{T}|_K$ denota a árvore $\mathcal{T}$ truncada em K, ou seja,

$$\mathcal{T}|_K = \{s \in \mathcal{T} : l(s) \leq K\} \cup \{s \in \mathcal{A}^k : s \prec s' \text{ para algum } s' \in \mathcal{T}\}.$$

Exemplo 2.1 *Considere* $\mathcal{A} = \{0, 1, 2\}$ *e a árvore* $\mathcal{T} = \{000, 100, 10, 1, 2\}$. *Representamos graficamente a árvore* $\mathcal{T}$ *na Figura 1. O conjunto* $\mathcal{T}$ *satisfaz a propriedade do sufixo, portanto é de fato uma árvore. Em* $\mathcal{T}$, *as sequências* $000, 100, 10, 1$ *e* 2 *são folhas,* $00, 0$ *e* $\emptyset$ *são nós internos,* $000, 100, 00, 10, 0, 1, 2$ *e* $\emptyset$ *são nós. O nó interno* 00 *possui apenas os descendentes* 000 *e* 100, *logo a árvore* $\mathcal{T}$ *não é completa. A árvore* $\mathcal{T}$ *truncada em 2 é dada por* $\mathcal{T}|_2 = \{00, 10, 1, 2\}$. *Além disso, a árvore* $\mathcal{T}$ *é limitada e irredutível.*

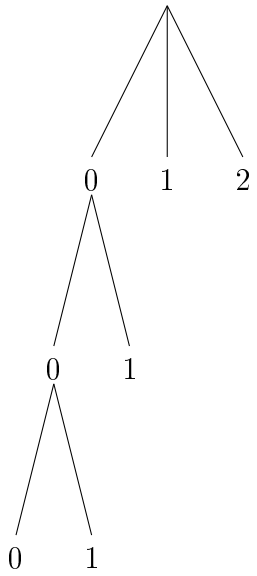

Figura 1 – Exemplo de árvore

Considere um processo estacionário e ergódico $\mathbf{X} = \{X_t : t \in \mathbb{Z}\}$ em um alfabeto finito $\mathcal{A}$. Dada uma sequência $s \in \mathcal{A}^*$, escrevemos

$$\mu_X(s) = \begin{cases} \mathbb{P}(X_1^{l(s)} = s), & \text{se } s \neq \emptyset, \\ 0, & \text{se } s = \emptyset. \end{cases}$$

Dadas as sequências $s \in \mathcal{A}^* \cup \mathcal{A}^\infty$ e $a \in \mathcal{A}$, escrevemos

$$p_X(a|s) = \begin{cases} \mathbb{P}(X_0 = a | X_{-l(s)}^{-1} = s), & \text{se } s \neq \emptyset, \\ \mu_X(a), & \text{se } s = \emptyset. \end{cases}$$

A seguir, definimos contexto.

Definição 2.1 *Uma sequência $s \in \mathcal{A}^k$ é um contexto finito para o processo* $\mathbf{X}$ *se satisfaz*

(1) $\mu_X(s) > 0$*;*

(2) Para toda sequência semi-infinita $x_{-\infty}^{-1}$ *que tem* s *como sufixo*

$$\mathbb{P}(X_0 = a | X_{-\infty}^{-1} = x_{-\infty}^{-1}) = p_X(a|s), \text{ para todo } a \in \mathcal{A};$$

(3) Nenhum sufixo de s *satisfaz (2).*

Um contexto infinito é uma sequência semi-infinita $\omega = x_{-\infty}^{-1}$ *cujos sufixos* x_{-k}^{-1}, $k = 1, 2, \ldots$ *tem probabilidade positiva e nenhum deles é um contexto finito.*

O conjunto de todos os contextos de um processo $\mathbf{X}$ é uma árvore irredutível. Essa árvore será chamada *árvore de contextos de* $\mathbf{X}$. As duas definições a seguir tratam da classe de modelos que utilizamos neste trabalho. Definimos o conceito de árvore probabilística de contextos e quando um processo $\mathbf{X}$ é compatível com esse modelo.

Definição 2.2 *Uma árvore probabilística de contextos em* $\mathcal{A}$ *é um par ordenado* $(\mathcal{T}, \overline{p})$ *que satisfaz*

(1) $\mathcal{T}$ *é uma árvore irredutível;*

(2) $\overline{p} = \{\overline{p}(.|s), s \in \mathcal{T}\}$ *é uma família de probabilidades de transição sobre* $\mathcal{A}$*.*

Definição 2.3 *Dizemos que o processo* $\mathbf{X}$ *é compatível com a árvore probabilística de contextos* $(\mathcal{T}, \overline{p})$*, se satisfaz*

(1) $\mathcal{T}$ *é a árvore de contextos do processo* $\mathbf{X}$*;*

(2) Para qualquer $s \in \mathcal{T}$ *e* $a \in \mathcal{A}$, $p_X(a|s) = \overline{p}(a|s)$*.*

Denotamos por $\mathcal{T}_X$ a árvore de contextos do processo $\mathbf{X}$. Se $\mathcal{T}_X$ tem profundidade $h(\mathcal{T}_X) = k < \infty$, então o processo $\mathbf{X}$ é uma cadeia de Markov de ordem k. Nesse caso, a árvore probabilística de contextos representa uma descrição mais econômica do processo, possuindo $(|\mathcal{A}| - 1)|\mathcal{T}_X|$ parâmetros em vez de $(|\mathcal{A}| - 1)|\mathcal{A}|^k$ parâmetros necessários para estimar uma cadeia de Markov de ordem k.

Seja $X_1, X_2, \ldots, X_n$ uma amostra aleatória do processo $\mathbf{X}$. Dado um inteiro positivo d, $N_n(s, a)$ denota o número de ocorrências da sequência $s \in \bigcup_{j=0}^{d} \mathcal{A}^j$ seguida pelo símbolo $a \in \mathcal{A}$, ou seja,

$$N_n(s, a) = \sum_{i=d+1}^{n} \mathbb{I}(X_{i-l(s)}^{i-1} = s, X_i = a).$$

Seja $N_n(s)$ o número de ocorrências de s, ou seja,

$$N_n(s) = \sum_{i=d+1}^{n} \mathbb{I}(X_{i-l(s)}^{i-1} = s).$$

Podemos ver que $N_n(s) = \sum_{a \in A} N_n(s,a)$. Se $s = \emptyset$, convencionamos $N_n(s,a) = \sum_{i=d+1}^{n} \mathbb{I}(X_i = a)$ e $N_n(s) = n - d$.

O conjunto de todas as sequências $s \in \bigcup_{j=0}^{d} \mathcal{A}^j$ que aparecem ao menos uma vez na amostra será denotado por $\mathcal{V}_n$, isto é,

$$\mathcal{V}_n = \left\{ s \in \bigcup_{j=0}^{d} \mathcal{A}^j : N_n(s) \geq 1 \right\}.$$

A seguir, definimos quando uma árvore é factível. O objetivo dessa definição é, a partir da amostra, fazer uma seleção inicial das possíveis árvores de contexto de $\mathbf{X}$.

Definição 2.4 *Uma árvore $\mathcal{T}$ é factível se satisfaz*
(1) $s \in \mathcal{V}_n$ para todo $s \in \mathcal{T}$;
(2) Cada sequência $s' \in \mathcal{V}_n$ é tal que $s' \preceq s$ ou $s \prec s'$ para algum $s \in \mathcal{T}$.

O conjunto de todas as árvores factíveis será denotado por $\mathcal{F}_n$. O objetivo é estimar a árvore de contextos $\mathcal{T}_X$ a partir de uma amostra de $\mathbf{X}$. Para tanto, devemos escolher uma árvore factível que se aproxime de $\mathcal{T}_X$. Se $h(\mathcal{T}_X) < \infty$, então devemos escolher d de modo que $h(\mathcal{T}_X) \leq d$ para que exista uma árvore factível que coincida com $\mathcal{T}_X$. Para estimar $\mathcal{T}_X$, não é necessário o conhecimento prévio da sua profundidade, portanto d pode ser uma função crescente de n.

Dada uma árvore $\mathcal{T}_X \subset \bigcup_{j=0}^{d} \mathcal{A}^j$ de um processo $\mathbf{X}$, a probabilidade de ocorrer x_1^n pode ser escrita como

$$\mathbb{P}(X_1^n = x_1^n) = \mathbb{P}(X_1^d = x_1^d) \prod_{s \in \mathcal{T}} \prod_{a \in \mathcal{A}} p_X(a|s)^{N_n(s,a)}.$$

Assumiremos $\mathbb{P}(X_1^d = x_1^d) = 1$. A máxima verossimilhança da amostra x_1^n é dada por

$$\mathrm{ML}_{\mathcal{T}}(x_1^n) = \prod_{s \in \mathcal{T}} \prod_{a \in \mathcal{A}} \widehat{p}_n(a|s)^{N_n(s,a)},$$

em que as probabilidades empíricas $\widehat{p}_n(a|s)$ são dadas por

$$\widehat{p}_n(a|s) = \begin{cases} \frac{N_n(s,a)}{N_n(s)}, & \text{se } N_n(s) > 0, \\ \frac{1}{|\mathcal{A}|}, & \text{se } N_n(s) = 0. \end{cases}$$

Para uma sequência $s \in \bigcup_{j=0}^{d} \mathcal{A}^j$, seja

$$\mathrm{ML}_s(x_1^n) = \prod_{a \in \mathcal{A}} \widehat{p}_n(a|s)^{N_n(s,a)}.$$

Portanto,

$$\mathrm{ML}_{\mathcal{T}}(x_1^n) = \prod_{s \in \mathcal{T}} \mathrm{ML}_s(x_1^n).$$

2.2 O Algoritmo Contexto

O Algoritmo Contexto proposto por Rissanen (1983) baseia-se, inicialmente, na maior árvore factível. Em seguida, mede a discrepância entre as probabilidades empíricas de transição do maior sufixo de um contexto e seus descendentes. Se a discrepância for maior que um dado valor limiar, o contexto será mantido; caso contrário, será podado. O procedimento continua até que não seja mais possível podar a árvore.

Para medir discrepância entre duas medidas de probabilidade em $\mathcal{A}$, usamos a *divergência de Kullback-Leibler*, definida abaixo.

Definição 2.5 *A divergência de Kullback-Leibler, definida para medidas de probabilidade P e Q em $\mathcal{A}$, é dada por*

$$D(P;Q) = \sum_{a \in \mathcal{A}} P(a) \log \frac{P(a)}{Q(a)}.$$

Por convenção, $P(a) \log \frac{P(a)}{Q(a)} = 0$, se $P(a) = 0$, e $P(a) \log \frac{P(a)}{Q(a)} = +\infty$, se $P(a) > Q(a) = 0$.

Para uma sequência $s \in \mathcal{V}_n$, seja

$$\Lambda_n(s) = \sum_{b \in \mathcal{A}: bs \in \mathcal{V}_n} N_n(bs) D(\widehat{p}_n(.|bs); \widehat{p}_n(.|s)).$$

Denotamos o limiar utilizado no Algoritmo Contexto por δ_n, em que $(\delta_n)_{n \in \mathbb{N}}$ é uma sequência de números reais de modo que $\delta_n \to \infty$ e $\delta_n/n \to 0$ quando $n \to \infty$. Podemos descrever o Algoritmo Contexto em três passos:

Passo 1: Seja $X_1, X_2, \ldots, X_n$ uma amostra aleatória do processo $\mathbf{X}$ e considere $\widehat{\mathcal{T}}_0$ a maior árvore factível obtida a partir dessa amostra, isto é, a árvore $\widehat{\mathcal{T}}_0$ que possui a maior cardinalidade $|\widehat{\mathcal{T}}_0|$.

Passo 2: Seja $S = \{\text{suf}(\omega) : \omega \in \widehat{\mathcal{T}}_0\}$. Aplique a função $\Lambda_n(s)$ para todo $s \in S$, com $N_n(s) \geq 2$, tal que não exista $s' \in S$ e $s \prec s'$. Se $\Lambda_n(s) < \delta_n$, substitua por s todos os elementos $as \in \widehat{\mathcal{T}}_0$, com $a \in \mathcal{A}$. Denotamos por $\widehat{\mathcal{T}}_1$ a árvore obtida após o procedimento.

Passo 3: Repita o Passo 2 para $\widehat{\mathcal{T}}_i$ em vez de $\widehat{\mathcal{T}}_{i-1}$ $(i = 1, 2, \ldots)$ até que a árvore obtida após o procedimento seja idêntica à inicial. Denotamos essa árvore por $\widehat{\mathcal{T}}_C(X_1^n)$.

Para uma amostra X_1^n, podemos também obter $\widehat{\mathcal{T}}_C(X_1^n)$ a partir da função $C_s(X_1^n)$, definida para todo $s \in \mathcal{V}_n$ e dada por

$$C_s(X_1^n) = \begin{cases} 0, & \text{se } N_n(s) \leq 1 \text{ ou } l(s) = d, \\ \max\{\mathbb{I}(\Lambda_n(s) \geq \delta_n), \max\limits_{b \in \mathcal{A}} C_{bs}(X_1^n)\}, & \text{se } N_n(s) > 1 \text{ e } l(s) < d. \end{cases}$$

Definição 2.6 *O estimador $\widehat{\mathcal{T}}_C(X_1^n)$ da árvore de contextos de $\mathbf{X}$ é o conjunto dado por*

$$\widehat{\mathcal{T}}_C(X_1^n) = \{s \in \mathcal{V}_n : C_s(X_1^n) = 0 \text{ e } C_{s'}(X_1^n) = 1 \text{ para todo } s' \prec s\}.$$

2.3 Uma Versão Modificada do Algoritmo Contexto

Uma modificação do Algoritmo Contexto foi proposta em Galves e Leonardi (2008). Nesta variante, a decisão de poda é tomada a partir das diferenças entre probabilidades condicionais empíricas. Seja $\Delta_n(s)$ o operador

$$\Delta_n(s) = \max_{a \in \mathcal{A}} |\widehat{p}_n(a|s) - \widehat{p}_n(a|\text{suf}(s))|, \quad \forall\ s \in \bigcup_{k=1}^{d} \mathcal{A}^k.$$

Definição 2.7 *Para quaisquer $\delta > 0$ e $d < n$, o estimador da árvore de contextos $\widehat{\mathcal{T}}_n^{\delta,d}$ é o conjunto de todas as sequências $s \in \bigcup\limits_{k=1}^{d} \mathcal{A}^k$, tais que $\Delta_n(a\text{suf}(s)) > \delta$, para algum $a \in \mathcal{A}$, e $\Delta_n(us) \leq \delta$, para todo $u \in \bigcup\limits_{k=1}^{d-l(s)} \mathcal{A}^k$. Se $\widehat{\mathcal{T}}_n^{\delta,d} = \varnothing$, consideraremos $\widehat{\mathcal{T}}_n^{\delta,d} = \{\emptyset\}$. No caso $l(s) = d$, definimos $\bigcup\limits_{k=1}^{0} \mathcal{A}^k = \varnothing$.*

Para que a versão modificada do Algoritmo Contexto seja fortemente consistente, vamos supor que a árvore probabilística de contextos $(\mathcal{T}, p)$ satisfaz as condições das duas definições a seguir.

Definição 2.8 *Dizemos que uma árvore probabilística de contextos $(\mathcal{T}, p)$ é fortemente não-nula, se satisfaz*

$$\inf_{a\in\mathcal{A},s\in\mathcal{T}}\{p(a|s)\} > 0.$$

Definição 2.9 *Seja a sequência $\{\alpha_k\}_{k\geq 0}$, definida por*

$$\alpha_0 = \sum_{a\in\mathcal{A}} \inf_{s\in\mathcal{T}}\{p(a|s)\},$$

$$\alpha_k = \inf_{u\in\mathcal{A}^k} \sum_{a\in\mathcal{A}} \inf_{s\in\mathcal{T},u\prec s}\{p(a|s)\}.$$

Dizemos que uma árvore probabilística de contextos $(\mathcal{T}, p)$ é somável se possui sequência $\{\beta_k\}_{k\geq 0}$, $\beta_k = 1 - \alpha_k$, tal que

$$\beta = \sum_{k\in\mathbb{N}} \beta_k < \infty.$$

Também é necessário restringir os valores de δ e d para que o estimador seja fortemente consistente. Com base nisso, dado um inteiro $k \geq 1$ e uma árvore probabilística de contextos $(\mathcal{T}, p)$, definimos

$$\mathcal{C}_k = \{u \in \mathcal{T}|_k : p(a|u) \neq p(a|\text{suf}(u)), \text{ para algum } a \in \mathcal{A}\},$$

$$D_k = \min_{u\in\mathcal{C}_k} \max_{a\in\mathcal{A}}\{|p(a|u) - p(a|\text{suf}(u))|\}.$$

Note que $D_k > 0$ para todo $k \geq 1$. Podemos enunciar agora o teorema que trata da convergência forte da versão modificada do Algoritmo Contexto introduzida em Galves e Leonardi (2008).

Teorema 2.1 *Seja $X_1, \ldots, X_n$ uma amostra aleatória de um processo $\mathbf{X} = \{X_t : t \in \mathbb{Z}\}$ estacionário e ergódico, compatível com a árvore probabilística de contextos $(\mathcal{T}, p)$ fortemente não-nula e somável. Então, para qualquer inteiro K, $0 < \delta < D_d$ e d satisfazendo*

$$d > \max_{u\notin\mathcal{T},l(u)\leq K} \min\{k : \exists\ s \in \mathcal{C}_k, u \prec s\},$$

temos

$$\widehat{\mathcal{T}}_n^{\delta,d}|_K = \mathcal{T}|_K$$

quase certamente quando $n \to \infty$.

Observação: Para que o estimador seja consistente, é necessário tomar δ pequeno o suficiente. Isso significa que o estimador $\widehat{\mathcal{T}}_n^{\delta,d}$ não é universal, porque os valores que δ pode assumir dependem do processo $\mathbf{X}$ que deu origem à amostra.

2.4 O Critério BIC

Seja $X_1, X_2, \ldots, X_n$ uma amostra do processo $\mathbf{X}$. A seleção de uma árvore factível $\mathcal{T}_0 \subset \mathcal{F}_n$ que estime $\mathcal{T}_X$ deve considerar os seguintes aspectos: a função de verossimilhança da amostra e a complexidade da árvore. O objetivo é escolher $\mathcal{T}_0$ de modo que a função de verossimilhança da amostra seja comparativamente alta, com preferência por modelos menos complexos. Definimos o Critério de Informação Bayesiana (BIC) a seguir.

Definição 2.10 *Dada uma amostra X_1^n, o Critério de Informação Bayesiana (BIC) para uma árvore factível $\mathcal{T}$ é definida como*

$$\mathrm{BIC}_{\mathcal{T}}(X_1^n) = -\log \mathrm{ML}_{\mathcal{T}}(X_1^n) + c|\mathcal{T}| \log n,$$

em que c é uma constante real positiva.

Uma característica do BIC é a constante de penalização c. Em Csiszár e Talata (2006), foi escolhida a constante $c = (|A| - 1)/2$. Desse modo, $c|\mathcal{T}|$ representa metade do número de parâmetros livres do modelo quando a árvore $\mathcal{T}$ é completa.

Definição 2.11 *O estimador BIC é dado por*

$$\widehat{\mathcal{T}}_{BIC}(X_1^n) = \arg\min_{\mathcal{T} \in \mathcal{F}_n} \mathrm{BIC}_{\mathcal{T}}(X_1^n).$$

Na prática, é inviável estimar a árvore de contextos calculando o critério BIC para cada árvore factível. Entretanto, Csiszár e Talata (2006) obtiveram um algoritmo eficiente para encontrar $\widehat{\mathcal{T}}_{BIC}(X_1^n)$. Para a conveniência do leitor, iremos descrevê-lo a seguir. Definimos, recursivamente, para todo $s \in \mathcal{V}_n$, a função

$$V_s(x_1^n) = \begin{cases} \max\{n^{-c}\ \mathrm{ML}_s(x_1^n), \prod_{b \in \mathcal{A}: bs \in \mathcal{V}_n} V_{bs}(x_1^n)\}, & \text{se } l(s) < d, \\ n^{-c}\ \mathrm{ML}_s(x_1^n), & \text{se } l(s) = d. \end{cases}$$

e a indicadora

$$\mathcal{X}_s(x_1^n) = \begin{cases} \mathbb{I}\left\{ \prod_{b \in \mathcal{A}: bs \in \mathcal{V}_n} V_{bs}(x_1^n) > n^{-c}\ \mathrm{ML}_s(x_1^n) \right\}, & \text{se } l(s) < d, \\ 0, & \text{se } l(s) = d. \end{cases}$$

Convencionamos que se $\{b \in \mathcal{A} : bs \in \mathcal{V}_n\} = \varnothing$, então $V_s(x_1^n) = n^{-c}\ \mathrm{ML}_s(x_1^n)$ e $\mathcal{X}_s(x_1^n) = 0$. Csiszár e Talata (2006) demonstraram que

$$\widehat{\mathcal{T}}_{BIC}(X_1^n) = \{s \in \mathcal{V}_n : \mathcal{X}_s(X_1^n) = 0 \text{ e } \mathcal{X}_{s'}(X_1^n) = 1 \text{ para todo } s' \prec s\}.$$

Csiszár e Talata (2006) também provaram a consistência forte do estimador BIC para árvores de contexto finitas e infinitas. Enunciamos abaixo esse resultado.

Teorema 2.2 *Csiszár e Talata (2006). Seja $X_1, \ldots, X_n$ uma amostra aleatória de um processo $\mathbf{X} = \{X_t : t \in \mathbb{Z}\}$ ergódico e estacionário, compatível com uma árvore probabilística de contextos $(\mathcal{T}, p)$. No caso em que $h(\mathcal{T}) < \infty$, o estimador BIC, com $d = o(log(n))$, satisfaz*

$$\widehat{\mathcal{T}}_{BIC}(X_1^n) = \mathcal{T}$$

quase certamente quando $n \to \infty$. No caso ilimitado, para qualquer K natural, o estimador BIC satisfaz

$$\widehat{\mathcal{T}}_{BIC}(X_1^n)\,|_K = \mathcal{T}\,|_K$$

quase certamente quando $n \to \infty$.

2.5 O Modelo de Contaminação Zero Inflado

Neste trabalho vamos verificar o comportamento dos estimadores em amostras contaminadas. Mais precisamente, o modelo de contaminação utilizado é o regime de contaminação Zero Inflado apresentado em Garcia e Moreira (2015). A ideia é que uma amostra de um processo contaminado tenha mais zeros do que teria sem a contaminação.

Definição 2.12 *Consideramos o processo $\mathbf{X} = \{X_t : t \in \mathbb{Z}\}$ estacionário e ergódico tomando valores no alfabeto $\mathcal{A} = \{0, 1\}$. Seja $\boldsymbol{\xi} = \{\xi_t : t \in \mathbb{Z}\}$ uma sequência de variáveis aleatórias i.i.d. com distribuição Bernoulli, independente do processo $\mathbf{X}$, com*

$$\mathbb{P}(\xi_t = 1) = 1 - \varepsilon,$$

em que ε é um parâmetro de perturbação fixado em $(0, 1)$. O modelo de contaminação Zero Inflado é dado por

$$Z_t = X_t \cdot \xi_t, \quad t \in \mathbb{Z}.$$

Podemos ver, pela Definição 2.12, que o processo $\mathbf{Z}$ terá uma quantidade de zeros maior ou igual ao processo $\mathbf{X}$. Quanto maior for ε, maior é a probabilidade de obter $Z_t = 0$, dado que $X_t = 1$, $t \in \mathbb{Z}$.

Pode-se mostrar que processo $\mathbf{Z}$ obtido pelo modelo de contaminação Zero Inflado possui árvore de contextos ilimitada mesmo se $\mathbf{X}$ for uma cadeia de Markov de ordem um.

3 METODOLOGIA

O desenvolvimento do trabalho foi realizado dentro do cenário da Teoria de Processos Estocásticos e teve um enfoque de Probabilidade Clássica.

Inicialmente, foram realizadas simulações de processos estocásticos com árvores de contextos e probabilidades de transição fixadas. Foram estudados o Algoritmo Contexto, a versão do Algoritmo Contexto apresentada em Galves e Leonardi (2008) e o algoritmo do estimador BIC introduzido em Csiszár and Talata (2006). Para cada estimador, em um cenário fixado, foram realizadas 100 simulações e foi informada a proporção de identificações corretas da árvore de contextos. Também foi aplicado em um caso o modelo de contaminação Zero Inflado apresentado por Garcia e Moreira (2015). As simulações foram realizadas através do software estatístico R.

Em seguida, foi utilizado um banco de dados composto por textos literários e não-literários extraídos de autores brasileiros e portugueses. Cada sílaba de um texto selecionado foi codificada em dois símbolos de acordo com as condições seguintes:

1. se a sílaba é tônica ou não;

2. se é início de palavra prosódica ou não,

em que palavra prosódica é definida como sendo uma palavra léxica juntamente com as palavras não acentuadas que a precedem, ver Vigario (2003).

Cada sílaba foi identificada com um dos pares ordenados $(0,0),(1,0),(0,1)$ e $(1,1)$, em que o primeiro símbolo indica se for início de palavra prosódica e o segundo indica se a sílaba for tônica. Para simplificar a notação, representamos os pares ordenados por inteiros, identificando $(0,0)=0$, $(0,1)=1$, $(1,0)=2$ e $(1,1)=3$. Adicionalmente, atribuímos ao fim do período o símbolo 4.

Exemplo 3.1 *Usaremos a sentença "O menino já comeu o doce." para uma maior compreensão da codificação.*

Sentença	*O*	*me*	*ni*	*no*	*já*	*co*	*meu*	*o*	*do*	*ce*	.
Início de palavra prosódica	*1*	*0*	*0*	*0*	*1*	*1*	*0*	*1*	*0*	*0*	
Sílaba tônica	*0*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	
Codificação	*2*	*0*	*1*	*0*	*3*	*2*	*1*	*2*	*1*	*0*	*4*

Portanto, cada texto foi convertido em uma sequência de inteiros tomando valores no alfabeto $\mathcal{A} = \{0, 1, 2, 3, 4\}$. O programa "silaba2008.pl" foi utilizado para efetuar a conversão e está incluso na pasta "SCRIPTS" do material suplementar anexo a Galves et al. (2012).

Foram selecionados 26 textos brasileiros e 26 textos portugueses, literários e não-literários. Só foram considerados textos que, após a conversão, resultaram em sequências de tamanho maior que cinco mil. Os textos estão disponíveis em `http://www.ime.usp.br/~tycho/prosody/vlmc/data/arquivo/`.

A classe de modelos que consideramos para modelar os dados linguísticos foram as árvores probabilísticas de contextos. Utilizamos o estimador BIC com a constante de penalização $c = 0,2$ para estimar as árvores de contextos associadas a textos brasileiros e portugueses. Os modelos encontrados foram utilizados como base para detectar diferenças rítmicas entre as duas línguas.

4 RESULTADOS E DISCUSSÃO

4.1 Simulações

4.1.1 Resultados

A seguir, denotaremos por E_{BIC} o estimador BIC que utiliza a constante de penalização $c = (|A| - 1)/2$. O Algoritmo Contexto e a versão modificada do Algoritmo Contexto proposta em Galves e Leonardi (2008) serão chamados E_{C1} e E_{C2}, respectivamente. Para o Algoritmo Contexto, consideraremos o limiar $\delta_n = C\log(n)$, em que C é uma constante positiva. A constante C de E_{C1} e o limiar δ de E_{C2} serão fixados. Vamos supor também que seja conhecida uma cota superior $d = 4$ para o comprimento da árvore de contextos em cada modelo.

Exemplo 4.1 *Neste exemplo, representamos uma árvore probabilística de contextos em que a árvore de contextos e as probabilidades de transição associadas a esses contextos são dadas pela Tabela 1. Foram utilizados os estimadores E_{BIC}, E_{C2}, com constantes $C = 0,5$ e $C = 1$, e E_{C3}, com limiares $\delta = 0,15$ e $\delta = 0,10$. O desempenho dos estimadores para o Exemplo 4.1 é apresentado na Tabela 2. Foram simuladas amostras de tamanhos 200, 300, 500, 1.000, 2.000 e 5.000.*

Tabela 1 – Contextos e probabilidades de transição associadas em $\mathcal{A} = \{0, 1\}$ para o Exemplo 4.1.

Contextos (s)	$p(0\|s)$
0	0,3
01	0,5
11	0,8

Exemplo 4.2 *Este exemplo é definido como o Exemplo 4.1 em um regime de contaminação Zero Inflado com $\varepsilon = 0,15$. Desejamos obter a árvore de contextos do processo antes da contaminação. Novamente, foram utilizados os estimadores E_{BIC}, E_{C2}, com constantes $C = 0,5$ e $C = 1$, e E_{C3}, com limiares $\delta = 0,15$ e $\delta = 0,10$. O desempenho dos estimadores para*

o Exemplo 4.2 é dado pela Tabela 3. Foram utilizadas amostras de tamanhos 10.000, 20.000, 50.000 e 100.000.

Tabela 2 – Proporção de identificações corretas da árvore de contextos do Exemplo 4.1 para 100 amostras simuladas de tamanhos 200, 300, 500, 1.000, 2.000 e 5.000.

Estimador	Tamanho da amostra (n)					
	200	300	500	1.000	2.000	5.000
E_{BIC}	0,46	0,70	0,94	1,00	1,00	1,00
E_{C1} $(C = 1)$	0,47	0,72	0,90	0,99	0,99	1,00
E_{C1} $(C = 0,5)$	0,60	0,74	0,81	0,92	0,94	0,94
E_{C2} $(\delta = 0,15)$	0,02	0,03	0,10	0,46	0,84	1,00
E_{C2} $(\delta = 0,10)$	0,00	0,00	0,02	0,22	0,41	0,88

Tabela 3 – Proporção de identificações corretas da árvore de contextos original do Exemplo 4.2 para 100 amostras simuladas de tamanhos 10.000, 20.000, 50.000 e 100.000.

Estimador	Tamanho da amostra (n)			
	10.000	20.000	50.000	100.000
E_{BIC}	0,99	0,97	0,68	0,12
E_{C1} $(C = 1)$	0,97	0,89	0,41	0,00
E_{C1} $(C = 0,5)$	0,70	0,43	0,01	0,00
E_{C2} $(\delta = 0,15)$	0,78	0,87	0,98	0,99
E_{C2} $(\delta = 0,10)$	0,96	1,00	1,00	1,00

4.1.2 Discussão

No exemplo 4.1, os estimadores E_{BIC} e E_{C1} $(C = 1)$ tiveram um desempenho melhor que os demais e identificaram corretamente todas as árvores de contexto a partir de $n = 1.000$. O estimador E_{C1} $(C = 0,5)$ teve um bom desempenho com amostras pequenas, mas não melhorou muito à medida que a amostra aumentou. Os estimadores E_{C2} $(\delta = 0,15)$ e E_{C2} $(\delta = 0,10)$ tiveram um péssimo desempenho com amostras pequenas, mas melhorou nas amostras maiores.

O bom desempenho inicial do estimador E_{C1} $(C = 0,5)$ se deve ao fato de esse estimador retornar árvores maiores que o E_{C1} $(C = 1)$, o que é vantajoso para amostras pequenas, mas revelou ser pouco eficiente para amostras maiores.

Sob algumas condições, a versão modificada do Algoritmo Contexto proposta em Galves e Leonardi (2008) recupera a árvore de contextos a partir de uma amostra contaminada do processo. Garcia e Moreira (2015) demostraram que esse estimador é fortemente consistente para um processo seguindo o regime de contaminação Zero Inflado. O mesmo não ocorre com o Algoritmo Contexto e o estimador BIC, que tenderão a estimar a árvore do processo contaminado à medida que a amostra aumenta.

Os resultados das simulações confirmam que o estimador BIC e o Algoritmo Contexto não recuperam a árvore original do processo contaminado. Conforme a amostra aumenta, menor é a proporção de identificações corretas para esses estimadores. Porém, a versão modificada do Algoritmo Contexto proposta em Galves e Leonardi (2008) apresentou melhora na identificação da árvore original conforme a amostra aumentou, indicando que o estimador é consistente nesse cenário.

É importante notar que a contaminação com $\varepsilon = 0,15$ é bastante elevada. Garcia e Moreira (2015) demonstraram a consistência forte da versão modificada do Algoritmo Contexto para contaminações bem menores que a proposta no Exemplo 4.2. Esse exemplo mostra que, mesmo em processos bastante contaminados, há a possibilidade de aplicar a versão modificada do Algoritmo Contexto.

O Exemplo 4.1 mostra que, se não houver suspeita de contaminação, os estimadores mais indicados são o Algoritmo Contexto e o estimador BIC. A versão modificada do Algoritmo Contexto tem a vantagem de recuperar a árvore de contextos de processos

contaminados, mas não é eficiente. Sua utilização somente deve ser considerada quando há suspeita de contaminação.

4.2 Aplicações em Linguística

4.2.1 Resultados

A seguir, apresentamos as árvores de contexto obtidas de 26 textos brasileiros e 26 textos portugueses. Foram obtidas 9 árvores distintas, dadas abaixo. Para efeito de notação, iremos denotar as árvores obtidas por Árvore 1, Árvore 2, e assim por diante. A Tabela 4 informa a frequência das árvores obtidas para cada língua.

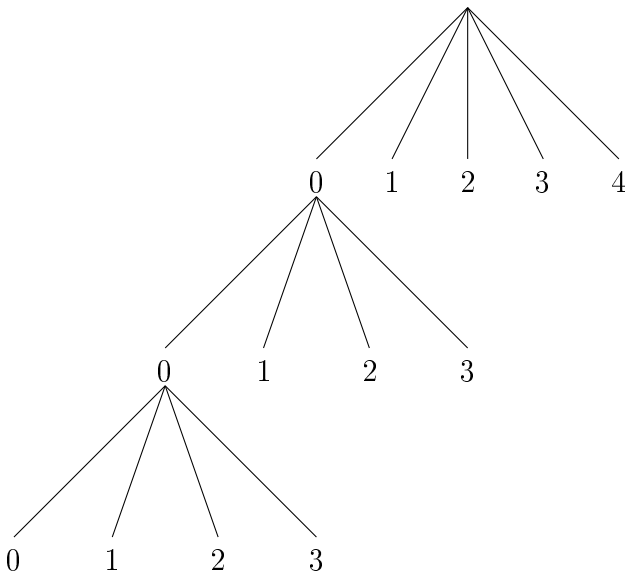

Figura 2 – Árvore 1

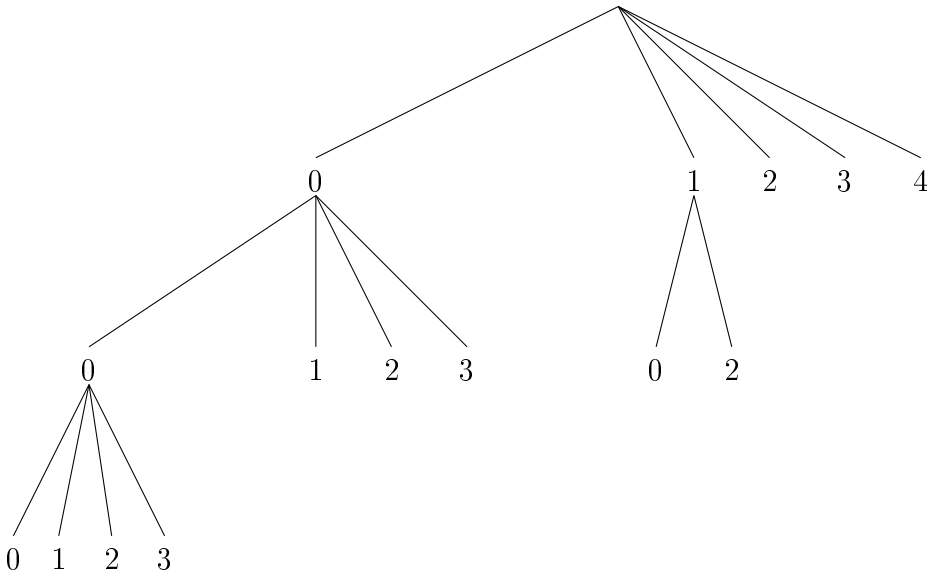

Figura 3 – Árvore 2

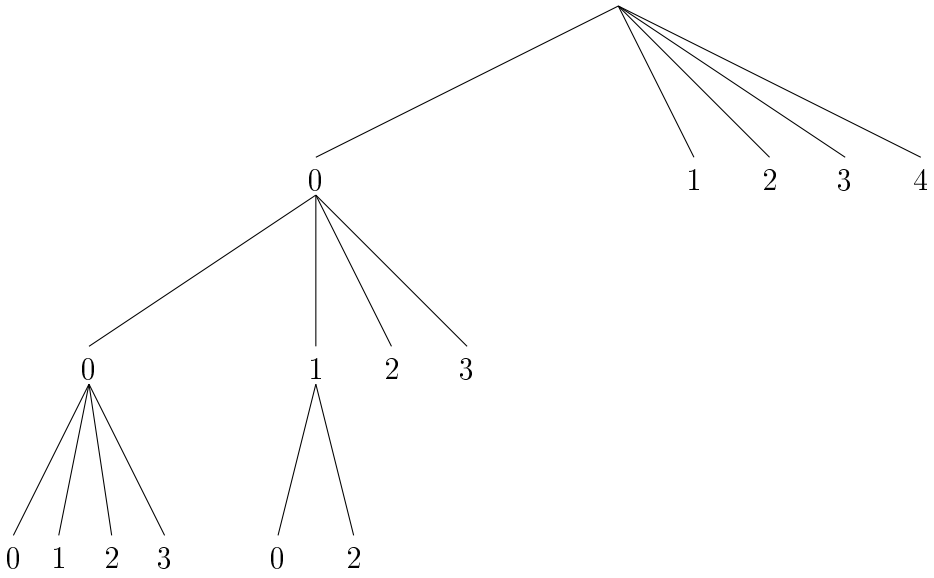

Figura 4 – Árvore 3

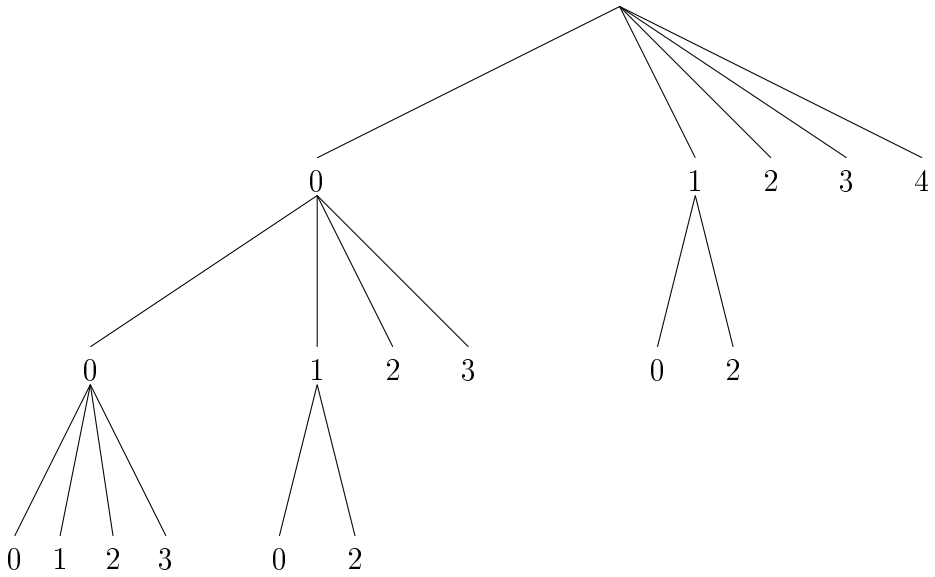

Figura 5 – Árvore 4

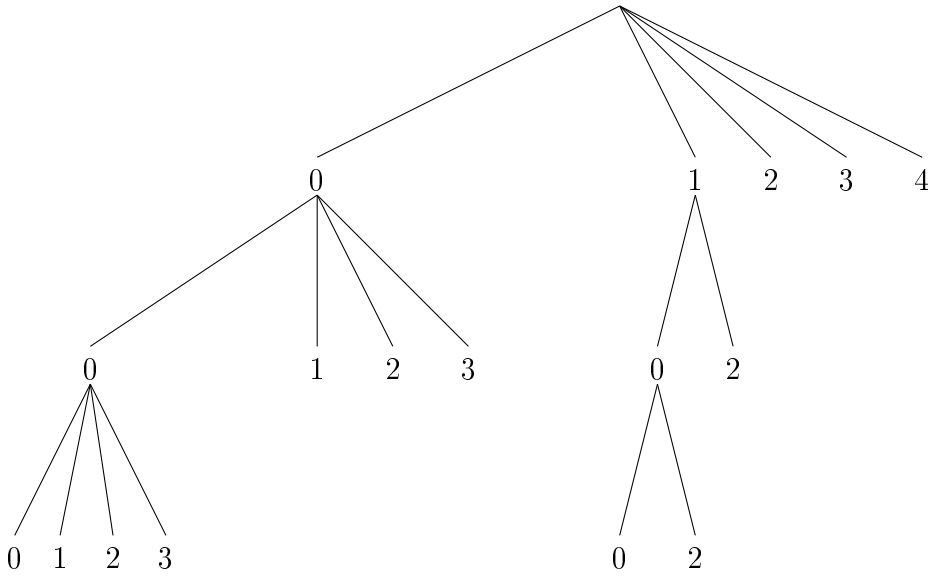

Figura 6 – Árvore 5

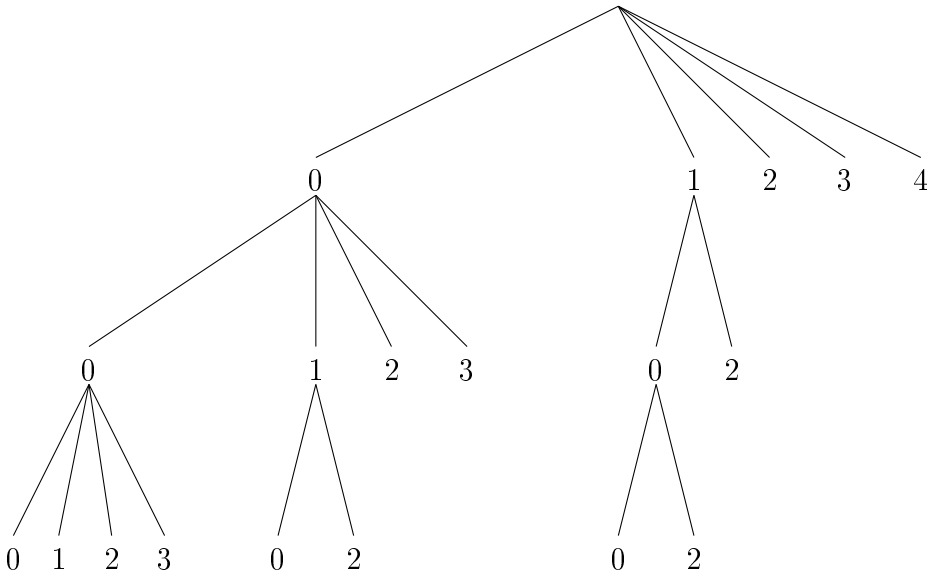

Figura 7 – Árvore 6

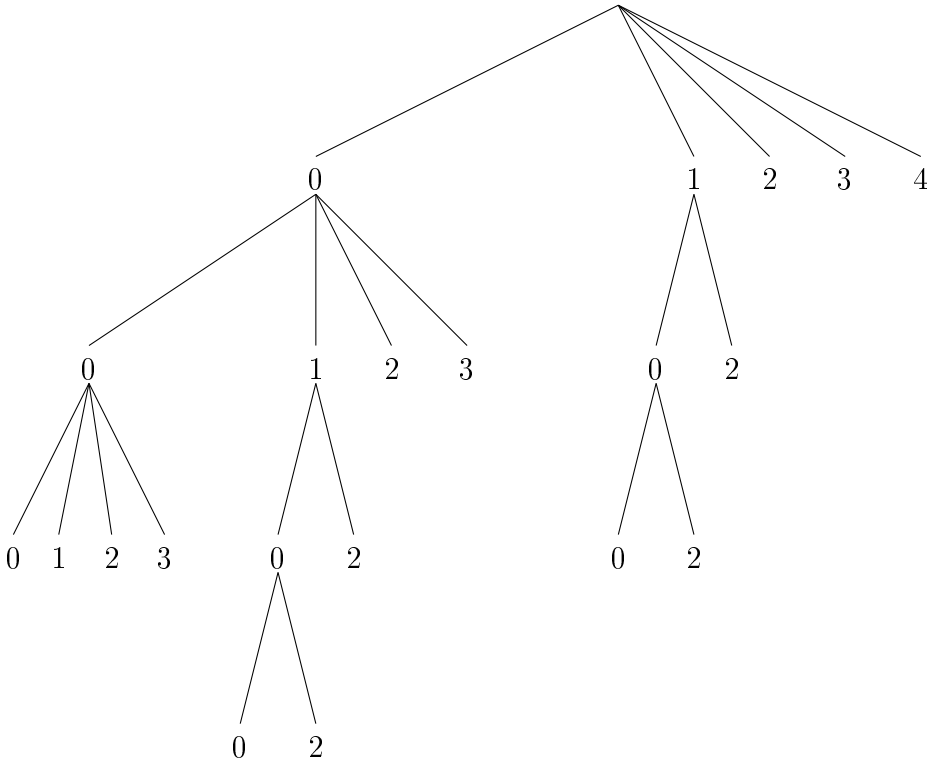

Figura 8 – Árvore 7

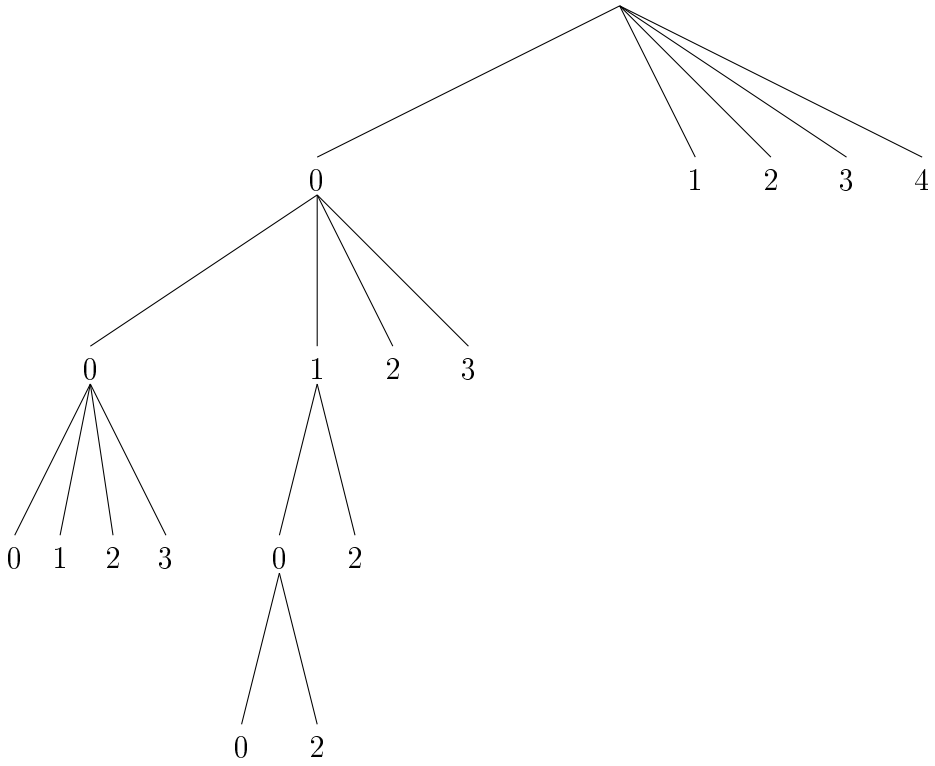

Figura 9 – Árvore 8

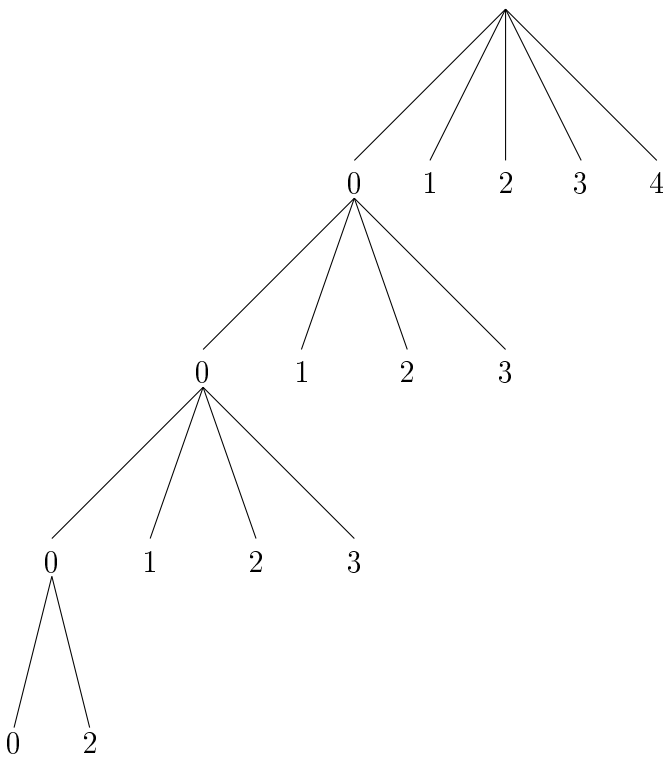

Figura 10 – Árvore 9

Tabela 4 – Frequência das árvores obtidas pelo estimador BIC.

Árvores	Português Brasileiro	Português Europeu
Árvore 1	9	17
Árvore 2	0	4
Árvore 3	6	2
Árvore 4	1	1
Árvore 5	0	1
Árvore 6	5	0
Árvore 7	4	0
Árvore 8	1	0
Árvore 9	0	1
Total	26	26

4.2.2 Discussão

Obtivemos resultados importantes com relação à frequência dos contextos. Destacamos eles a seguir.

1. Nas árvores de português brasileiro e português europeu, os contextos 100, 200, 300, 20, 30, 2, 3 e 4 apareceram 100% dos textos;
2. As sequências 0 e 00 não aparecem como contexto em nenhum texto das duas línguas.
3. O contexto 000 apareceu em 100% dos textos brasileiros e em 96% dos textos portugueses;
4. O contexto 1 apareceu em 62% dos textos brasileiros e em 77% dos textos portugueses;
5. O contexto 01 apareceu em 4% dos textos brasileiros e em 19% dos textos portugueses;
6. Os contextos 001 e 201 apareceram em 35% dos textos brasileiros e em 4% dos textos portugueses;
7. O contexto 10 apareceu em 35% dos textos brasileiros e em 88% dos textos portugueses;
8. Os contextos 010 e 210 apareceram em 46% dos textos brasileiros e em 12% dos textos portugueses;
9. Os contextos 0010 e 2010 apareceram em 19% dos textos brasileiros e não apareceram nos textos portugueses;
10. A árvore 1 é a árvore de contextos mais frequente das duas línguas. Ela aparece em 35% dos textos brasileiros e em 65% dos textos portugueses.

Não é o objetivo deste trabalho a interpretação linguística dos resultados, porém é nítida a diferença entre as duas línguas. Alguns resultados são compatíveis com Galves et al. (2012), como o fato de 100, 200, 300, 20, 30, 3 e 4 serem contextos para ambas as línguas. As árvores obtidas a partir de textos brasileiros foram, em média, maiores que as de textos portugueses. Isso ocorreu porque contextos como 001, 201, 010, 210, 0010 e 2010 foram mais presentes em textos brasileiros do que em textos portugueses.

5 CONSIDERAÇÕES FINAIS

As árvores probabilísticas de contexto são uma classe de modelos promissora, com aplicações em diversas áreas, tais como teoria da informação, genética e linguística. A utilização dessa classe de modelos para a detecção de ritmo em textos escritos é uma abordagem inovadora. Galves et al. (2012) aborda esse tema e introduz o *Smallest Maximizer Criterion* (SMC), que é um critério de seleção de modelos na classe das árvores probabilísticas de contexto. O SMC é livre de constantes, ao contrário do BIC, que precisa da especificação de uma constante de penalização. Apesar de a consistência dos algoritmos não depender das constantes utilizadas, a escolha das constantes é importante para amostras finitas. Nas aplicações linguísticas deste trabalho, a constante de penalização foi escolhida com base nos resultados obtidos em Galves et al. (2012).

A versão modificada do Algoritmo Contexto não poderia ser aplicada nesse caso por dois motivos. O primeiro é que esse estimador precisa de amostras muito maiores para que a estimação seja adequada. O segundo é que as árvores de contexto associadas aos textos brasileiros e portugueses precisariam ser completas. Nenhuma das árvores obtidas pelo estimador BIC foram completas.

A aplicação dessa classe de modelos em linguística não é restrita a Brasil e Portugal. Um trabalho semelhante poderia ser feito com base no português de Angola, Moçambique, Guiné-Bissau, Cabo Verde, São Tomé e Príncipe e Timor Leste. Também pode-se pensar em analisar idiomas diferentes do português.

REFERÊNCIAS

Bühlmann, P. and Wyner, A. J., Variable length Markov chains, **Ann. Statist.** 27, 480-513, 1999.

Collet, P., Galves, A., Leonardi, F., Random Perturbations of Stochastic Processes with Unbounded Variable Length Memory. **Eletronic Journal of Probability**, Vol. 13, Paper nº. 48, 13451361,2008.

Csiszár, I. and Talata, Z., Context tree estimation for not necessarily finite memory processes, via BIC and MDL, **IEEE Trans. Inform. Theory** 52, Number3, 1007-1016, 2006.

F. Ferrari and A. Wyner (2003), Estimation of general stationary processes by variable length Markov chains, **Scand. J. Statist.** 30, no. 3, 459-480

Galves, A., Galves, C., Garcia, J. E., Garcia, N. L. and Leonardi, F., Context tree selection and linguistic rhythm retrieval from written texts, **Annals of Applied Statistics**, 6 1, 186-209 (2012)

Galves, Antonio. Leonardi, Florencia., Exponential inequalities for empirical unbounded context trees. **Progress in Probability** 60 (2008), 257-270.

Galves, A., Locherbach, E., Stochastic chains with memory of variable length. **TICSP** Series 38: 117-133, 2008.

Garcia, N. L., Moreira, L., Stochastically Perturbed Chains of Variable Memory, **Journal of Statistical Physics**, Volume 159, Número 5, 1107-1126, 2015.

Garivier, A. and Leonardi, F. Context tree selection: a unifying view. ArXiv:1011.2424v3. **Stochastic processes and their applications** 121, pp. 2488-2506, 2011.

Matta, D. H., **Algoritmos de estimação para Cadeias de Markov de Alcance Variável - aplicações a detecção do ritmo em textos escritos**. Dissertação (Mestrado em Estatística) - Instituto de Matemática, Estatística e Computação Científica, UNICAMP. Campinas, 2008

Rissanen, J., A universal data compression system, **IEEE Trans. Inform. Theory** 29(5): 656-664, (1983).

The R Project for Statistical Computing, http://www.r-project.org.

Vigário, M. (2003). **The prosodic word in European Portuguese**, Mouton de Gruyter.

APÊNDICES

Apêndice A: Algoritmo Contexto em R

```
fcontexto <- function(dados,d,constante,perturbacao = 0){

  # perturbação

  if (perturbacao != 0){
    for (i in 1:length(dados)){
      dados[i] <- dados[i]*rbinom(1,1,1-perturbacao)
    }
  }

  # valor de |A| (considerando A = {0,1,...,|A| - 1})

  alfabeto <- max(dados)+1

  # limiar delta_n

  limiar <- constante*log(length(dados))

  # função para converter (i,j) em sequência

  fconverte <- function(i,j){
    conversao <- character(1)
    for (l in (d+1-j):1){
      conversao <- paste(floor((i-1)/alfabeto^(l-1)),conversao,sep="")
      i <- ((i-1) %% alfabeto^(l-1)) + 1
    }
    conversao
  }
```

```
# função para completar matriz

fcompleta <- function(matriz,q) {
  d <- (ncol(matriz)-1)
  for (j in 1:d){
    for (i in 1:(q^d)){
      matriz[floor((i-1)/q)+1,j+1] <- matriz[floor((i-1)/q)+1,j+1] + matriz[i,j]
    }
  }
  matriz
}

# contagem de N_n(s,a) e N_n(s)

num <- array(0,c(alfabeto^d,d+1,alfabeto))

for (tempo in (d+1):length(dados)){
  i <- 0
  ajuste <- 0
  for (passado in (tempo-d):(tempo-1)){
    i <- i + (alfabeto^ajuste)*(dados[passado])
    ajuste <- ajuste + 1
  }
  num[i+1,1,(dados[tempo])+1] <- num[i+1,1,(dados[tempo])+1] + 1
}

for (i in 1:alfabeto){
  num[,,i] <- fcompleta(num[,,i],alfabeto)
}
```

```
numt <- apply(num,c(1,2),sum)

# probabilidades de transição estimadas

tr <- array(0,c(alfabeto^d,d+1,alfabeto))

for (j in 1:(d+1)){
  for (i in 1:alfabeto^(d+1-j)){
    for (k in 1:alfabeto){
      if (numt[i,j] == 0) tr[i,j,k] <- 1/alfabeto
      else tr[i,j,k] <- num[i,j,k]/numt[i,j]
    }
  }
}

# função divergência

fdiv <- function(i,j,b){
  p <- rep(0,alfabeto)
  q <- rep(0,alfabeto)
  for (a in 1:alfabeto){
    p[a] <- p[a] + tr[(i-1)*alfabeto+b,j-1,a]
    q[a] <- q[a] + tr[i,j,a]
  }
  div <- numeric(alfabeto)
  for (a in 1:alfabeto){
    if (p[a] == 0) div[a] <- 0
    else if (q[a] == 0) div[a] <- Inf
    else div[a] <- p[a]*log(p[a]/q[a])
  }
```

```
  sum(div)
}

# função Delta

fdelta <- function(i,j){
  delta <- 0
  for (b in 1:alfabeto){
    if (numt[(i-1)*alfabeto+b,j-1] != 0){
      delta <- delta + numt[(i-1)*alfabeto+b,j-1]*fdiv(i,j,b)
    }
  }
  delta
}

# achando a matriz C

matrizc <- matrix(0,alfabeto^d,d+1)

for (l in (d-1):0){
  for (i in 1:(alfabeto^l)){
    for (proximo in 1:alfabeto){
      if (matrizc[(i-1)*alfabeto+proximo,(d-l)] == 1){
        matrizc[i,(d+1-l)] <- 1
      }
    }
    if (matrizc[i,(d+1-l)] == 0 && numt[i,(d+1-l)] >= 1){
      matrizc[i,(d+1-l)] <- as.integer(fdelta(i,d-l+1) > limiar)
    }
  }
```

```
  }

  # achando a árvore

  arvore <- character()
  arvore[1] <- "sequencia vazia"
  index <- 1
  valor <- 0
  for (i in 1:alfabeto^d){
    for (j in 1:d){
      valor <- 0
      if (matrizc[i,j] == 0 && matrizc[floor((i-1)/alfabeto)+1,j+1] == 1){
        valor <- 1
      }
      if (valor == 1 && numt[i,j] != 0){
        arvore[index] <- fconverte(i,j)
        index <- index + 1
      }
    }
  }
  arvore
}
```

Apêndice B: Versão modificada do Algoritmo Contexto em R

```
fgalves <- function(dados,d,delta,perturbacao = 0){

  # pertubação

  if (perturbacao != 0){
    for (i in 1:length(dados)){
      dados[i] <- dados[i]*rbinom(1,1,1-perturbacao)
    }
  }

  # valor de |A| (considerando A = {0,1,...,|A| - 1})

  alfabeto <- max(dados)+1

  # função para converter (i,j) em sequência

  fconverte <- function(i,j){
    conversao <- character(1)
    for (l in (d+1-j):1){
      conversao <- paste(floor((i-1)/alfabeto^(l-1)),conversao,sep="")
      i <- ((i-1) %% alfabeto^(l-1)) + 1
    }
    conversao
  }

  # função para completar matriz

  fcompleta <- function(matriz,q) {
    d <- (ncol(matriz)-1)
```

```
  for (j in 1:d){
    for (i in 1:(q^d)){
      matriz[floor((i-1)/q)+1,j+1] <- matriz[floor((i-1)/q)+1,j+1] + matriz[i,j]
    }
  }
  matriz
}

# contagem de N_n(s,a) e N_n(s)

num <- array(0,c(alfabeto^d,d+1,alfabeto))

for (tempo in (d+1):length(dados)){
  i <- 0
  ajuste <- 0
  for (passado in (tempo-d):(tempo-1)){
    i <- i + (alfabeto^ajuste)*(dados[passado])
    ajuste <- ajuste + 1
  }
  num[i+1,1,(dados[tempo])+1] <- num[i+1,1,(dados[tempo])+1] + 1
}

for (i in 1:alfabeto){
  num[,,i] <- fcompleta(num[,,i],alfabeto)
}

numt <- apply(num,c(1,2),sum)

# probabilidades de transição estimadas
```

```
tr <- array(0,c(alfabeto^d,d+1,alfabeto))

for (j in 1:(d+1)){
  for (i in 1:alfabeto^(d+1-j)){
    for (k in 1:alfabeto){
      if (numt[i,j] == 0) tr[i,j,k] <- 1/alfabeto
      else tr[i,j,k] <- num[i,j,k]/numt[i,j]
    }
  }
}

# matriz com os Deltas

adelta <- array(0,c(alfabeto^d,d,alfabeto))

for(a in 1:alfabeto){
  for (j in 1:d){
    for (i in 1:alfabeto^(d-j+1)){
      adelta[i,j,a] <- abs(tr[i,j,a]-tr[floor((i-1)/alfabeto)+1,j+1,a])
    }
  }
}

mdelta <- apply(adelta,c(1,2),max)

# achando a matriz de zeros e uns

matriz <- matrix(0,alfabeto^d,d+1)

for (j in 1:d){
```

```
    for (i in 1:alfabeto^(d-j+1)){
      if (matriz[i,j] == 1){
        matriz[floor((i-1)/alfabeto)+1,j+1] <- 1
      }
      else if (matriz[floor((i-1)/alfabeto)+1,j+1] == 0){
        matriz[floor((i-1)/alfabeto)+1,j+1] <- as.integer(mdelta[i,j] > delta)
      }
    }
  }

  # achando a árvore

  arvore <- character()
  arvore[1] <- "sequencia vazia"
  index <- 1
  valor <- 0
  for (i in 1:alfabeto^d){
    for (j in 1:d){
      valor <- 0
      if (matriz[i,j] == 0 && matriz[floor((i-1)/alfabeto)+1,j+1] == 1){
        valor <- 1
      }
      if (valor == 1){
        arvore[index] <- fconverte(i,j)
        index <- index + 1
      }
    }
  }
  arvore
}
```

Apêndice C: Estimador BIC em R

```
fbic <- function(dados,d,constante,perturbacao = 0){

  # perturbação

  if (perturbacao != 0){
    for (i in 1:length(dados)){
      dados[i] <- dados[i]*rbinom(1,1,1-perturbacao)
    }
  }

  # valor de |A| (considerando A = {0,1,...,|A| - 1})

  alfabeto <- max(dados)+1

  # função para converter (i,j) em sequência

  fconverte <- function(i,j){
    conversao <- character(1)
    for (l in (d+1-j):1){
      conversao <- paste(floor((i-1)/alfabeto^(l-1)),conversao,sep="")
      i <- ((i-1) %% alfabeto^(l-1)) + 1
    }
    conversao
  }

  # função para completar matriz

  fcompleta <- function(matriz,q) {
    d <- (ncol(matriz)-1)
```

```
  for (j in 1:d){
    for (i in 1:(q^d)){
      matriz[floor((i-1)/q)+1,j+1] <- matriz[floor((i-1)/q)+1,j+1] + matriz[i,j]
    }
  }
  matriz
}

# contagem de N_n(s,a) e N_n(s)

num <- array(0,c(alfabeto^d,d+1,alfabeto))

for (tempo in (d+1):length(dados)){
  i <- 0
  ajuste <- 0
  for (passado in (tempo-d):(tempo-1)){
    i <- i + (alfabeto^ajuste)*(dados[passado])
    ajuste <- ajuste + 1
  }
  num[i+1,1,(dados[tempo])+1] <- num[i+1,1,(dados[tempo])+1] + 1
}

for (i in 1:alfabeto){
  num[,,i] <- fcompleta(num[,,i],alfabeto)
}

numt <- apply(num,c(1,2),sum)

# probabilidades de transição estimadas
```

```
  tr <- array(0,c(alfabeto^d,d+1,alfabeto))

  for (j in 1:(d+1)){
    for (i in 1:alfabeto^(d+1-j)){
      for (k in 1:alfabeto){
        if (numt[i,j] == 0) tr[i,j,k] <- 1/alfabeto
        else tr[i,j,k] <- num[i,j,k]/numt[i,j]
      }
    }
  }

  # achando a matriz v e a matriz x

  matriz <- matrix(0,alfabeto^d,d+1)
  matrizv <- matrix(0,alfabeto^d,d+1)
  matrizx <- matrix(0,alfabeto^d,d+1)

  for (j in 1:(d+1)){
    for (i in 1:alfabeto^(d+1-j)){
      for (k in 1:alfabeto){
        if (tr[i,j,k] != 0){
          matriz[i,j] <- matriz[i,j] + num[i,j,k]*log(tr[i,j,k])
        }
        matriz[i,j] <- matriz[i,j] - constante*log(length(dados))
        if (numt[i,j] == 0) matriz[i,j] <- 0
      }
    }
  }

  for (i in 1:alfabeto^d){
```

```
  matrizv[i,1] <- matriz[i,1]
}
for (j in 2:(d+1)){
  for (i in 1:alfabeto^(d+1-j)){
    for (anterior in 1:alfabeto){
      matrizv[i,j] <- matrizv[i,j] + matriz[(i-1)*alfabeto+anterior,j-1]
    }
    if (matriz[i,j] > matrizv[i,j]){
      matrizv[i,j] <- matriz[i,j]
    }
  }
}

for (j in 1:(d+1)){
  for (i in 1:alfabeto^(d+1-j)){
    matrizx[i,j] <- as.integer(matrizv[i,j] > matriz[i,j])
  }
}

# achando a árvore

arvore <- character()
arvore[1] <- "sequencia vazia"
index <- 1
valor <- 0
for (i in 1:alfabeto^d){
  for (j in 1:d){
    valor <- 0
    sufixo <- i
    while (matrizx[i,j] == 0 && matrizx[floor((sufixo-1)/alfabeto)+1,j+valor+1] =
```

```
        valor <- valor + 1
        sufixo <- floor((sufixo-1)/alfabeto+1)
        if (j+valor > d) break
      }
      if (valor == d-j+1 && numt[i,j] > 0){
        arvore[index] <- fconverte(i,j)
        index <- index + 1
      }
    }
  }
  arvore
}
```