Universidade de Brasília
Departamento de Estatística

Os ritmos escritos do português brasileiro e moçambicano: um estudo comparativo em Cadeias com Ordem Variável

Arthur Rodrigues Melo

Monografia apresentada para obtenção do título de Bacharel em Estatística.

Brasília
2018

Arthur Rodrigues Melo

Os ritmos escritos do português brasileiro e moçambicano: um estudo comparativo em Cadeias com Ordem Variável

Orientador:
Prof. Dr. **Lucas Moreira**

Monografia apresentada para obtenção do título de Bacharel em Estatística.

Brasília
2018

AGRADECIMENTOS

Primeiramente, agradeço a Universidade de Brasilia por ter proporcionado tempo, espaço e ferramentas para a minha construção como ser social. Em particular, como servidor e representante dessa instituição, prestos meus agradecimentos ao Prof. Dr. Lucas Moreira, que orientou distintamente os cursos desse trabalho e reavivou em mim a busca pelo conhecimento estatístico. Também menciono aqui a minha gratidão pelo ex-aluno do curso de Estatística (e atualmente doutorando em Matemática) Alex Barros Azevedo Bonfim pelo grande auxílio no desenvolvimento dessa monografia. Agradeço ainda meus dois grandes amigos de curso, Guilherme Mendes e Rafael Ribeiro, pelos auxílios e experiências que tivemos no decorrer de nossas graduações e escritas.

Agradeço imensamente a minha família por ter me dado a oportunidade de crescimento num ambiente sadio, amoroso e com valores de alteridade.

Agradeço aos meus amigos (especialmente ao Pedro Augusto, Gabriel Alves, Victor Franco e a Ana Luiza) e a minha namorada (Elis Uchoa de Lima) pela oportunidade que me concederam de aprender a amar um outro ser humano sem qualquer ressalva.

Finalmente, agradeço a todos os seres sencientes que contribuíram, direta ou indiretamente, na realização dessa monografia.

"To be thoroughly lazy is a tough job,
but somebody has to do it.
Industrious people build industry.
Lazy people build civilization."
(Kazuaki Tanahashi)

Resumo

Neste trabalho comparamos o ritmo escrito do português brasileiro e do português moçambicano por meio de cadeias com ordem variável tomando valores num alfabeto $\mathcal{A} = \{0, 1, 2, 3, 4\}$. Para isso, decodificamos textos não-literários de ambas as línguas de acordo com a presença ou ausência de tonicidade silábica e início de palavra prosódica. Inicialmente, três estimadores foram estudados: o Algoritmo Contexto, proposto por Rissanen (1983), a versão modificada do Algortimo Contexto, proposta por Galves e Leonardi (2008), e o estimador BIC, encontrado em Csiszár e Talata (2006). A fim de analisar o melhor estimdor para a aplicação, estudos simulados foram realizados para os três estimadores mencionados, para diferentes constantes de penalização (no caso do estimador BIC), limiares de poda (Algoritmo Contexto e a sua versão modificada), tamanhos de amostra e estruturas de contextos. Notamos que o estimador BIC apresentou o melhor desempenho para os casos estudados, sendo usado, assim, na aplicação. Ao total, obtivemos o retorno de dez árvores de contextos distintas para os 56 textos analisados do português brasileiro e do português moçambicano.

Palavras-chave: Cadeias de Ordem Variável; Árvores de Contexto; Comparação Rítmica; Linguística.

Abstract

In this paper, we compare the written rhythm of Brazilian and Mozambican Portuguese variant through a chain with variable length taking values in alphabet $\mathcal{A} = \{0, 1, 2, 3, 4\}$. For this, we decoded non-literary texts in both languages according to the presence or absence of their syllabic tonicity and their prosodic word onset. Initially, three estimators were studied: the Context Algorithm, proposed by Rissanen (1983), a modified version of the Context Algorithm, proposed by Galves and Leonardi (2008), and the BIC estimator, found in Csiszár and Talata (2006). In order to analyze the best estimator for the application, simulated studies were performed for the three previously mentioned estimators, for different penalty constants (in the case of the BIC estimator), pruning thresholds (Context Algorithm and its modifeid version), samples sizes and contextual structures. We noticed that the BIC estimator presented the best performance for the studied cases, being thus used in the application. In total, we obtained the return of ten distinct contexts trees for the 56 analyzed texts in Brazilian Portuguese and Mozambican Portuguese.

Key words: Chains With Variable Length; Context Trees; Rhythmic Comparison; Linguistics.

SUMÁRIO

Introdução

Na Linguística, a área de conhecimento responsável pelo estudo da natureza e funcionamento das variações de tom, intensidade, duração e ritmo encontradas nos discursos humanos é conhecido como prosódia. Cabe à prosódia, por exemplo, analisar quando determinada entoação significa uma afirmação ou uma integorração dentro de uma língua em particular.

O elemento prosódico que garante fluidez e movimento a uma língua por meio da tonicidade entre intervalos discursivos é conhecido como ritmo. Cada língua possui suas idiossincrasias rítmicas, pois foram expostas a diferentes contextos filológicos, culturais, sociais e históricos. No presente trabalho, exploramos essas distinções rítmicas entre o português brasileiro e o português de Moçambique dentro de uma perspectiva quantitativa. A equivalência lexical (ou seja, o conjunto de palavras é o mesmo para as duas línguas, com exceções de regionalismos, neologismos e correlatos) e a similitude na duração silábica entre tais manifestações do português possibilitam e legitimam a nossa comparação. Para tanto, serão analisados textos escritos dessas línguas por meio de cadeias de ordem variável.

Cadeias de ordem variável (também conhecidas como *árvores probabilísticas de contexto ou cadeiras com alcance variável*) são uma classe de modelos análoga às cadeias markovianas com ordem k, porém, mais generalizantes e ricas que estas (Bühlmann and Winder, 1999), pois as suas ordens variam conforme uma função do passado. Isso permite uma economia no número de parâmetros utilizados no modelo e o seu emprego em casos não-Markovianos (Galves et al., 2009). Foram introduzidas por Rissanen (1983) como uma alternativa superior ao algoritmo de Ziv-Lempel (1978). Rissanen chamou de *contexto* a porção do passado que influencia na probabilidade de transição do próximo estado. Visto que nenhum contexto é sufixo de outro contexto, podemos representar o espaço de estados minimal (ou seja, o conjunto de contextos gerados a partir do nosso processo) por meio de

uma árvore (árvore de contextos).

No mesmo artigo, Rissanen também introduziu uma forma de estimarmos cadeias de ordem variável denominada *Algoritmo Contexto*. Basicamente, podamos a nossa árvore por meio de uma função ganho baseada na divergência de Kullback-Leibler de forma que haja uma perda aceitável de informação em troca de uma eficiência na descrição do modelo. Variações desse estimador são encontradas em Galves e Leonardi (2008), que propuseram um novo operador para poda baseado nas diferenças sucessivas entre probabilidades de transição empíricas, Csiszár e Talata (2006), onde os critérios de estimação são fundamentados por meio do critério de informação bayseana (comumente escrito como BIC), Ferrari e Wyner (2003), que avaliaram a estimação para o caso não-limitado, etc. Aplicações de cadeias com ordem variável em Linguística são encontradas em Charlotte, Galves et al (2009), onde os ritmos escritos encontrados no português brasileiro e europeu foram comparados através dessa classe de modelos.

Além das aplicações em Linguística, cadeias de ordem variável também são utilizadas na Teoria da Informação, compressão de dados, bioquímica, bioinformática, musicologia e aprendizado de máquinas.

O presente trabalho encontra-se disposto da seguinte forma: no Capítulo 1, mostramos conceitos e definições básicas da temática tratada, assim como a descrição do Algoritmo Contexto, a sua versão modificada proposta em Galves e Leonardi (2008) - junto com suas suposições - e o estimador BIC. No Capítulo 2, apresentamos as bases metodológicas usadas, descrevendo a forma como as simulações foram usadas, como os textos foram selecionados, codificados e interpretados para um alfabeto $\mathcal{A} = \{0, 1, 2, 3, 4\}$. O Capítulo 3 foi reservado para o estudo simulado de cadeias de ordem variável, pois todos os algoritmos de estimação tratados nesse trabalho possuem algum tipo de constante definido *a priori* e, assim, uma análise mais pormenorizada para diferentes tamanhos e tipos de árvores é de extrema importância. No Capítulo 4, apresentamos e discutimos os resultados obtidos para a aplicação do estimador BIC na comparação rítmica feita entre os textos do português brasileiro e moçambicano. Finalmente, reservamos o Capítulo 5 para a conclusão do texto, apresentando uma reflexão condensada das contribuições, dificuldades e sugestões encontradas no presente trabalho.

Capítulo 1

Revisão Bibliográfica

A seguir, apresentamos uma breve revisão das notações utilizadas neste trabalho, assim como um delineamento do aporte teórico que sustentará as etapas de estimação e análise dos dados.

1.1 Notações e Conceitos Preliminares

Seja $\mathcal{A} = \{0, 1, ..., N-1\}$ um alfabeto finito com tamanho $|\mathcal{A}| = N$, $N \in \mathbb{N}, N > 0$. Dados dois inteiros $m \leq n$, denotamos por a_m^n a sequência $a_m a_{m+1} ... a_n$, sendo $l(a_m^n) = n - m + 1$ o seu comprimento, $a_i \in \mathcal{A}$. Caso $n < m$, $a_m^n = \emptyset$ e $l(a_m^n) = 0$. A sequência semi-infinita $...a_{n-1}a_n$ de símbolos em $\mathcal{A}$ é denotada por $a_{-\infty}^n$ e possui comprimento $l(a_m^n) = \infty$.

Denotamos os conjuntos de todas as sequências de tamanho j com símbolos em $\mathcal{A}$ e de todas as sequências semi-infinitas com símbolos em $\mathcal{A}$, respectivamente, por

$$\mathcal{A}_{-\infty}^{-1} = \mathcal{A}^{\{...,-2,-1\}} \; e \; \mathcal{A}^* = \bigcup_{j=0}^{\infty} \mathcal{A}_{-j}^{-1},$$

sendo que $\mathcal{A}^0$ representa o conjunto que contém apenas a sequência vazia.

Dadas as sequências $u \in \mathcal{A}^* \cup \mathcal{A}^\infty$ e $v \in \mathcal{A}^*$, denotamos por uv a sequência resultante da concatenação entre u e v, com comprimento $l(u) + l(v)$. Por exemplo, considere u_∞^{-n-1} e v_{-n}^{-1}. Então $uv = ..., u_{-n-1}, v_{-n}, ..., v_{-2}, v_{-1}$. Caso $v = \emptyset$, $uv = u$. O mesmo vale para $u = \emptyset$.

Considere as sequências $w \in \mathcal{A}^*$ e $u \in \mathcal{A}^* \cup \mathcal{A}^\infty$. Caso exista $v \in \mathcal{A}^* \cup \mathcal{A}^\infty$, com $l(v) \geq 1$, tal que $u = vw$, então w é um sufixo de u. Essa relação pode ser simbolizada por $w \prec u$. Se $w \prec u$ ou $w = u$, então $w \preceq u$. O maior sufixo de uma sequência w finita é denotado por $suf(w)$.

Um subconjunto contável de $\mathcal{A}^* \cup \mathcal{A}^\infty$ será uma árvore $\mathcal{T}$ se satisfaz a *propriedade do sufixo*, ou seja, nenhum $s_1 \in \mathcal{T}$ é sufixo de algum $s_2 \in \mathcal{T}$. *Folhas* são todos os elementos de $\mathcal{T}$. Os *nós internos* são elementos de $\mathcal{T}$ que são sufixos de outras folhas. São *descendentes* de um nó interno s todas as sequências $as, a \in \mathcal{A}$. Uma árvore $\mathcal{T}$ é *completa* caso todo nó interno possua, exatamente, $|\mathcal{A}|$ descendentes e *irredutível* se nenhum $s \in \mathcal{T}$ puder ser substituído por um sufixo de s sem violar a propriedade do sufixo.

A cardinalidade de $\mathcal{T}$ é denotada por $|\mathcal{T}|$ e a sua *profundidade* é definida como

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

Se $h(\mathcal{T}) < \infty$, a nossa árvore será *limitada*. Caso contrário, $\mathcal{T}$ será considerada *ilimitada*. $\mathcal{T}|_K$ denotará a árvore truncada em um inteiro K, ou seja,

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

A fim de facilitar a compreensão e fluidez do texto, apresentamos a seguir as definições de cadeias de Markov de ordem k, função alcance e função contexto, respectivamente, seguidas pela definição formal de uma cadeia de Markov com ordem variável. Denotaremos as variáveis aleatórias por letras maiúsculas (por exemplo, X_t), os valores determinísticos fixados por letras minúsculas (como x_t) e X_k^t como o vetor $(X_k, X_{k-1}, ..., X_t)$ de tamanho $|X_k^t| = t - k + 1$ (o mesmo valendo para o caso determinístico representado, por exemplo, por x_k^t).

Definição 1.1 *Seja $X_t, t \in \mathbb{Z}$, um processo estocástico tomando valores em um alfabeto $\mathcal{A}, |\mathcal{A}| = N$. Tal processo é uma cadeia de Markov de ordem k caso exista $k \in \{0, 1, ...\}$ tal que, para todo $t \in \mathbb{Z}$, temos*

$$P(X_t = x_t | X_{-\infty}^{t-1} = x_{-\infty}^{t-1}) = P(X_t = x_t | X_{t-k}^{t-1} = x_{t-k}^{t-1}). \tag{1.1}$$

Definição 1.2 *Seja $\mathcal{A}^k$ o conjunto de sequências com comprimento k do processo $X_t, t \in \mathbb{Z}$. Definimos a função alcance $\varrho : \mathcal{A}^k \longrightarrow \{0, 1, ..., k\}$ de tal forma que*

$$\varrho(x_{t-k}^{t-1}) = min\{\varrho \leq k | P(X_t = x_t | X_{t-k}^{t-1} = x_{t-k}^{t-1}) = P(X_t = x_t | X_{t-\varrho}^{t-1} = x_{t-\varrho}^{t-1})\}. \quad (1.2)$$

Ou seja, delimitamos o número mínimo de passos anteriores necessários para escolhermos qual será o próximo símbolo do alfabeto (para o caso independente, $k = 0$ e $\varrho = 0$).

Definição 1.3 *Seja $\mathcal{A}^k$ o conjunto de sequências com comprimento k do processo $X_t, t \in \mathbb{Z}$. Definimos a função contexto $C : \mathcal{A}^k \longrightarrow \sum_{m=1}^{k} \mathcal{A}^m$ como*

$$C : x_{t-k}^{t-1} \longrightarrow x_{t-\varrho(t-k)^{t-1}}^{t-1}. \quad (1.3)$$

O vetor resultante $C(x_{t-k}^{t-1}) = (x_{t-\varrho}, ..., x_{t-1})$ é chamado de contexto, ou seja, a parte do passado necessária para o cômputo da probabilidade de transição do próximo símbolo.

Definição 1.4 *Seja $X_t, t \in \mathbb{Z}$, uma cadeia de Markov estacionária tomando valores em um alfabeto $\mathcal{A}, |\mathcal{A}| = N$, e $C(.)$ sua respectiva função contexto. Tal processo será uma cadeia com ordem variável caso $k \in \{0, 1, ...\}$ seja o menor valor tal que*

$$|C(x_{-\infty}^{t})| = \varrho(x_{-\infty}^{t}) \leq k, \forall x_{-\infty}^{t} \in \mathcal{A}^{\infty} \ . \quad (1.4)$$

Basicamente, uma cadeia com ordem variável k contém uma cadeia de Markov com ordem fixa k em sua estrutura, mas com uma propriedade a mais: podemos agrupar determinadas probabilidades de transição da cadeia, gerando, assim, uma característica de memória variável. Isso faz com que o número de passos anteriores necessários para a predição do elemento atual seja uma função do próprio passado, variando de símbolo para símbolo.

Como a distribuição probabilística de uma cadeia com ordem variável é totalmente determinada pelas suas respectivas probabilidades de transição, podemos representar o seu espaço de estados minimal através de uma árvore, chamada de *árvore probabilística de contextos*. Os elementos dessa árvore são os contextos gerados pela função contexto associada à cadeia com alcance variável em questão. Note que, de acordo com a definição da função contexto, a propriedade do sufixo é satisfeita.

Genericamente, podemos caracterizar uma cadeia de ordem variável da seguinte forma:

- Raízes no topo;

- Ramos crescem para baixo;
- Todo nó interno tem, no máximo, $|\mathcal{A}|$ descendentes;
- O contexto $u = C(x_{-\infty}^{-1})$ é representado por um ramo. O seu sub-ramo do topo é determinado por x_{-1}, o próximo sub-ramo por x_{-2} e assim por diante.

As definições a seguir formalizam o conceito de uma cadeia de ordem variável e quando essa classe de modelos é compatível com um processo $\mathbf{X}$:

Definição 1.5 *Seja $X_t, t \in \mathbb{Z}$, um processo estacionário e ergódico tomando valores em um alfabeto $\mathcal{A}, |\mathcal{A}| = N$. Dada uma sequência $s \in \mathcal{A}^*$, temos que*

$$\mu_X(s) = \begin{cases} \mathbb{P}(X_1^{l(s)} = s), & \text{se } s \neq \emptyset, \\ \mu_X(a), & \text{se } s = \emptyset \ . \end{cases} \tag{1.5}$$

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

$$p_x(a|s) = \begin{cases} \mathbb{P}(X_0 = a | X_{-l(s)}^{-1} = s), & \text{se } s \neq \emptyset, \\ \mu_X(a), & \text{se } s = \emptyset \ . \end{cases} \tag{1.6}$$

Definição 1.6 *Chamamos de cadeia de ordem variável em $\mathcal{A}$ o par ordenado $(\mathcal{T}, \overline{p})$ que satisfaz as seguintes condições:*

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 1.7 *O processo $\boldsymbol{X}$ será compatível com a cadeia de ordem variável $(\mathcal{T}, \overline{p})$ caso satisfaça as condições:*

1. *$\mathcal{T}$ é a árvore de contextos do processo $\boldsymbol{X}$;*
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 cadeia de ordem variável do processo $\mathbf{X}$.

Exemplo 1.1 *Seja* $\mathcal{A} = \{0, 1, 2\}$ *o nosso alfabeto e* $\mathcal{T} = \{000, 100, 10, 01, 1, 2\}$ *a nossa árvore. A representação gráfica da árvore está na Figura 1. Note que* $\mathcal{T}$ *satisfaz a propriedade do sufixo, sendo, de fato, uma árvore. As sequências* $\{000, 10, 01, 11, 2\}$ *são folhas de* $\mathcal{T}$ *e* $\{00, 0, 1, \emptyset\}$ *os seus nós internos. A árvore* $\mathcal{T}$ *não é completa, pois nem todos nós internos possuem comprimento igual a três (cardinalidade de* $\mathcal{A}$*), é limitada e irredutível. Ainda, a árvore* $\mathcal{T}$ *truncada em 2 é dada por* $\mathcal{T}|_2 = \{00, 10, 11, 01, 2\}$.

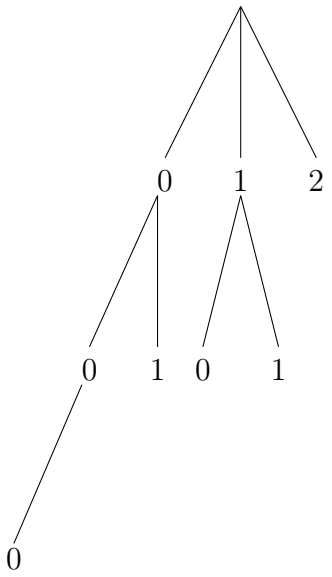

Figura 1.1 – Exemplo de árvore

1.2 Processos de Estimação de Árvores de Contextos

Seja $X_1, ..., X_n$ uma amostra aleatória do processo $\mathbf{X}$. Dado um inteiro positivo d, para toda sequência finita $s = x_{-j}^0$, tal que $j \leq d$, $N_n(s, a)$ representa o número de vezes que a sequência s ocorreu seguida pelo símbolo $a \in \mathcal{A}$ na amostra selecionada, ou seja,

$$N_n(s, a) = \sum_{i=d+1}^{n} \mathbb{I}(X_{i-j}^i = s, X_{i+1} = a). \tag{1.7}$$

Então o número de ocorrências da sequência s em toda a amostra é

$$N_n(s) = \sum_{a \in \mathcal{A}} N_n(s, a). \tag{1.8}$$

O conjunto de todas as sequências finitas $s = x_{-j}^0$, tal que $j \leq d$, que possuem ao menos uma aparição na amostra será denotado por $\mathcal{V}_n$. Ou seja,

$$\mathcal{V}_n = \left\{ s \in \bigcup_{j=0}^{d} \mathcal{A}^j : N_n(s) \geq 1 \right\}. \tag{1.9}$$

A seguir, definimos um conjunto de árvores factíveis a fim de selecionar candidatos para a árvore de contextos de $\mathbf{X}$.

Definição 1.8 *Uma árvore $\mathcal{T}$ é factível se*

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 árvores factíveis geradas a partir da amostra será denotado por $\mathcal{F}_n$. Precisamos encontrar uma árvore factível que se aproxime de $\mathcal{T}_X$ para fins de estimação. Note que a profundidade de $\mathcal{T}_X$ não pode ser maior que d (caso a árvore seja limitada), pois as sequências $s \in \bigcup_{j=0}^{d} \mathcal{A}^j$. Ademais, não se faz necessário o conhecimento *a priori* da profundidade de $\mathcal{T}_X$ para estimarmos a mesma. Assim, d pode ser uma função crescente do tamanho n de nossa amostra .

Para todo elemento $a \in \mathcal{A}$ e para toda sequência finita s, a probabilidade de transição empírica é dada por

$$\hat{p}_n(a|s) = \begin{cases} \frac{N_n(s,a)}{N_n(s)}, & \text{se } N_n(s) > 0 \\ \frac{1}{|\mathcal{A}|}, & \text{se } N_n(s) = 0 \ . \end{cases} \tag{1.10}$$

Note que a definição de $\hat{p}_n(a|s)$ é assintoticamente equivalente ao estimador de máxima verossimilhança $\frac{N_{(s,a)}}{N_{(s)}}$ da probabilidade de transição $p(a|s)$.

1.3 O Algoritmo Contexto

O Algoritmo Contexto proposto por Rissanen (1983) pode ser descrito em três etapas básicas: primeiro, selecionamos a maior árvore factível da nossa amostra; após isso, comparamos a discrepância entre as probabilidades de transição empíricas relativas ao maior sufixo de um contexto e seus descendentes (caso a discrepância seja maior que um dado valor limítrofe, mantemos o contexto; caso contrário, o podamos). Por fim, continuamos o processo de comparação até não ser mais possível uma nova poda da árvore.

Usamos a divergência de Kullback-Leibler, definida abaixo, para mensurar a discrepância entre duas probabilidades em $\mathcal{A}$.

Definição 1.9 *A divergência de Kullback-Leibler entre as medidas de probabilidades P e Q em $\mathcal{A}$ é dada por:*

$$D(P;Q) = \sum_{a\in\mathcal{A}} P(a) log \frac{P(a)}{Q(a)} \ . \tag{1.11}$$

Caso $P(a) = 0$, então $D(P;Q) = 0$; se $P(a) > Q(a) = 0$, então $D(P;Q) = +\infty$.

Podamos a sequência bs para s de acordo com a seguinte função ganho:

$$\Lambda_n(s) = \sum_{b\in\mathcal{A}:bs\in\mathcal{V}_n} N_n(bs) D(\hat{p}_n(.|bs); \hat{p}_n(.|s)). \tag{1.12}$$

Denotamos o limiar para poda utilizado no Algoritmo Contexto por δ_n.

A interpretação por trás do Algoritmo Contexto é simples: o limiar para poda está relacionado à quantidade de informação que poderíamos perder ao usarmos apenas o maior sufixo de um contexto no lugar de seus descendentes. Caso a função ganho Λ_n retornasse um valor menor que tal limite, o processo de estimação tornar-se-ia mais enxuto sem perda significativa de informação.

A seguir, explicitamos as etapas do Algoritmo Contexto:

Etapa 1 : Seja $X_1, ..., X_n$ uma amostra aleatória do processo $\mathbf{X}$ e considere $\hat{\mathcal{T}}_0$ como a maior árvore factível da nossa amostra - ou seja, a árvore cuja cardinalidade é a maior;

Etapa 2 : Seja $S = \{suf(w) : w \in \hat{\mathcal{T}}_0\}$. Aplicamos, então, a função $\Lambda_n(s)$ em todos os elementos de S que tenham aparecido pelo menos duas vezes na nossa amostra, de forma que não exista $s' \in S$ e $s \prec s'$. Caso $\Lambda_n(s) < \delta_n$, substituímos por s todos os elementos $as \in \hat{\mathcal{T}}_0$. Chamamos de $\hat{\mathcal{T}}_1$ a nova árvore obtida após o procedimento;

Etapa 3 : Repetimos a Etapa 2 até não ser mais possível realizar o procedimento em $\hat{\mathcal{T}}_i$. Denotamos a última árvore obtida por $\hat{\mathcal{T}}_C(X_1^n)$.

1.4 Versão Modificada do Algoritmo Contexto

Galves e Leonardi (2008) propuseram uma variante do Algoritmo Contexto onde a divergência de Kullback-Leibler em $\Lambda_n(s)$ é substituída por uma simples diferença entre probabilidades de transição empíricas para as sequências s e $suf(s)$. Nesse caso, consideramos o operador $\Delta_n(s)$ tal que

$$\Delta_n(s) = \max_{a\in\mathcal{A}} |\hat{p}_n(a|s) - \hat{p}_n(a|suf(s))| \ . \tag{1.13}$$

Definição 1.10 *Para quaisquer $\delta > 0$ e $d < n$, a árvore de contexto estimada $\hat{\mathcal{T}}_n^{\delta,d}$ é o conjunto de todas as sequências $s \in \bigcup_{j=0}^{d} \mathcal{A}^k$ tais que $\Delta_n(asuf(s)) > \delta$, para algum $a \in \mathcal{A}$, e $\Delta_n(us) \leq \delta$, para todo $u \in \bigcup_{k=1}^{d-l(s)} \mathcal{A}^k$. Caso $\hat{\mathcal{T}}_n^{\delta,d} = \emptyset$, então $\hat{\mathcal{T}}_n^{\delta,d} = \{\emptyset\}$. Para o caso $l(s) = d$, definimos $\bigcup_{k=1}^{0} \mathcal{A}^k = \emptyset$.*

A fim de garantir a consistência forte da versão alternativa do Algoritmo Contexto, faz-se necessário supor que a nossa árvore probabilística de contexto satisfaz as seguintes definições.

Definição 1.11 *Um processo $\boldsymbol{X}$ é não-nulo caso*

$$a_X = \inf_{a \in \mathcal{A}, s \in \mathcal{T}} \{p(a|s)\} > 0 \ . \tag{1.14}$$

Definição 1.12 *Seja a sequência $\{a_k\}_{k \geq 0}$ definida por*

$$a_0 = \sum_{a \in \mathcal{A}} \inf_{s \in \mathcal{T}} \{p(a|s)\}, \tag{1.15}$$

$$a_k = \inf_{u \in \mathcal{A}^k} \sum_{a \in \mathcal{A}} \inf_{s \in \mathcal{T}, u \prec s} \{p(a|s)\}. \tag{1.16}$$

Dizemos que um processo $\boldsymbol{X}$ é somável se possui a sequência $\{b_k\}_{k \geq 0}, b_k = 1 - a_k$ tal que

$$b = \sum_{k \in \mathcal{N}} b_k < \infty. \tag{1.17}$$

Um processo somável $\mathbf{X}$ garante que dois passados coincidindo nos primeiros k símbolos possuam, assintoticamente, a mesma influência na predição do próximo símbolo da sequência, à medida que k cresce.

1.5 Critério BIC

Seja $X_1, ..., X_n$ uma amostra aleatória do processo $\mathbf{X}$. A árvore factível $\hat{\mathcal{T}}_0$ selecionada nesse método de estimação deve considerar a função de verossimilhança da amostra e a complexidade da árvore, de modo que escolhamos $\hat{\mathcal{T}}_0$ tal que a função de verossimilhança seja comparativamente alta e a complexidade do modelo seja baixa. Dada uma amostra X_1^n, o Critério de Informação Bayesiana (BIC) para uma árvore factível $\mathcal{T}$ é

$$BIC_{\mathcal{T}}(X_1^n) = -logML_{\mathcal{T}}(X_1^n) + c|\mathcal{T}|log\ n, \tag{1.18}$$

tal que $c \in \mathbb{R}$. A constante de penalização $c = (|\mathcal{A}| - 1)/2$ foi definida por Czizár e Talata (2006) e busca priorizar modelos cujo número de parâmetros livres é menor em comparação a outros mais complexos. Definimos o estimador BIC a seguir.

Definição 1.13 *O estimador BIC é dado por*

$$\hat{\mathcal{T}}_{BIC}(X_1^n) = \arg\min_{\mathcal{T}\in\mathcal{F}_n} \mathrm{BIC}_{\mathcal{T}}(X_1^n) \ . \tag{1.19}$$

Na prática, a tarefa de calcular o valor desse estimador para uma árvore de contextos é árdua, pois o número de árvores factíveis pode ser grande. Visto isso, Czizár e Talata (2006) propuseram um algoritmo recursivo e bastante útil para o cômputo de $\hat{\mathcal{T}}_{BIC}(X_1^n)$.

Definimos, recursivamente, para todo $s \in \mathcal{V}_n$, a função

$$V_s(x_1^n) = \begin{cases} \max\{n^{-c}\ \mathrm{ML}_s(x_1^n), \prod_{b\in\mathcal{A}:bs\in\mathcal{V}_n} V_{bs}(x_1^n)\}, & \text{se } l(s) < d \\ n^{-c}\ \mathrm{ML}_s(x_1^n), & \text{se } l(s) = d \ . \end{cases} \tag{1.20}$$

e a indicadora

$$\mathcal{X}_s(x_1^n) = \begin{cases} \mathbb{I}\left\{ \prod_{b\in\mathcal{A}:bs\in\mathcal{V}_n} V_{bs}(x_1^n) > n^{-c}\ \mathrm{ML}_s(x_1^n) \right\}, & \text{se } l(s) < d \\ 0, & \text{se } l(s) = d \ . \end{cases} \tag{1.21}$$

Caso $\{b \in \mathcal{A} : bs \in \mathcal{V}_n\} = \emptyset$, então $V_{bs}(x_1^n) = n^{-c}\ \mathrm{ML}_s(x_1^n)$ e $\mathcal{X}_s(x_1^n = 0)$. Czizár e Talata (2006) demonstraram que

$$\hat{\mathcal{T}}_{BIC}(X_1^n) = \{s \in \mathcal{V}_n : \mathcal{X}_s(x_1^n) = 0 \ e\ \mathcal{X}_{s'}(x_1^n) = 1 \ \textit{para todo}\ s' \prec s\} \ . \tag{1.22}$$

Capítulo 2

Metodologia

A fim de comparar o desempenho dos estimadores escolhidos para nossa análise, realizamos estudos simulados onde supomos conhecidas as árvores de contextos e as correspondentes probabilidades de transição. Encontramos a proporção de identificações corretas dessa árvore para o Algoritmo Contexto, a versão modificada do Algoritmo Contexto apresentada em Galves e Leonardi (2008) e o estimador BIC introduzido em Czizár and Talata (2006), através da simulação de 100 conjuntos de dados distintos. Realizamos as simulações por meio do software livre R (versão 3.5.1).

Em seguida, selecionamos textos não-literários de autores brasileiros e moçambicanos a partir de um grupo de escritos pré-escolhidos. A escolha de não incluir textos literários na amostra se deve a um possível viés de seleção, pois nesse tipo de texto as chances de encontrarmos diferenças rítmicas significantes relacionadas a neologismos, regionalismos ou a outros maneirismos linguísticos são maiores. Como textos não-literários costumam ser escritos em um formato mais padronizado, excluiríamos dramaticamente esse problema.

Cada sílaba será codificada conforme a atribuição de dois símbolos distintos da seguinte forma:

1. a sílaba é tônica ou não;

2. é início de palavra prosódica ou não.

Palavra prosódica é definida como uma componente linguística que referencia unidades morfológicas de uma forma mais geral que a palavra morfológica (Hildebrandt, 2014). Por exemplo, a frase "o menino" possui duas palavras morfológicas ("o" e "menino"),

mas apenas uma palavra prosódica (“o menino”), pois o artigo “o” encontra-se subordinado à tônica da palavra “menino”.

Assim, identificamos cada sílaba com um dos seguintes pares ordenados: (0,0), (1,0), (0,1), (1,1). O primeiro elemento de cada par indica início de palavra prosódica (o valor “1” representa início de palavra prosódica e o valor “0” indica a sua ausência), enquanto o segundo indica a tonicidade da sílaba (similarmente, “1” indica sílaba tônica e “0” indica sílaba átona). A fim de simplificar a notação, representamos os pares ordenados por números inteiros, sendo (0,0) = 0, (0,1) = 1, (1,0) = 2 e (1,1) = 3. Ademais, os finais de período serão representados pelo símbolo 4.

Exemplo 2.1 *Usamos a sentença “O menino já comeu o doce.” para ilustrarmos a 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*

Dessa forma, todos os textos serão convertidos em uma sequência de números inteiros no alfabeto $\mathcal{A} = \{0, 1, 2, 3, 4\}$. A codificação dos textos será realizada no software Perl através do programa “silaba2008.pl”, que é gratuito e encontra-se em *www.ime.usp.br/» tycho/prosody/vlmc/tools /silaba.pl.*.

Capítulo 3

Estudos simulados dos estimadores de Árvores de Contextos

Nessa seção, verificamos por meio de simulações o desempenho dos estimadores de árvores de contextos descritos no Capítulo 1. Para tal propósito, analisamos diferentes valores previamente fixados para a constante C, que incorpora o limiar δ_n para poda no Algoritmo Contexto (onde $\delta_n = Clog(n)$), e que representaria $c|\mathcal{T}|$ no termo $c|\mathcal{T}|log(n)$ para o critério BIC de uma árvore factível $\mathcal{T}$, dada uma amostra X_1^n. O limiar δ para a versão modificada do Algoritmo Contexto também foi analisado e pré-fixado. Para fins de notação e simplicidade, denotaremos por A_{BIC}, A_{C1} e A_{C2} o estimador BIC, o Algoritmo Contexto e a versão modificada do Algoritmo Contexto proposta em Galves e Leonardi (2008), respectivamente.

Exemplo 3.1 *Considere uma árvore probabilística de contextos cujos contextos e estrutura probabilística sejam dados pela Tabela 1. Foram simuladas cem amostras para os tamanhos* $n = 1000, n = 5000, n = 10000, n = 20000$ *e* $n = 100000$*. Para* A_{C1}*, a constante* C *pertence ao intervalo [1, 2] e possui incrementos centesimais; para* A_{C1}*, a constante* C *encontra-se no intervalo [0,1], com os mesmos incrementos. Verificamos a proporção de acertos para cada estimador (ou seja, o número de vezes que cada estimador identificou corretamente a árvore em questão) e tamanho de amostra na Figura 3.1 e Figura 3.2*

Tabela 3.1 – Contextos e probabilidades de transição para o Exemplo 3.1 em $\mathcal{A} = \{0, 1, 2, 3, 4\}$

Contextos	p(0|s)	p(1|s)	p(2|s)	p(3|s)	p(4|s)
1	0.2	0.3	0.25	0.1	0.15
2	0.3	0.2	0.2	0.2	0.1
3	0.35	0.3	0.15	0.1	0.1
4	0	0.4	0.2	0.3	0.1
10	0.4	0.2	0.1	0.15	0.15
20	0.35	0.3	0.2	0.1	0.05
30	0.05	0.15	0.15	0.3	0.35
100	0	0.3	0.15	0.2	0.35
200	0.2	0.2	0.3	0.15	0.15
300	0	0.25	0.1	0.2	0.45
0000	0.25	0.2	0.1	0.3	0.15
2000	0.35	0.1	0.2	0.2	0.15

Exemplo 3.2 *Considere uma árvore probabilística de contextos cujos contextos e estrutura probabilística sejam dados pela Tabela 2. Foram realizadas cem simulações com os mesmos tamanhos de amostras do Exemplo 3.1. Todos os estimadores serão terão suas constantes testadas no intervalo [0,1], com incrementos centesimais. Podemos verificar a proporção de acertos para cada estimador e tamanho de amostra na Figura 3.3, Figura 3.4 e Figura 3.5*

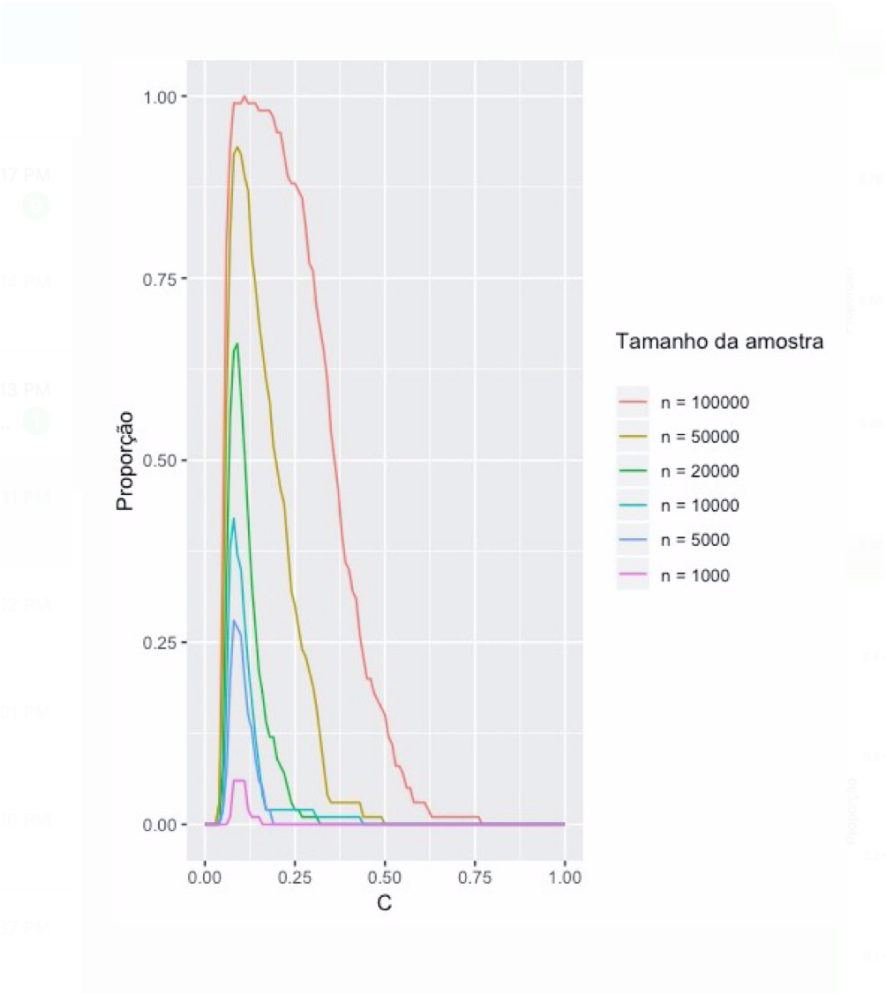

Figura 3.1 – Proporção de acertos para A_{BIC} no Exemplo 3.1

Notamos que, para a árvore usada no Exemplo 3.1, o estimador de melhor desempenho, independentemente do tamanho amostral, foi A_{BIC}. O estimador A_{C1} apresentou desempenho modesto no geral.

Para A_{BIC}, valores de C entre 0.09 e 0.13 retornaram desempenhos mais satisfatórios para o estimador. Sugerimos uma explicação, a luz da Definição 1.13: o estimador BIC prioriza modelos menos complexos; quanto menor for o valor da constante C, menor será o valor da constate de penalização c e, consequentemente, do termo $c|\mathcal{T}|log(n)$. Portanto, teremos valores menores também para $BIC_{\mathcal{T}}(X_1^n)$. É interessante notar que A_{BIC} só apresenta resultados satisfatórios de identificação quando o tamanho da amostra é consideravelmente grande. Isso se dá pelo próprio caráter do critério BIC e pela quantidade de parâmetros a serem estimados: árvores mais profundas tendem a possuir mais parâmetros de estimação,

aumentado, assim, a complexidade do modelo. O estimador BIC, por conseguinte, precisará de uma quantidade maior de informação para recuperar corretamente a árvore em questão. Tal processo se dá no crescimento do tamanho amostral.

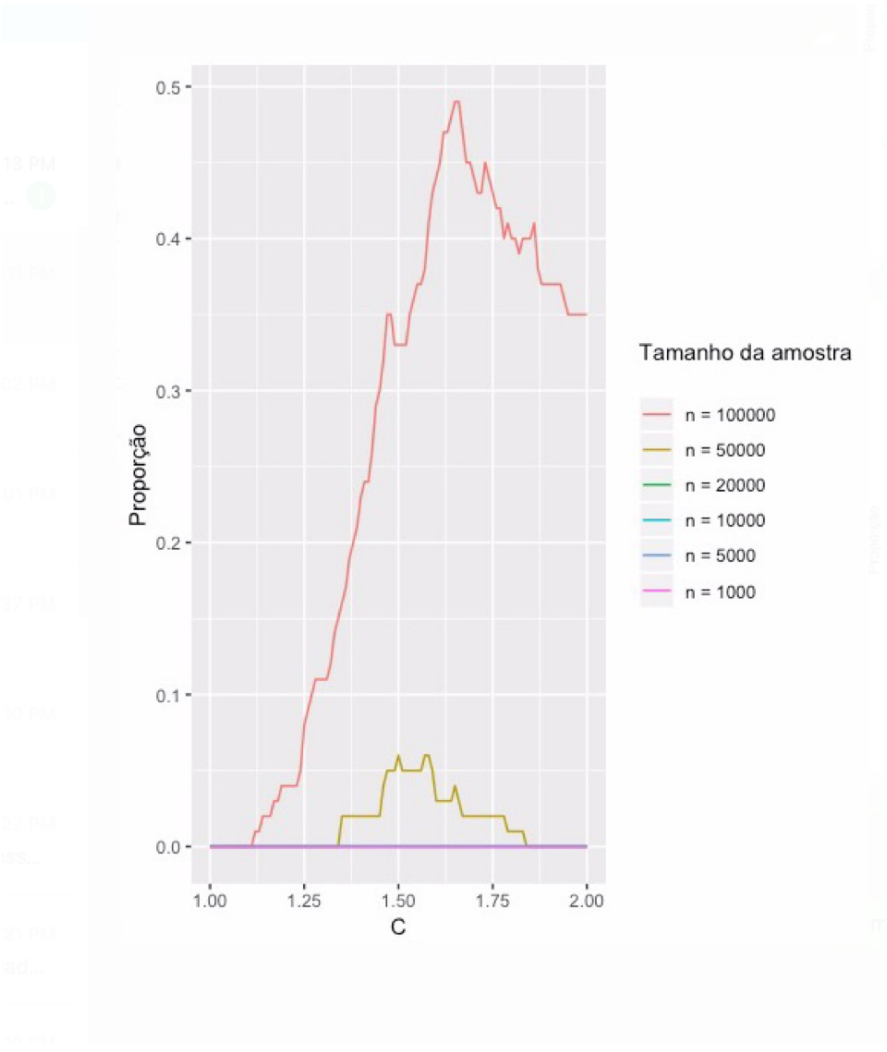

Figura 3.2 – Proporção de acertos para o estimador A_{C1} no Exemplo 3.1

Tabela 3.2 – Contextos e probabilidades de transição para o Exemplo 3.2 em $\mathcal{A} = \{0, 1, 2, 3\}$

Contextos	p(0|s)	p(1|s)	p(2|s)
00	0.2	0.15	0.65
10	0.4	0.35	0.25
20	0.5	0.35	0.15
01	0.25	0.2	0.55
11	0.25	0.35	0.4
21	0.15	0.4	0.45
02	0.05	0.4	0.55
12	0.35	0.3	0.35
22	0.25	0.3	0.45

Para A_{C1}, os resultados foram bastante diferentes. Primeiramente, notamos que o estimador não recupera corretamente a árvore escolhida na maior parte dos casos, para qualquer tamanho de amostra utilizado aqui (a proporção máxima de acertos foi igual a 0.47, para $n = 100000$). Também é possível verificar um valor mais elevado para a constante C (o estimador obteve melhor desempenho com valores entre 1.5 e 1.65). Essas duas características podem ser explicadas pelo caráter do estimador: o algoritmo sugerido por Rissanen costuma ser mais permissivo, gerando árvores com maior número de contextos; para a poda ser mais eficiente, o limiar δ_n precisa ser maior, assim como o tamanho da amostra. Isso reduziria a quantidade total de contextos e o número de parâmetros a serem estimados.

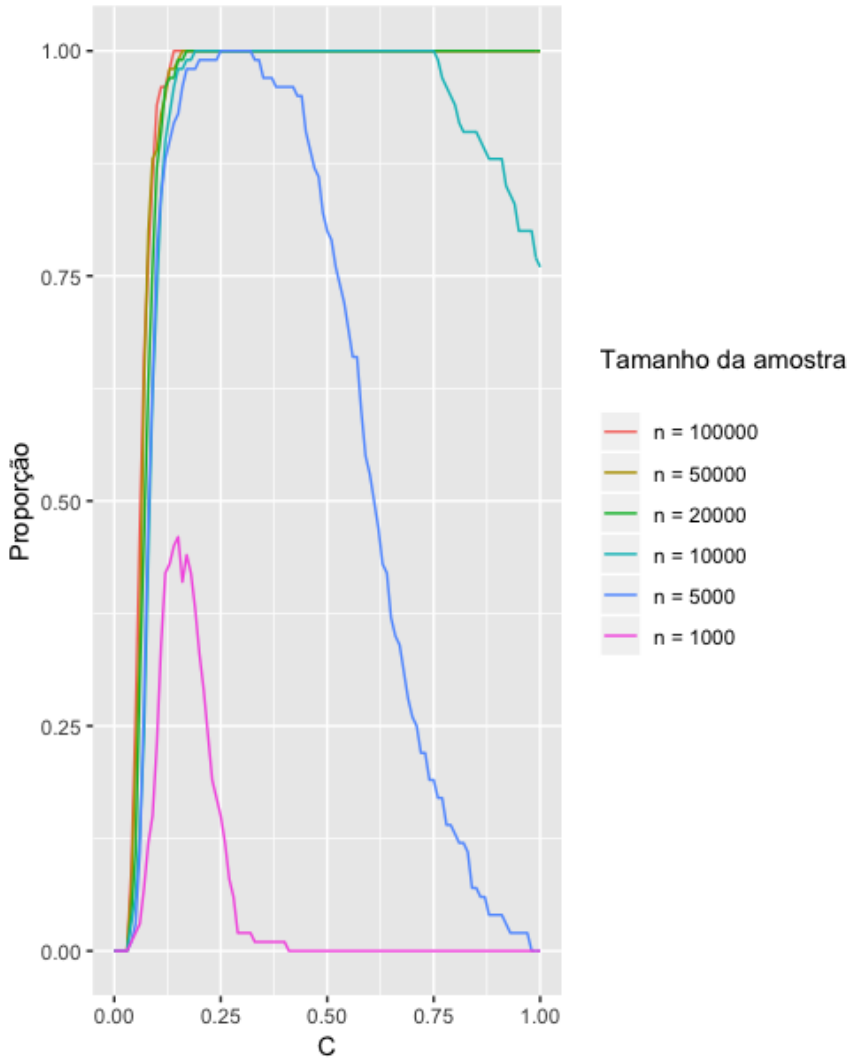

Figura 3.3 – Proporção de acertos para A_{BIC} no Exemplo 3.2

O estimador A_{C2} não foi utilizado no Exemplo 3.1, pois a árvore utilizada fere a condição de não-nulidade, já que, por exemplo, $p(0|4) = 0$. Assim, A_{C2} não retornaria a árvore corretamente, para qualquer tamanho amostral.

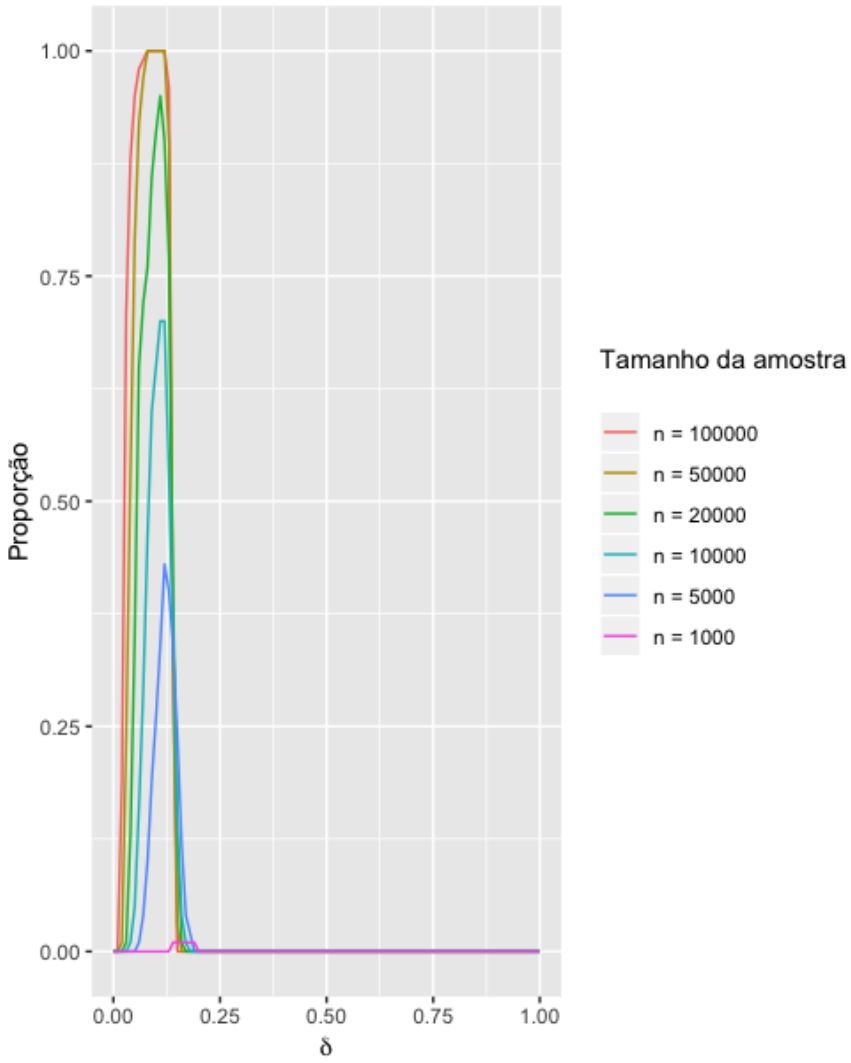

Figura 3.4 – Proporção de acertos para A_{C2} no Exemplo 3.2

Já no Exemplo 3.2, verificamos um bom desempenho para todos os estimadores. O que convergiu mais rapidamente e de forma mais abrangente foi A_{C1} (todas as árvores de contexto foram identificadas corretamente para $n = 5000$ em diante, com valores de C entre 1.1 e 2). O estimador A_{BIC} também recuperou todas as árvores de contexto corretamente para valores de n maiores ou iguais a cinco mil, para valores de C entre 0.25 e 0.3. A_{C2} não teve um bom desempenho para amostras pequenas, mas alcançou uma boa performance com amostras grandes. Os valores de δ estavam entre 0.08 e 0.12.

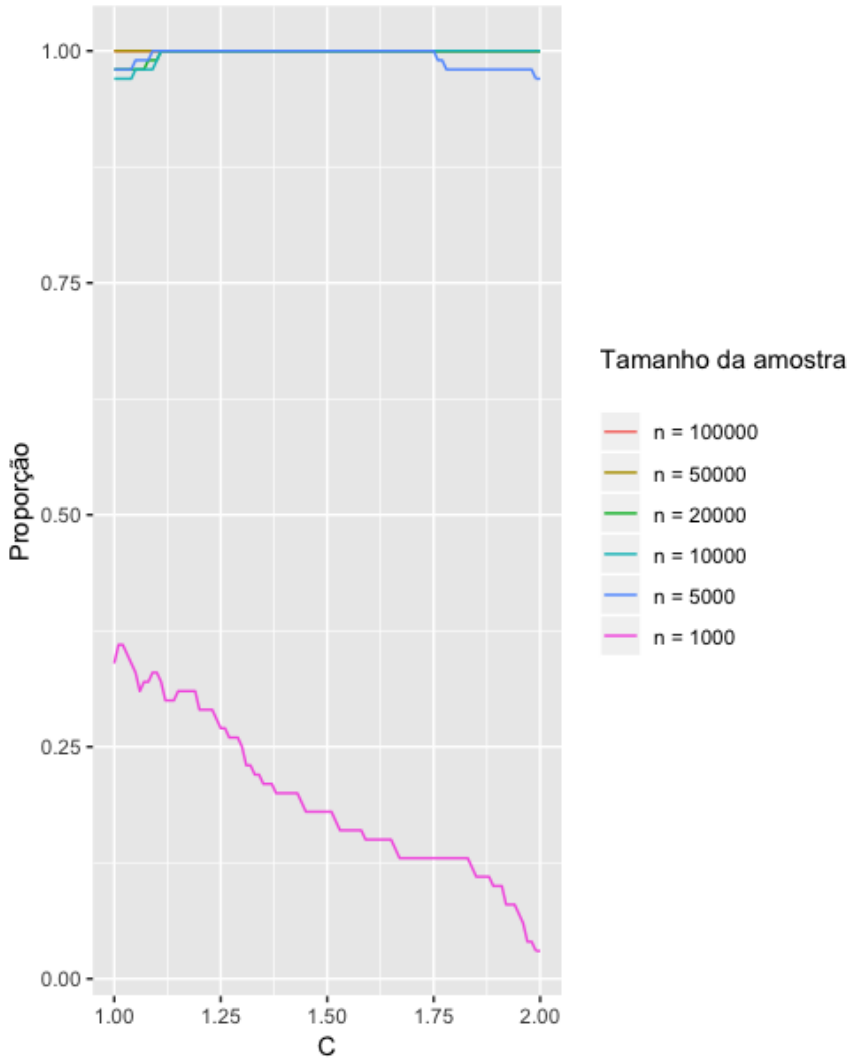

Figura 3.5 – Proporção de acertos para A_{C1} no Exemplo 3.2

Percebemos que a profundidade da árvore, a quantidade de contextos e tamanho do alfabeto afetam diretamente o desempenho dos estimadores tratados no presente trabalho. O Exemplo 3.1 nos indica que o estimador BIC seria o mais adequado para testar árvores que sejam incompletas e de maior profundidade. O Exemplo 3.2, por sua vez, indica que todos os estimadores funcionam com acurácia para árvores completas e de profundidade reduzida.

Capítulo 4

Aplicações de Cadeias de Ordem Variável em Linguística

Embora as duas formas mais conhecidas da Língua Portuguesa - a saber, o português brasileiro e o português europeu - possuam níveis de semelhança (como a ortográfica, principalmente após o Acordo Ortográfico de 1990), em suas totalidades encontramos profundas diferenças. Uma das principais se encontra no campo prosódico. No geral, as línguas são classificadas (quanto ao seu ritmo) como acentuais e silábicas. Uma língua com ritmo acentual possui tempos de duração diferentes para cada sílaba, enquanto uma língua silábica apresentará tempos de duração iguais. O português europeu é comumente descrito como uma língua com ritmo acentual e, ainda, como uma língua mono-acentual, onde não há a presença de acentos rítmicos (Andrade, 1997). Por outro lado, o português brasileiro é comumente referenciado como uma língua de ritmo misto (Frota e Vigário, 2000), ou seja, apresenta aspectos de uma língua com ritmo acentual e silábico.

O português moçambicano (língua oficial da República de Moçambique) apresenta semelhanças nos níveis fonológicos e morfossintáticos com o português brasileiro (Petter, 2007) - fatores estes que fundamentam em grande parte o ritmo de uma língua. Por exemplo, tanto no português moçambicano como no português brasileiro encontramos bastante articulação nas vogais, sejam elas átonas ou tônicas (característica de difícil registro no português europeu). Esperamos, portanto, algum tipo de compatibilidade com tais características na análise que se segue.

Baseado na seção anterior, escolhemos a constante de penalização $c = 0.05$

para a aplicação do estimador BIC em 28 textos oriundos do português brasileiro e 28 textos vindos do português moçambicano (abreviados, a partir desse ponto, para PB e PM, respectivamente). Os textos foram retirados de portais de informação (físicos ou eletrônicos) com grande tiragem no Brasil e em Moçambique. O algoritmo usado retornou 10 árvores diferentes, dispostas abaixo com a notação Árvore 1, Árvore 2, Árvore 3, etc. A frequência para o ocorrimento de cada árvore no PB ou PM está contida na Tabela 4.1.

Tabela 4.1 – Frequência das árvores encontradas pelo estimador BIC para $c = 0.05$

Árvores	PB	PM
Árvore 1	23	21
Árvore 2	0	1
Árvore 3	0	1
Árvore 4	0	1
Árvore 5	2	1
Árvore 6	0	1
Árvore 7	0	1
Árvore 8	0	1
Árvore 9	2	1
Árvore 10	1	0
Total	28	28

A Árvore 1 é a mais comum para PB e PM, ocorrendo em aproximadamente 78 e 75 % dos textos, respectivamente. É oportuno mencionar que a mesma árvore também foi a mais comum em trabalhos anteriores, como em Bomfim (2015), onde foram analisados textos do português brasileiro e do português europeu. Uma possível explicação para isso seria a normatização em textos escritos causada pelo Acordo Ortográfico de 1990, que, de certa forma, igualaria manifestações rítmicas dessas variantes linguísticas por meio da padronização ortográfica.

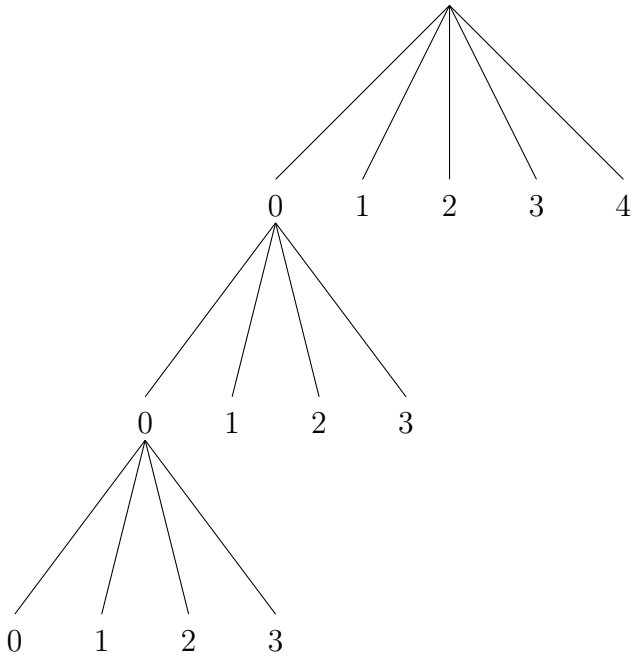

Figura 4.1 – Árvore 1

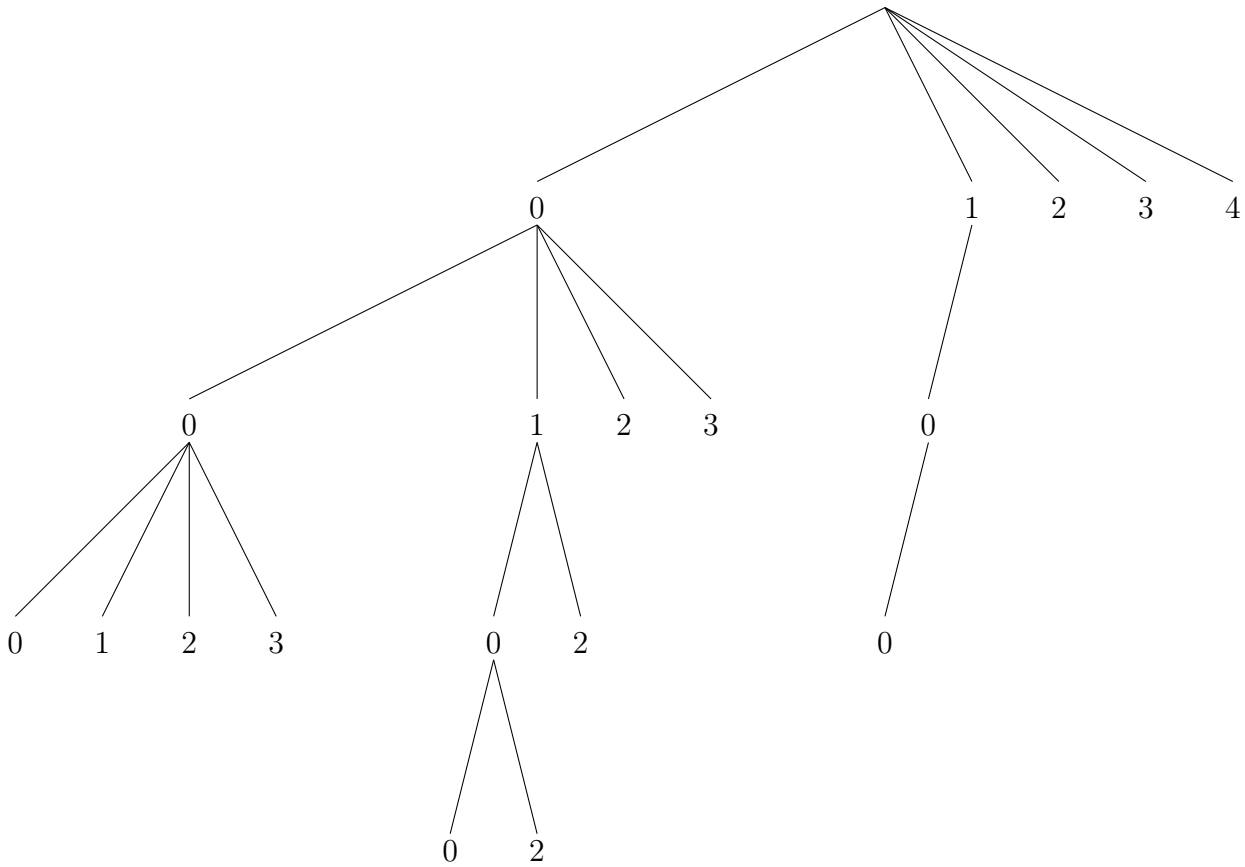

Figura 4.2 – Árvore 2

Os contextos 100, 200, 300, 20, 30, 2, 3 e 4 apareceram em todos os textos analisados para PB e PM, resultado também partilhado em outros trabalhos na literatura, como em Bomfim (2015) e Galves et al. (2012). Esses contextos representam estruturas tônicas básicas na língua portuguesa. Por exemplo, 100, 200 e 300 representariam, respectivamente, sílaba tônica sem início de palavra prosódica, início de palavra prosódica em uma sílaba átona e início de palavra prosódica em uma sílaba tônica seguidas de duas sílabas átonas que não são início de palavra prosódica (representadas pela porção "00" em cada um dos três contextos), manifestações quase que universais no Português. O fato do algoritmo ter identificado essas estruturas como importantes para a predição de um próximo símbolo em todos os textos é positivo, pois coaduna com a realidade linguística conhecida.

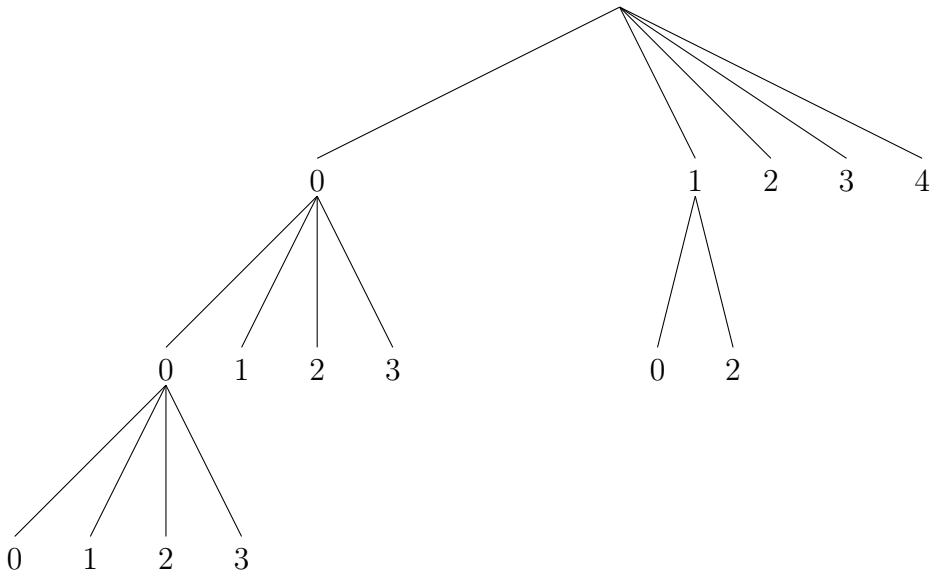

Figura 4.3 – Árvore 3

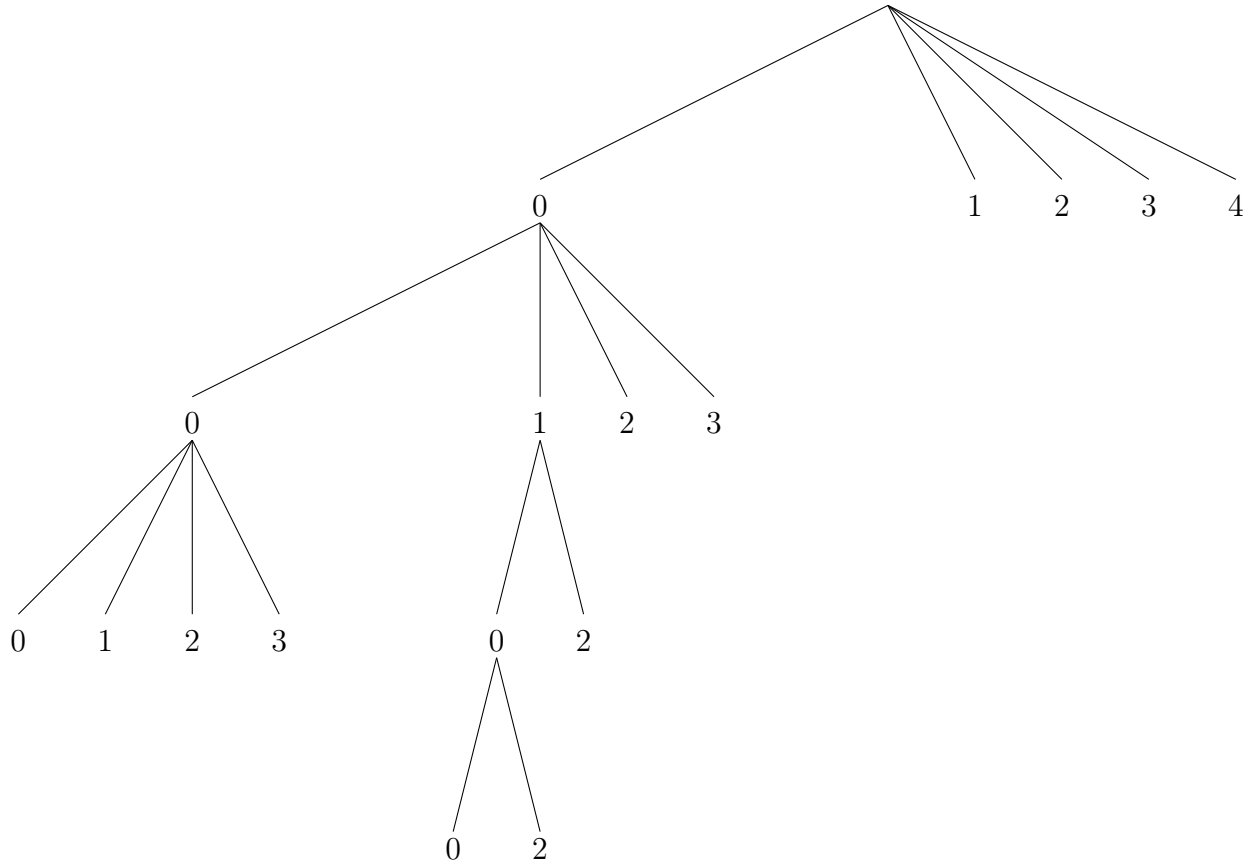

Figura 4.4 – Árvore 4

Porém, é curioso que o contexto 1 tenha aparecido em aproximadamente 93 e 82 % dos textos em PB e PM, respectivamente, pois o mesmo representaria, intuitivamente, uma estrutura basilar na tônica da língua portuguesa - a saber, o de uma sílaba tônica sem início de palavra prosódica (como a sílaba "ni" no Exemplo 2.1) - e, portanto, universal. O algoritmo usado pode não ter identificado o contexto 1 em todos os textos pela quantidade de informação disponível para análise. Através dos estudos simulados no Capítulo 4, notamos que constantes de penalização entre 0 e 0,1 costumavam apresentar padrões mais conservadores de poda - gerando, assim, uma maior quantidade de ramos - em amostras de menor tamanho.

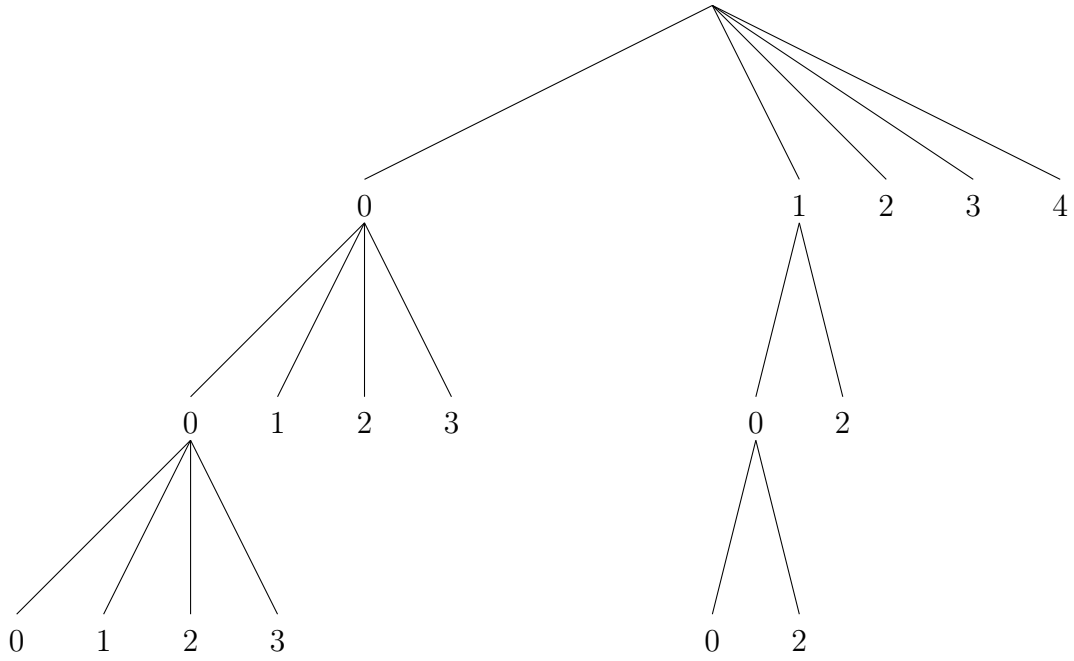

Figura 4.5 – Árvore 5

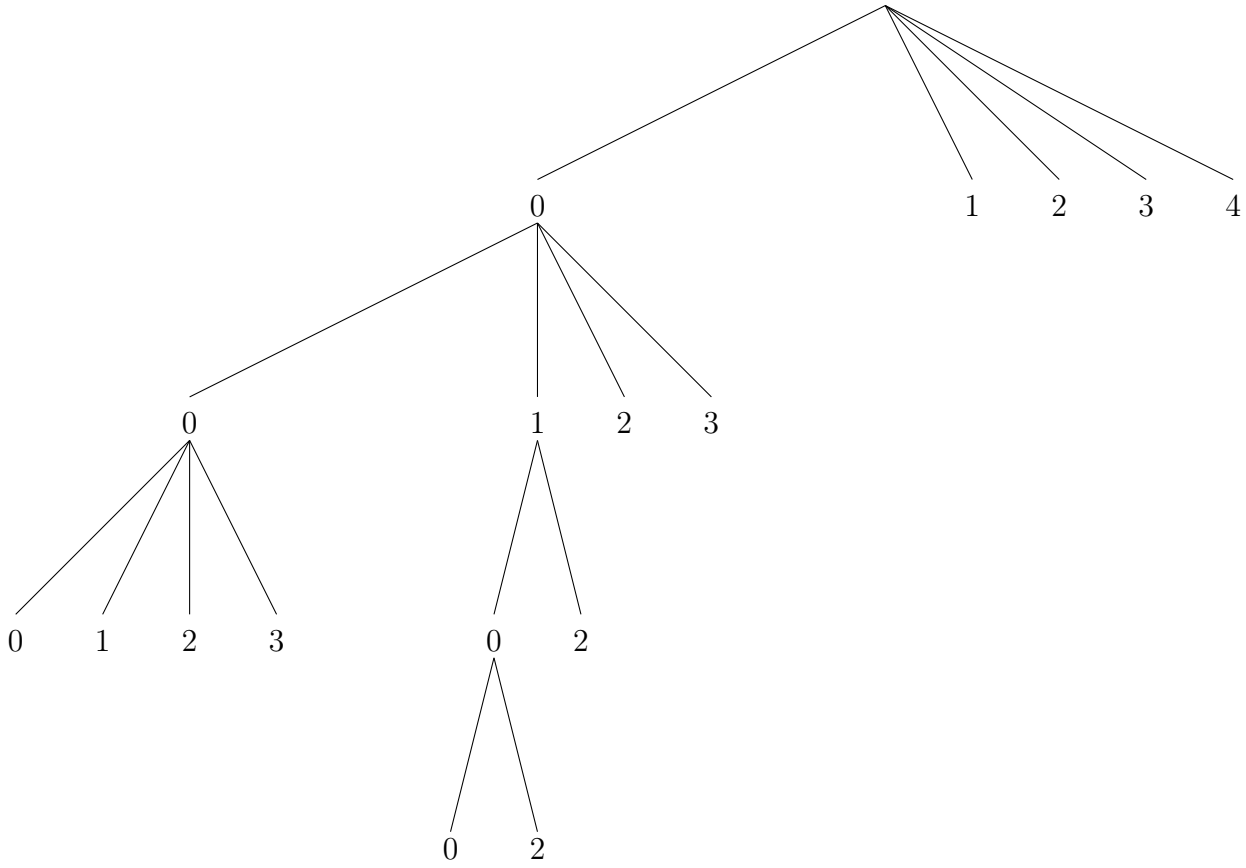

Figura 4.6 – Árvore 6

Apenas a Árvore 1 e a Árvore 5 foram comuns às duas línguas, o que do ponto de vista estrutural é razoável, já que as duas árvores possuem padrões bastante semelhantes (salvo ramificações mais profundas partindo de 0 e 1, que, inclusive, podem ter sido gerados pelos motivos explicitados no parágrafo anterior).

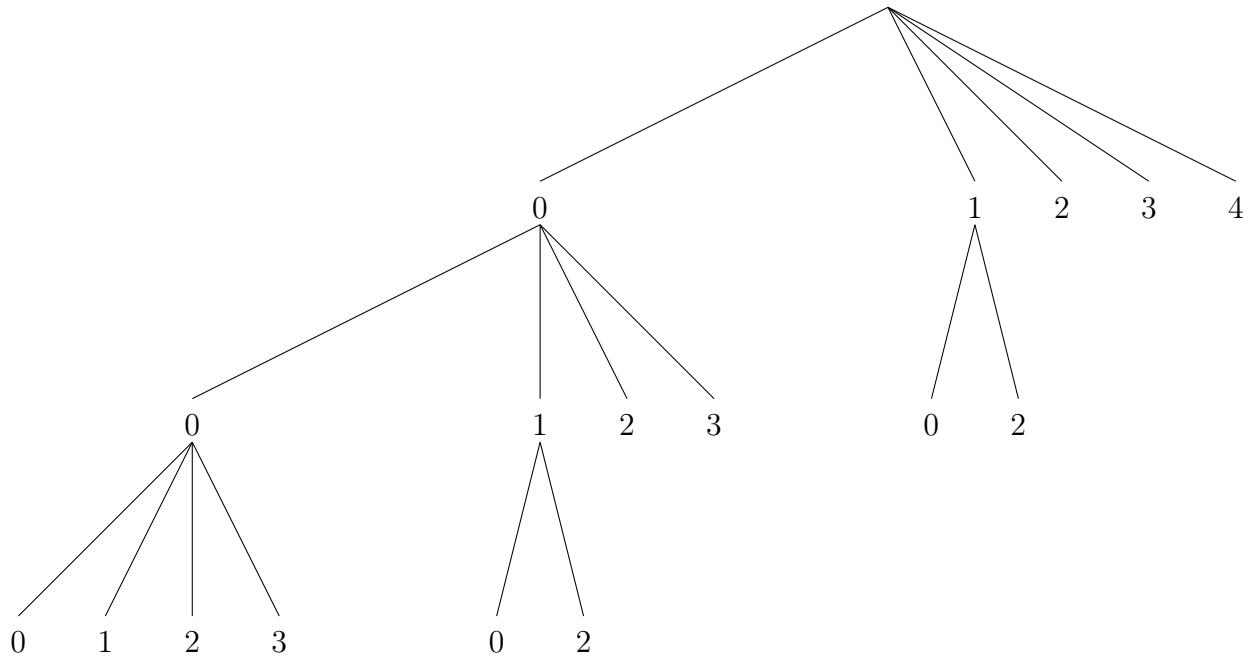

Figura 4.7 – Árvore 7

Partindo de um caráter quantitativo, percebemos similaridades entre PB e PM. Uma quantidade considerável de contextos ocorre na mesma frequência para as duas línguas, assim como a proporção de árvores comuns às variantes aqui analisadas.

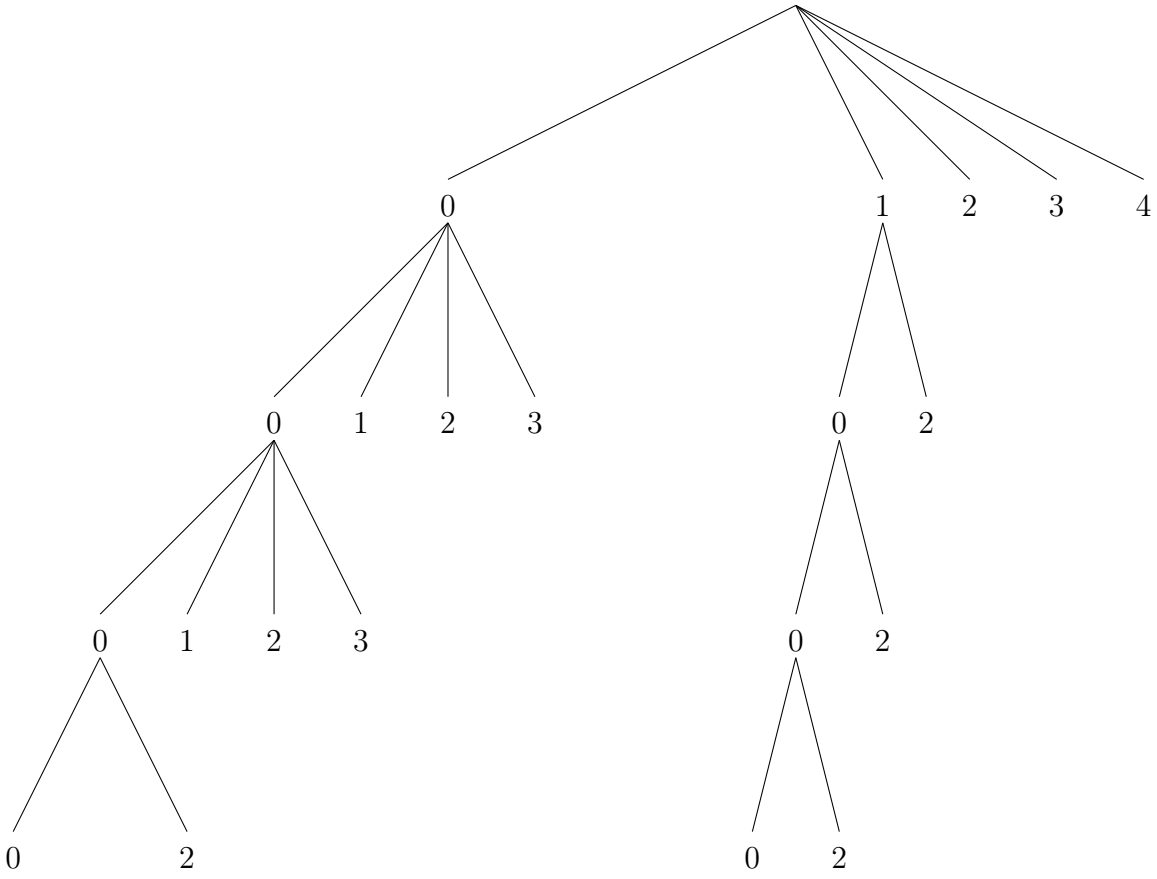

Figura 4.8 – Árvore 8

Cabe ressaltar que todas as interpretações linguísticas aqui mencionadas possuem caráter meramente exploratório, com a intenção de fomentar discussões a partir das árvores geradas.

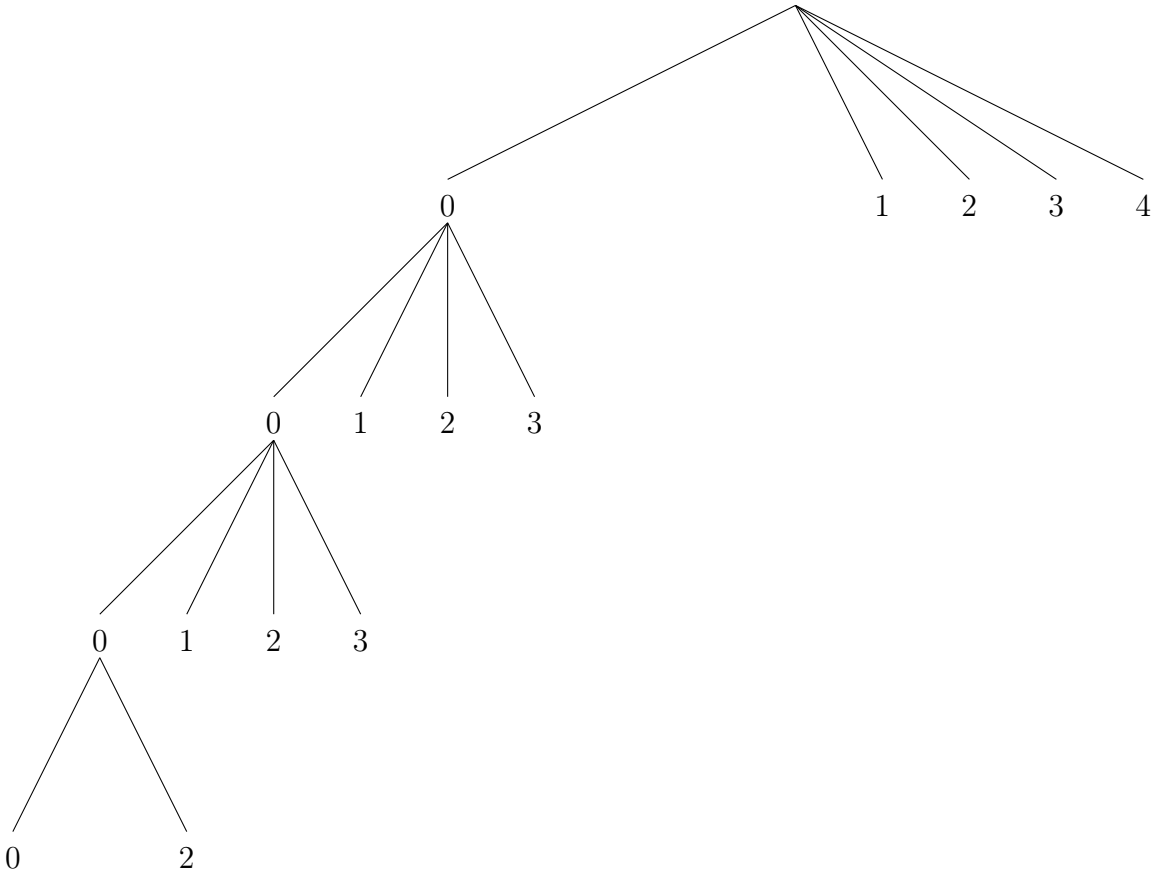

Figura 4.9 – Árvore 9

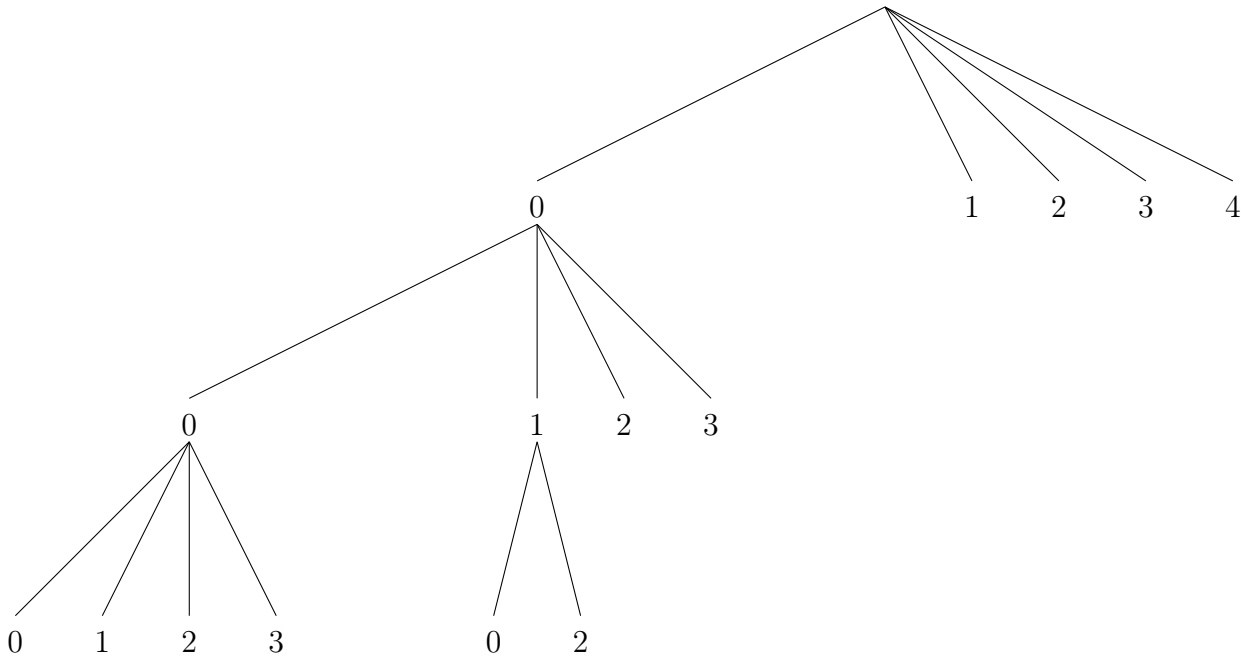

Figura 4.10 – Árvore 10

Capítulo 5

Considerações Finais

Percebemos que cadeias de ordem variável são uma classe de modelos bastante flexível, eficiente e inovadora nas suas aplicações. Lartillot et. al (2003) as usaram para identificar características específicas de gêneros e compositores musicais na tentativa de "humanizar" o aprendizado de máquinas voltado para a geração independente de peças musicais. Isso e a aplicação realizada no presente trabalho servem como exemplos do potencial que essa classe de modelos tem no campo de estudos multidisciplinares.

A análise de desempenho para os estimadores e constantes aqui tratados pode ser vista como um pequeno acréscimo à literatura sobre o tema - mesmo não sendo algo inovador ou exclusivo na área. O estimador BIC mostra-se como o mais adequado para a análise geral, mas, principalmente, de árvores incompletas e mais profundas, em grandes tamanhos amostrais e com pequenas constantes de penalização.

O uso de modelos com contaminação estocástica, como o Zero Inflado apresentado em Garcia e Moreira (2015), é uma alternativa interessante para aplicações futuras na área de linguística comparada, pois poderia considerar em sua estrutura erros de transcrição e vieses decorrentes de regionalismos e neologismos (como explicitados na metodologia do presente trabalho). Ademais, a análise com amostras maiores também deve ser encorajada, pois vimos nos estudos simulados que o estimador BIC possui desempenho satisfatório a partir de $n = 20000$, e, atualmente, é inexistente na literatura sobre o tema o uso de bancos de dados com valores mínimos de amostras tão grandes.

Finalmente, ressaltamos que embora a linguagem escrita e a linguagem falada possuam conexões, as duas são consideradas como processos distintos (Cunha, 2010). A

análise de textos escritos e padronizados limita o potencial comparativo entre as diversas formas do português, pois as diferenças prosódicas se dariam principalmente na forma falada e popular dessas variantes. Sugerimos, para trabalhos futuros, a coleta e análise de discursos sonoros gerados por falantes que tenham o português (em qualquer variante) como primeira língua.

Referências Bibliográficas

Referências Bibliográficas

[1] ABAURRE, Maria Bernadete; GALVES, Charlotte. As diferenças rítmicas entre o português europeu e o português brasileiro: uma abordagem otimalista e minimalista. DELTA, São Paulo , v. 14, n. 2, p. 377-403, 1998 .

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

[3] Csiszár, I. and Talata, Z., Context tree estimation for not necessarily finite memory processes, via BIC and MDL, IEEE Trans. Inform. Theory 52, Number 3, 1007?1016, 2006.

[4] Cunha, Ana Paula Nobre da - As segmentações não convencionais da escrita inicial (...) Revista de Estudos Linguísticos da Universidade do Porto - Vol. 7 - 2012 - 45 - 63

[5] Frota, S. & M. Vigário (2000). Aspectos de prosódia comparada: ritmo e entoação no PE e no PB. In Actas do XV Encontro da Associação Portuguesa de Linguística, 533-555. Braga: APL.

[6] Ferrari, F. and Wyner, A., Estimation of general stationary processes by variable length Markov chains, Scand. J. Statist. 30, Number 3, 459?480, 2003.

[7] 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

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

[9] Garivier, A., Consistency of the unlimited BIC Context Tree estimator. IEEE Trans. Inform. Theory 52, Number 10, 4630?4635, 2006.

[10] Hildebrandt, K. A., The Prosodic Word. In J. R. Taylor (Ed.) The Oxford Handbook of the Word, Oxford, 2014

[11] Lartillot, O., Dubnov, S., Assayag, G., Bejerano, G., A system for computer music generation by learning and improvisation in a particular style, IEEE Computer J. 36, Number 10, 73?80, 2003.

[12] 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

[13] Meireles, A. R., Tozetti, J. P., Borges, R. R. Speech rate and rhythmic variation in Brazilian Portuguese. In: Speech Prosody 2010 Conference, 2010, Chicago. Proceedings of the Speech Prosody 2010 Conference . Chicago: RG. v.1. p.1 - 4, 2010.

[14] Petter, Margarida (2007). Uma hipótese explicativa do contato entre o português e as línguas africanas. In: Papia, Brasília.

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

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

Apêndices

Apêndice A - Códigos para os algoritmos

Esse apêndice conterá as funções usadas no software R para o Algoritmo Contexto, a sua versão alternativa proposta por Galves e Leonardi (2008) e o estimador BIC.

```
############## BIC #################
fbic = function(dados,d,constante,perturbacao = 0){

# perturba?????o

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

# fun?????o para converter (i,j) em sequ??ncia

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
}

# fun?????o 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 transi?????o 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 ??rvore

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] == 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
}

########### Galves ############
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 }

########## Algoritmo Contexto ##########
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 - Códigos para as aplicações

O Apêndice B contém o código usado para analisar os 56 textos usados nesse trabalho.

```
setwd("/Volumes/NO NAME/Rascunhuras TCC - Copia/textos brasileiros")
ldf <- list()
listfiles <- dir(pattern = ".cod")
for (k in 1:length(listfiles)){
ldf[[k]] <- as.vector(scan(listfiles[k], what = 'character', quote = ""))
}
for (k in 1:length(ldf)) {
ldf[[k]] <- strsplit(ldf[[k]], "")
}
```

```
for (k in 1:length(ldf)) {
ldf[[k]][[1]] <- as.numeric(ldf[[k]][[1]])
}

for (k in 1:length(ldf)) {
ldf[[k]][[1]] <- fbic(ldf[[k]][[1]], d = 4, constante = 0.1)
}



setwd("/Users/henri/Documents/tccs")
ldf2 <- list() # creates a list
listfiles2 <- dir(pattern = ".cod") # creates the list of all the csv files
listfiles2 = listfiles2[-29]
for (k in 1:length(listfiles2)){
ldf2[[k]] <- as.vector(scan(listfiles2[k], what = 'character', quote = ""))
}
for (k in 1:length(ldf2)) {
ldf2[[k]] <- strsplit(ldf2[[k]], "")
}

for (k in 1:length(ldf2)) {
ldf2[[k]][[1]] <- as.numeric(ldf2[[k]][[1]])
}

for (k in 1:length(ldf2)) {
ldf2[[k]][[1]] <- fbic(ldf2[[k]][[1]], d = 4, constante = 0.1)
}
```

```
lista = c(ldf2, ldf)
```

```
tabulate(match(ldf, unique(ldf)))
tabulate(match(ldf2, unique(ldf2)))
tabulate(match(lista, unique(lista)))
```