Independência parcial no problema da Satisfazibilidade Probabilística

Eduardo Menezes de Morais

Tese apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Doutor em Ciências

Programa: Doutorado em Ciências da Computação

Orientador: Prof. Dr. Marcelo Finger

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro da CAPES

São Paulo, dezembro de 2017

Versão para revisão da Comissão Julgadora

Independência parcial no problema da Satisfazibilidade Probabilística

Esta é a versão original da dissertação/tese elaborada pelo candidato (Eduardo Menezes de Morais), tal como submetida à Comissão Julgadora.

Resumo

MORAIS, E. M. **Independência parcial no problema da Satisfazibilidade Probabilística**. 2017. Tese (Doutorado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2017.

O problema da Satisfazibilidade Probabilística, PSAT, apesar da sua flexibilidade, torna exponencialmente complexa a modelagem de variáveis estatisticamente independentes. Esta tese busca desenvolver algoritmos e propostas de relaxamento para permitir o tratamento eficiente de independência parcial pelo PSAT. Apresentamos uma aplicação do PSAT ao problema da etiquetagem morfossintática que serve tanto de motivação como de demonstração dos conceitos apresentados.

Palavras-chave: Lógica probabilística, Independência estatística, Conjuntos de distribuição de probabilidade, Programação linear e multilinear, Convexidade.

Abstract

MORAIS, E. M. **Partial Independence in the Probabilistic Satisfiability Problem**. 2017. Tese (Doutorado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2017.

The Probabilistic Satisfiability Problem, PSAT, despite its flexibility, makes it exponentially complicated to model statistically independent variables. This thesis develops algorithms and relaxation proposals that allow an efficient treatment of partial independence with PSAT. We also present an application of PSAT on the Part-of-speech tagging problem to serve both as motivation and showcase of the presented concepts.

Keywords: Probabilistic Logic, Statistical independence, Sets of probability distributions, Linear and multilinear programming, Convexity.

Sumário

Capítulo 1

Introdução

Este trabalho se iniciou como o estudo da aplicação do problema da Satisfazibilidade Probabilística, PSAT, para realizar classificação morfossintática de sentenças de linguagem natural. Porém, durante o desenvolvimento da pesquisa, algumas limitações e dificuldades relacionadas à independência de variáveis levaram aos desenvolvimentos apresentados aqui.

O problema da classificação morfossintática (*part-of-speech tagging*, em inglês) é uma questão clássica na área de Processamento de Linguagem Natural que serve de pré-requisito para muitas outras tarefas, como reconhecimento de voz, recuperação de informação e extração de estrutura sintática. Ele consiste em associar a cada palavra de uma sentença uma classe morfossintática. Classes morfossintáticas agrupam palavras com funções sintáticas e variações morfológicas semelhantes. Exemplos de classes morfossintáticas em gramáticas tradicionais são substantivo, verbo, adjetivo, advérbio, etc.

Para diferenciar as classes morfossintáticas, é preciso levar em conta o contexto em que cada palavra está inserida. Diversos métodos clássicos de resolução do problema da classificação morfossintática consideram apenas um contexto local limitado para evitar um número exponencial de variáveis ou restrições. Isso significa que esses métodos são incapazes de considerar a influência que palavras distantes podem ter. Por exemplo, considere as frases: “Que saudades eu sinto” e “Que saudades eu sinto todos já sabem”. Em cada uma dessas frases a palavra “que” se encaixa em uma classe morfossintática diferente, advérbio e conjunção, respectivamente. Porém, as palavras que modificam a classe da palavra “que” podem estar arbitrariamente afastadas dela (“Que saudades eu sinto me corroer por dentro... todos já sabem”), o que impossibilita que diversos métodos possam diferenciar corretamente as classes das palavras dessas frases.

O problema da Satisfazibilidade Probabilística, PSAT, não assume independência estatística de nenhuma de suas variáveis e por isso foi considerado adequado para modelar influências independentes da distância. Porém, não conter nenhuma informação sobre independência de variáveis também se mostrou prejudicial. Ainda que se apresente como uma vantagem em relação a outros métodos de classificação morfossintática, a flexibilidade oferecida pelo PSAT torna-se também uma desvantagem na medida em que dificulta a modelagem de variáveis estatisticamente independentes. Isto é, de certa forma, o contrário do

que acontece com outras ferramentas de modelagem logico-probabilística. O tratamento de variáveis dependentes, muitas vezes, é justamente o gargalo desses métodos.

Com essa motivação nos focamos em como introduzir relações de independência no problema PSAT. Propostas para permitir o uso de independência no PSAT já foram apresentadas, por exemplo, em Cozman *et al.* (2008). Nosso trabalho sugere um novo algoritmo, que, para casos especiais, é mais eficiente. Esses casos especiais surgem com naturalidade ao utilizarmos uma versão mais fraca da independência mútua, a independência parcial.

1.1 Objetivos

O objetivo desta tese é avançar o entendimento e o escopo do PSAT como ferramenta de modelagem e inferência logico-probabilística. A modelagem, bem como a extensão, de Modelos Ocultos de Markov como instância PSAT contribui para este entendimento não apenas de forma direta, mas por nos permitir perceber algumas de suas deficiências. Especificamente, a forma de tratamento da independência.

Tendo identificado essas limitações, desenvolvemos propostas de relaxamento do problema e um algoritmo para permitir o tratamento eficiente de independência pelo PSAT.

1.2 Organização do trabalho

Iniciamos apresentando, no capítulo 2, os conceitos necessários para entender tanto o problema que serviu de motivação à nossa investigação quanto as ferramentas que usaremos para propor métodos de tratamento da independência parcial com lógica probabilística. Alguns algoritmos importantes para a implementação de nossas propostas são apresentados neste capítulo, quais sejam o problema da Satisfazibilidade Probabilística, Modelos Ocultos de Markov e conceitos de otimização linear e não linear.

O capítulo 3 serve ao mesmo tempo de motivação e exemplo de aplicação para o que será desenvolvido no capítulo 4. Nele se discutem o relaxamento e a aplicação de Modelos Ocultos de Markov ao problema de etiquetagem morfossintática. Outro aspecto relevante do trabalho apresentado nesse capítulo é a técnica de modelagem utilizada para representar Modelos Ocultos de Markov como uma instância PSAT. São demonstrados os desafios e potencialidades do PSAT, que por si só podem levar a trabalhos futuros de modelagem e extensão de outros modelos estatísticos interessantes, com a vantagem de já contar com um algoritmo para encontrar soluções que satisfaçam estes modelos. Os principais resultados apresentados nesse capítulo também são descritos, de forma mais sintética, no artigo de Morais *et al.* (2015).

Finalmente, no capítulo 4 discutimos como estender o problema PSAT para utilizar independência, apresentamos trabalhos que já estudaram o fenômeno, como Cozman *et al.* (2008), e desenvolvemos um novo algoritmo que, em casos comuns o suficiente para serem considerados úteis, consegue encontrar modelos independentes de maneira muito mais eficiente.

Os resultados originais desse trabalho se encontram no capítulo 3 e no capítulo 4. Eles incluem um novo método de etiquetagem morfossintática, baseada em PSAT, e um novo algoritmo para encontrar distribuições de probabilidade com independência parcial.

Capítulo 2

Fundamentos

Aqui será feita uma exposição dos conceitos de: problema da Satisfazibilidade Probabilística, Modelos Ocultos de Markov e otimização linear e não linear. O entendimento dessas concepções torna viável a compreensão do problema que motivou a realização desta pesquisa e das ferramentas utilizadas para propor métodos de tratamento da independência parcial com lógica probabilística.

2.1 Lógica proposicional clássica

A linguagem da lógica proposicional clássica é formada por um conjunto enumerável de símbolos $X = \{x_1, \ldots x_k\}$, representando variáveis booleanas combinadas pelos conectivos lógicos binários $\wedge$ e $\vee$ e pelo conectivo unário $\neg$, possivelmente com parênteses para esclarecer a ordem das operações.

Uma *fórmula bem formada* da lógica proposicional, ou simplesmente *fórmula*, nesse alfabeto é definida pelas seguintes regras:

- Toda variável é uma fórmula;
- Seja α uma fórmula, $\neg\alpha$ também é uma fórmula;
- Sejam α e β fórmulas, $(\alpha \wedge \beta)$ e $(\alpha \vee \beta)$ também o são;
- Nada além dessas regras é uma fórmula.

As variáveis assumem um valor verdade, verdadeiro ou falso, representados como 1 e 0 respectivamente. Uma *valoração* é uma associação de um valor verdade para cada variável presente em uma fórmula. Portanto, em uma fórmula com n variáveis distintas existem 2^n possíveis valorações.

Dado uma valoração v para as variáveis de uma fórmula, podemos definir a valoração de uma fórmula considerando a semântica dos operadores lógicos.

- $v(\alpha \wedge \beta) = 1$ se, e somente se, $v(\alpha) = 1$ e $v(\beta) = 1$;
- $v(\alpha \vee \beta) = 1$ se, e somente se, $v(\alpha) = 1$ ou $v(\beta) = 1$;

- $v(\neg\alpha) = 1$ se, e somente se, $v(\alpha) = 0$.

Chamamos de *literal* uma variável (x_i) ou sua negação ($\neg x_i$). Uma disjunção ($\vee$) de literais é chamada de *cláusula*.

2.2 Lógica probabilística e o problema PSAT

O *Problema da Satisfazibilidade Probabilística*, *PSAT*, e a lógica probabilística em geral são bastante antigos e foram estudados inicialmente por Boole (1854) e redescobertos por diversos pesquisadores até a sua introdução na comunidade de Inteligência Artificial por Nilsson (1986).

Uma instância de um problema PSAT é uma associação entre fórmulas lógicas e probabilidades, descritas por um conjunto $\Gamma = \{P(\phi_i) = p_i | 1 \leq i \leq k\}$, onde ϕ_i é uma fórmula da lógica proposicional clássica sobre um conjunto de n variáveis e $p_i \in [0, 1]$. Por uma questão de tratabilidade computacional, podemos considerar apenas os valores de p_i racionais. O conjunto Γ é chamado *base de conhecimento probabilística*.

Uma *distribuição de probabilidades discreta* sobre um conjunto enumerável X é uma função que associa a cada elemento desse conjunto uma probabilidade de modo que a soma de todas as probabilidades seja 1.

$$\sum_{x \in X} P(x) = 1$$

Dada uma distribuição de probabilidades sobre todas as possíveis valorações das fórmulas do conjunto S, podemos definir a probabilidade de uma fórmula. A probabilidade de uma fórmula é a soma da probabilidade das valorações em que essa fórmula é verdadeira. Ou seja, se π for uma distribuição de probabilidade sobre o conjunto de valorações V_S, a probabilidade da fórmula ϕ é dada por:

$$P_\pi(\phi) = \sum_{v \in V_S} \{\pi(v) | v(\phi) = 1\} \tag{2.1}$$

Intuitivamente, a probabilidade de uma fórmula é a probabilidade de todos os "mundos possíveis" nos quais essa fórmula é verdadeira.

Dizemos que essa instância do problema PSAT é *satisfazível* se as associações $\{P(\phi_i) = p_i\}$ são *consistentes* para todos os elementos de Γ.

Definição 1. *Seja $V_S = \{v_1, \ldots, v_{2^n}\}$ o conjunto de valorações possíveis sobre as variáveis de S, dizemos que as associações de probabilidades $\{P(\phi_i) = p_i\}$ são **consistentes** se existir uma distribuição de probabilidades sobre V_S, chamada π, que satisfaça a Equação 2.1*

Exemplo 1. *Considere o problema PSAT:*

$$P(a \vee b \vee c) = 1$$
$$P(a \wedge b) = 0,61; P(a \wedge c) = 0,60; P(b \wedge c) = 0,59$$

Considere as valorações $v_1 = \{a = b = c = 1\}$, $v_2 = \{a = b = 1; c = 0\}$, $v_3 = \{a = c = 1; b = 0\}$ e $v_4 = \{a = 0; b = c = 1\}$.

A distribuição de probabilidades π que associa os seguintes valores a estas valorações: $\pi(v_1) = 0,4; \pi(v_2) = 0,21; \pi(v_3) = 0,2; \pi(v_4) = 0,19$; *e zero a todas as demais valorações, é uma solução para esse problema PSAT, como podemos ver substituindo esses valores nas equações abaixo, equivalentes à Equação 2.1 para cada fórmula:*

$$\begin{aligned} \pi(v_1) + \pi(v_2) + \pi(v_3) + \pi(v_4) &= 1 && (2.2)\\ \pi(v_1) + \pi(v_2) &= 0,61 && (2.3)\\ \pi(v_1) + \pi(v_3) &= 0,6 && (2.4)\\ \pi(v_1) + \pi(v_4) &= 0,59 && (2.5) \end{aligned}$$

Expressando essas equações na forma matricial, temos:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0,4 \\ 0,21 \\ 0,2 \\ 0,19 \end{bmatrix} = \begin{bmatrix} 1 \\ 0,61 \\ 0,60 \\ 0,59 \end{bmatrix} \quad (2.6)$$

O problema PSAT pode ser expresso como um problema de programação linear. Definimos a matriz $A_{k\times 2^n} = [a_{ij}]$, tal que $a_{ij} = 1$ se a valoração j satisfaz a fórmula i, caso contrário $a_{ij} = 0$, e definimos também o vetor $p_{k\times 1} = [p_i]$. Uma instância PSAT é satisfazível se há um vetor π que satisfaz:

$$\begin{aligned} A\pi &= p \\ \pi &\geq 0 \\ \sum \pi_i &= 1 \end{aligned} \quad (2.7)$$

A última restrição pode ser omitida adicionando a A e a p uma linha inteira de uns.

Segundo o Lema de Carathéodory (Prasolov e Tikhomirov, 2001), não precisamos de uma matriz A com tantas colunas, pois esse lema garante que, se esse problema tem solução, ele tem uma solução com apenas $k + 1$ elementos de π diferentes de zero. Então, podemos eliminar os elementos com zero da matriz e reduzir a matriz A a uma matriz $k + 1 \times k + 1$ (a matriz básica), tornando o problema PSAT um problema NP-completo.

Na forma de problema de programação linear, o exemplo 1 é expresso como a equação matricial 2.6.

2.2.1 Forma normal atômica

Uma forma normal permite padronizar a apresentação de instâncias do problema e facilitar seu estudo. Os trabalhos de Bona (2011) e Finger e De Bona (2010, 2011) apresentaram uma forma normal atômica à qual todas as instâncias PSAT podem ser convertidas em tempo polinomial adicionando no máximo um número linear de novas variáveis.

Uma instância PSAT na forma normal atômica pode ser particionada em dois conjuntos (Γ, Ψ), onde Γ é um conjunto de fórmulas proposicionais com probabilidade associada 1 e

Ψ possui probabilidades associadas apenas a variáveis (átomos).

$$\Gamma = \{P(s_i) = 1 | 1 \leq i \leq m\}$$
$$\Psi = \{P(y_i) = p_i | y_i \text{ é um átomo e } 1 \leq i \leq k\}$$

Essa forma transforma o PSAT em um problema de encontrar uma distribuição de probabilidades compatível com Ψ restrito a um problema SAT Γ.

Exemplo 2. *O problema do exemplo 1 pode ser escrito na forma normal atômica como:*

$$\Gamma = \left\{ \begin{array}{cccc} \neg x \vee a & \neg y \vee a & \neg z \vee b & a \vee b \vee c \\ \neg x \vee b & \neg y \vee c & \neg z \vee c & \\ \neg a \vee \neg b \vee x & \neg a \vee \neg c \vee y & \neg b \vee \neg c \vee z & \end{array} \right\}$$

$$\Psi = \{P(x) = 0.61; P(y) = 0.60; P(z) = 0.59\}$$

Para construirmos uma instância PSAT (Γ, Ψ), a partir de uma instância PSAT $\Delta = \{P(s_i) = p_i | 1 \leq i \leq k\}$, basta adicionar k novas variáveis, $y_1, \ldots, y_k$. Com isso, podemos criar o problema PSAT na forma normal atômica $\Gamma = \{P(y_i \leftrightarrow s_i) = 1 | 1 \leq i \leq k\}$, e $\Psi = \{P(y_i) = p_i | 1 \leq i \leq k\}$. Uma prova de que o novo problema é equivalente ao problema original Δ pode ser vista em de Bona (2011, Teorema 2.5.1).

2.2.2 Forma de otimização

Além da *forma de decisão* do problema PSAT, em que queremos saber se existe uma distribuição de probabilidade qualquer que torne a base de conhecimento probabilística consistente, podemos considerar a *forma de otimização* do PSAT ou *OPSAT* (Nilsson, 1986). Se considerarmos uma fórmula lógica diferente $\varphi \notin \Gamma$, normalmente as restrições 2.7 não impõem um único valor de probabilidade para essa fórmula[1] e sim um intervalo de probabilidades, cada uma obtida a partir de um diferente valor de π que satisfaz a Equação 2.7. O problema OPSAT consiste em encontrar a distribuição π que maximiza ou minimiza $P_\pi(\varphi)$, como em 2.8.

$$\begin{array}{rc} \mathsf{max\ /\ min} & P_\pi(\varphi) \\ \mathsf{sujeito\ a} & A\pi = p \\ & \sum \pi_i = 1 \\ & \pi \geq 0 \end{array} \tag{2.8}$$

O problema OPSAT é NP-difícil. Do ponto de vista da Programação Linear, as associações de probabilidade Γ definem uma região viável. O problema PSAT consiste em encontrar uma solução viável qualquer, de preferência uma solução básica viável em que apenas $k+1$ elementos são maiores que zero; e o problema OPSAT consiste em encontrar uma solução básica viável que otimize a equação linear 2.1.

[1] A menos que o vetor linha $A_\varphi = (a_{\varphi,j})$, em que $a_{\varphi,j} = 1$ se, e somente se, a valoração j satisfaz φ, seja uma combinação linear de outras linhas da matriz A.

2.2.3 Probabilidades condicionais

Desde Boole (1854) já se considerou o uso de probabilidades condicionais com o problema PSAT. Jaumard *et al.* (1991) mostrou como é possível resolver as versões condicionais do PSAT e do OPSAT em que probabilidades condicionais podem aparecer tanto na base de conhecimento probabilístico quanto na função objetivo do OPSAT.

No primeiro caso, a base de conhecimento é representada por $\Gamma = \{P(\phi_i|\psi_i) = p_i\}$, onde ϕ_i e ψ_j são fórmulas da lógica proposicional.

Podemos estender a definição de probabilidade de uma fórmula para considerar a *probabilidade condicional de uma fórmula* levando em conta a definição de probabilidade condicional, $P(a|b) = \frac{P(a \wedge b)}{P(b)}$. Desse modo, dada uma distribuição de probabilidade π, dizemos que:

$$P_\pi(\phi_i|\psi_i) = p_i \iff P_\pi(\phi_i \wedge \psi_i) = p_i P_\pi(\psi_i) \tag{2.9}$$

Essa expressão pode ser introduzida na forma matricial (2.7) com o uso de uma nova variável p_{ψ_i} e duas novas restrições:

$$\begin{aligned} A_{\phi_i \wedge \psi_i}\pi - p_i p_{\psi_i} &= 0 \\ A_{\psi_i}\pi - p_{\psi_i} &= 0 \end{aligned} \tag{2.10}$$

As matrizes-linha $A_{\phi_i \wedge \psi_i}$ e A_{ψ_i} são formadas analogamente à matriz A em 2.7, na qual um elemento na coluna j possui valor 1 se, e somente se, a valoração j satisfaz $(\phi_i \wedge \psi_i)$ e ψ_i respectivamente.

Problemas OPSAT cuja função objetivo é uma probabilidade condicional também podem ser resolvidos com o uso de novas restrições, como pode ser visto em Hansen e Jaumard (2000). Normalmente uma função objetivo fracional transformaria o problema em um problema de otimização hiperbólica.

$$\begin{aligned} \text{max / min} \quad & \tfrac{P_\pi(\phi \wedge \psi)}{P_\pi(\psi)} \\ \text{sujeito a} \quad & A\pi = p \\ & \textstyle\sum \pi_i = 1 \\ & \pi \geq 0 \end{aligned} \tag{2.11}$$

Porém, devido a um resultado de Charnes e Cooper (1962), podemos reduzir o problema 2.11 a um problema linear com uma variável extra t:

$$\begin{aligned} \text{max / min} \quad & P_\pi(\phi \wedge \psi) \\ \text{sujeito a} \quad & A_\psi \pi = 1 \\ & \textstyle\sum \pi_i = t \\ & A\pi = pt \\ & \pi \geq 0 \\ & t \geq 0 \end{aligned} \tag{2.12}$$

O problema 2.12 possui o mesmo valor ótimo de π que o problema 2.11, e para obter o

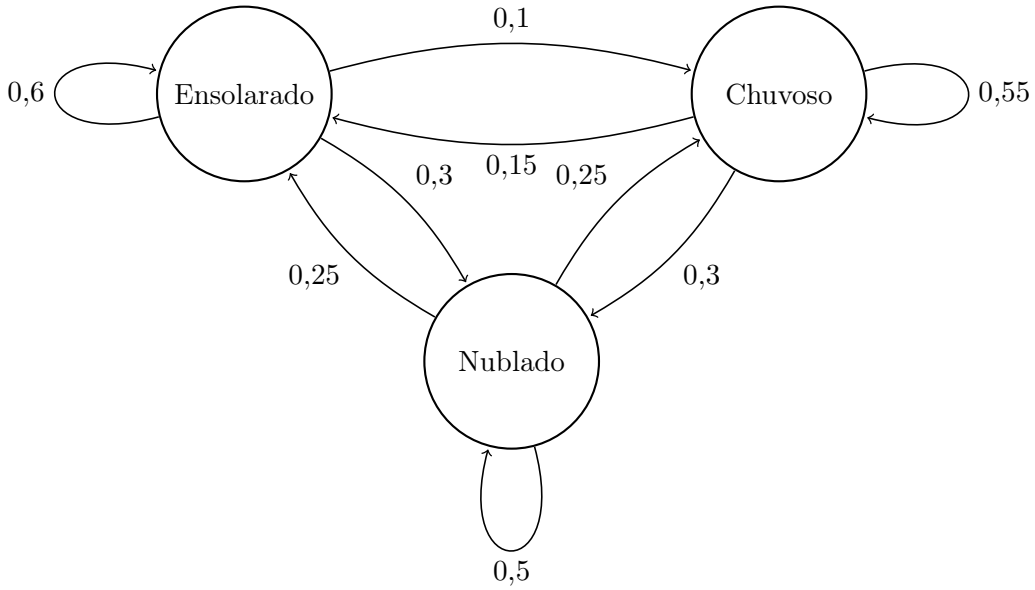

Figura 2.1: *Uma Cadeia de Markov*

valor ótimo da função otimizada basta dividi-lo pelo valor ótimo de t.

Esse problema também é NP-completo pelas mesmas razões do PSAT.

2.3 Modelos Ocultos de Markov

Modelos Ocultos de Markov (também chamados *HMMs*, *Hidden Markov Models*) são um processo estocástico cujos estados não podem ser observados diretamente (por isso é chamado *oculto*). O que é observado é uma sequência de sinais produzidos por outro processo estocástico dependente apenas no estado oculto atual.

A teoria dos HMMs foi desenvolvida no final dos anos 1960 (Baum *et al.*, 1970), baseada nas técnicas de Andrei Markov (Markov, 1913) usadas para modelar sequências de letras em trabalhos de literatura russa. Desde então foram utilizados em diversas tarefas de Processamento de Linguagem Natural e outras áreas.

Uma Cadeia de Markov (um modelo de Markov não oculto de primeira ordem) é um modelo de uma sequência de variáveis aleatórias $T = (t_1, \ldots, t_M)$ onde a distribuição de probabilidade de uma variável t_i depende apenas do valor da variável anterior t_{i-1} e essas probabilidades não variam com o tempo[2]. Por exemplo, considere a Cadeia de Markov representada na Figura 2.1. As arestas do gráfico indicam qual a probabilidade de transitar para um novo estado. Dado que hoje está ensolarado, podemos calcular a probabilidade de observar a sequência (*ensolarado*, *nublado*, *nublado*, *chuvoso*, *ensolarado*) simplesmente multiplicando as probabilidades de transição. Neste caso, $p(ensolarado|ensolarado) \times p(nublado|ensolarado) \times p(nublado|nublado) \times p(chuvoso|nublado) \times p(ensolarado|chuvoso) = 0{,}6 \times 0{,}3 \times 0{,}5 \times 0{,}25 \times 0{,}15 = 0{,}003375$.

[2] Por depender apenas da variável anterior, é dito que os estados de uma Cadeia de Markov "não tem memória"

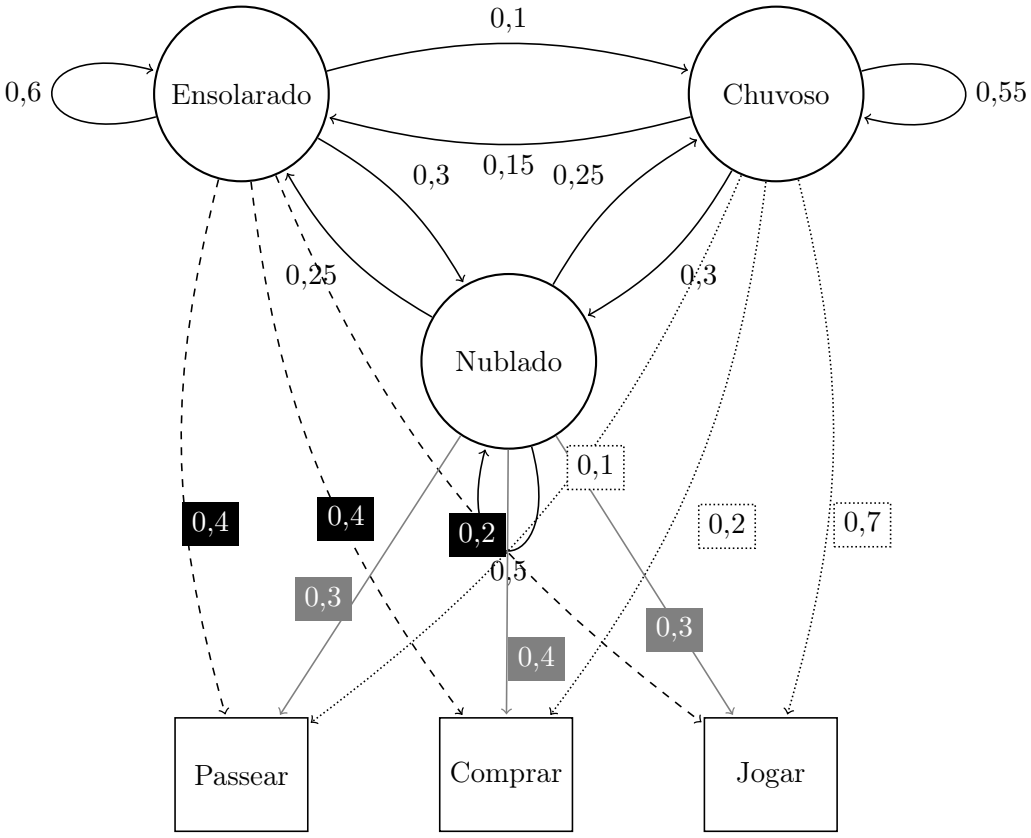

Figura 2.2: *Um Modelo Oculto de Markov*

Em um Modelo Oculto de Markov, os estados (ocultos) transitam da mesma maneira. Porém, não é possível observar diretamente os estados, e os símbolos observáveis são gerados aleatoriamente. Considere o exemplo da Figura 2.2. Nesse cenário não podemos observar diretamente o clima, mas podemos saber sobre atividades realizadas no dia, e a probabilidade de realizar cada atividade varia com o clima. Dessa maneira, uma sequência (*passear*, *comprar*, *jogar*) pode ser gerada por diferentes sequências de estado, mesmo havendo uma sequência *mais provável*.

2.3.1 Definição

Dado um conjunto de estados $S = \{s_1 \dots s_N\}$ e um de observações $O = \{o_1 \dots o_K\}$, considere a sequência de estados T e de observações W geradas por esses estados.

$$T = (t_1, \dots, t_M) \quad t_k \in S \tag{2.13}$$

$$W = (w_1, \dots, w_M) \quad w_k \in O \tag{2.14}$$

Seja $i, j \in [1, N]; k, \ell \in [1, M]; h \in [1, K]$. Assumimos que a probabilidade de transição de um estado não muda com o tempo, ou seja,

$$P(t_k = s_j | t_{k-1} = s_i) = P(t_\ell = s_j | t_{\ell-1} = s_i) \quad \forall k, \ell \tag{2.15}$$

Como esses valores são estacionários, podemos definir a matriz A, chamada *matriz de transição*, que guarda a probabilidade de que o estado s_j siga o estado s_i em T.

$$A = [a_{ij}], \quad a_{ij} = P(t_k = s_j | t_{k-1} = s_i) \tag{2.16}$$

Chamamos B a matriz de probabilidades de um estado s_i produzir uma observação o_h:

$$B = [b_{ih}], \quad b_{ih} = P(w_k = o_h | t_k = s_i) \tag{2.17}$$

Seja q a distribuição de probabilidade do estado inicial:

$$q = [q_i], \quad q_i = P(t_1 = s_i) \tag{2.18}$$

A definição formal de um Modelo Oculto de Markov (de primeira ordem) é

$$H = (A, B, q) \tag{2.19}$$

Duas suposições feitas por todo HMM são a *Propriedade de Markov* e a *Propriedade de Independência*.

A Propriedade de Markov, ou "falta de memória", afirma que o estado atual depende apenas do estado anterior.

$$P(t_k | t_{k-1}) = P(t_k | t_{k-1}, \ldots, t_1) \tag{2.20}$$

A Propriedade da Independência afirma que uma observação depende apenas do estado atual do modelo, ou seja, é independente de outras observações e estados passados.

$$P(w_k | t_k) = P(w_k | t_M, \ldots, t_1, w_{k-1}, \ldots, w_1) \tag{2.21}$$

2.3.2 Inferências em HMMs

Existem várias inferências que podem ser realizadas sobre Modelos Ocultos de Markov. Algumas das mais estudadas são:

- Calcular a probabilidade de uma sequência de observações (calcular $P(W|H)$).
- Descobrir a sequência de estados que mais provavelmente produziu uma sequência de observações ($\max_T P(W|H,T)$).
- Aprender os parâmetros de um modelo $H = (A, B, q)$ para melhor adaptá-lo a um conjunto de exemplos.

Vamos nos focar no segundo problema, achar a explicação mais provável para um conjunto de observações, pois é o problema utilizado no capítulo 3. Esse problema também é chamado *decodificação* (Blunsom, 2004). Um algoritmo eficiente para esse problema é o algoritmo de Viterbi (Viterbi, 1967).

O algoritmo de Viterbi é um algoritmo de programação dinâmica que, em cada passo ℓ, calcula:

$$\delta_\ell(i) = \max_{t_1 \ldots t_{\ell-1}} P(t_1, \ldots, t_{\ell-1}, t_\ell = s_i, w_1, \ldots, w_\ell | H) \tag{2.22}$$

$$= \max_j \delta_{\ell-1}(j) a_{ji} b_{ih}, \text{ onde } w_\ell = o_h \tag{2.23}$$

A variável $\delta_\ell(i)$ indica a probabilidade da sequência de estados mais provável para uma sequência de observações parcial $(w_1, \ldots, w_\ell)$ que termina no estado s_i. Essa variável pode ser computada eficientemente pela Equação 2.23 considerando todas as variáveis $\delta_{\ell-1}$, uma vez que a probabilidade de atingir um estado qualquer no momento ℓ depende apenas do estado no momento $\ell - 1$.

A inicialização do algoritmo é dada por $\delta_1(i) = q_i b_{ih}$ para $w_1 = o_h$. A resposta final do problema é dada por $\max_i \delta_M(i)$. É necessário salvar ponteiros para os estados escolhidos na Equação 2.23 para poder reconstruir a sequência de estados após calcular o maior δ_M.

O algoritmo de Viterbi faz grande uso das Propriedades de Markov e de Independência para poder calcular eficientemente cada δ_ℓ.

2.3.3 HMMs de n-ésima ordem

A Propriedade de Markov pode ser relaxada para que um estado dependa de dois ou mais estados anteriores ao invés de apenas um. Nesse caso a Equação 2.20 se torna:

$$P(t_k | t_{k-1}, \ldots, t_{k-n}) = P(t_k | t_{k-1}, \ldots, t_1) \tag{2.24}$$

Esse modelo é chamado *HMM de n-ésima ordem*. É possível expressar um modelo de n-ésima ordem com um HMM de primeira ordem. Basta considerar estados compostos, formados pela composição de n estados do HMM de ordem maior, e modificar a matriz de transição apropriadamente.

Portanto, todas as propriedades e algoritmos que se aplicam a HMMs de primeira ordem também podem ser aplicados a HMMs de n-ésima ordem. Porém, existe um crescimento exponencial no tamanho das matrizes necessárias para expressar o HMM e, consequentemente, no tempo necessário para realizar inferências.

Uma variação de HMMs de ordem alta são as *Cadeias de Markov de Tamanho Variável* (VLMC) (Bühlmann e Wyner, 1999; Mächler e Bühlmann, 2012). Nesse modelo, o número de estados levados em consideração para cada transição de estado varia de acordo com os estados anteriores. Mesmo as VLMC têm um tamanho máximo para os contextos, uma ordem k, porém, normalmente o contexto usado é menor que k.

2.4 Otimização não linear

Problemas cuja função objetivo ou cujas restrições não são lineares aparecem naturalmente em diversas áreas do conhecimento, como modelos econômicos, ecológicos, industriais

e físicos (Bubeck, 2015; Hendrix e G.-Tóth, 2010). Esses problemas não podem ser resolvidos pelo algoritmo Simplex ou similares. Se a região viável ou função objetivo do problema não for *convexa*, além de não linear estamos tratando de um problema de otimização não convexa. Problemas de otimização não convexa devem se preocupar com mínimos locais, como veremos no Teorema 2.

Definição 2. *Seja $X \in \mathbb{R}^n$ um conjunto de pontos, uma* corda *entre um par de pontos $p, q \in X$ é o conjunto de pontos:*

$$\text{CORDA}(p, q) = \{x | x = \lambda p + (1 - \lambda)q, \lambda \in [0, 1]\}$$

Definição 3. *Um conjunto X é dito um* conjunto convexo *se para qualquer par de pontos $p, q \in X$, uma corda entre esses dois pontos também está em X. Ou seja:*

$$\lambda p + (1 - \lambda)q \in X, \forall \lambda \in [0, 1]$$

Definição 4. *Uma função $f : X \to \mathbb{R}$, sendo X um conjunto convexo não vazio, é dita* convexa *se*

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2) \quad \forall x_1, x_2 \in X, \forall \lambda \in [0, 1]$$

Uma função definida sobre o mesmo domínio é dita côncava *se*

$$f(\lambda x_1 + (1 - \lambda)x_2) \geq \lambda f(x_1) + (1 - \lambda)f(x_2) \quad \forall x_1, x_2 \in X, \forall \lambda \in [0, 1]$$

Os rótulos de côncava e convexa não são mutualmente exclusivos. Uma função linear é tanto côncava quanto convexa. Funções que não atendem nenhuma das duas definições são chamadas não convexas e não côncavas.

Se uma função $f(x)$ é convexa, $-f(x)$ é côncava e vice-versa.

Definição 5. *Um problema de otimização é dito* convexo *se a função objetivo é convexa, no caso da minimização, ou* côncava *no caso da maximização, e a região viável do problema é convexa.*

O Teorema 1 a seguir nos permite saber quando a região viável de um problema de otimização é convexa.

Teorema 1 (Teorema 3.9 de Hendrix e G.-Tóth (2010))**.** *Seja X um conjunto convexo, $g : X \to \mathbb{R}$ uma função convexa e $h \in \mathbb{R}$, o conjunto $X_h = \{x \in X | g(x) \leq h\}$ é convexo.*

Com esse teorema sabemos que problemas com funções objetivos e desigualdades convexas são convexos. No caso de igualdades, apenas funções lineares podem ser usadas se queremos um problema convexo, pois uma igualdade $g(x) = h$ pode ser transformada em $g(x) \leq h$ e $-g(x) \leq h$. Como funções lineares são tanto côncavas como convexas, sabemos que $-g(x)$ também é convexo e, portanto, ambas as desigualdades são convexas.

Porém, demais igualdades, mesmo que convexas, não se aplicam nesse teorema e podem levar a problemas de otimização não convexos.

Definição 6. *Um ponto x^* é dito* mínimo local *de uma função f se existe ϵ tal que para todo x onde $\|x - x^*\| < \epsilon$, $f(x) \geq f(x^*)$.*

Definição 7. *Um ponto x^* é dito* mínimo global *de uma função f se $f(x^*) \leq f(x) \quad \forall x$.*

A grande utilidade de se resolver problemas convexos é demonstrada no teorema a seguir.

Teorema 2 (Teorema 3.10 Hendrix e G.-Tóth (2010))**.** *Seja f uma função convexa em um conjunto convexo X, então todo mínimo local é um mínimo global.*

O Teorema 2 pode ser provado notando que, se existe um x' tal que $f(x') < f(x^*)$, então, por f ser convexa, todos os pontos entre x' e x^* são menores ou iguais à corda ligando x' e x^*, $f(\lambda x' + (1-\lambda)x^*) \leq \lambda f(x') + (1-\lambda)f(x^*) \quad \forall \lambda \in [0,1]$. No entanto, isso é uma contradição com a definição de mínimo local e a hipótese de que $f(x') < f(x^*)$. A convexidade da região viável é necessária para que possa existir uma corda entre x' e x^*.

Esse teorema facilita muito a busca por mínimos globais, pois podemos ter uma atitude "gulosa" na otimização, sempre buscando uma direção que minimiza a função objetivo, e, se em algum momento não houver mais direção que não aumente a função, podemos concluir que encontramos um mínimo global.

2.4.1 Otimização não convexa

Apesar de serem mais difíceis de resolver devido à falta de garantias discutidas na seção anterior, problemas não convexos aparecem muitas vezes na prática. Um exemplo de problema não convexo é a programação inteira.

Um dos métodos mais utilizados para otimizar problemas não convexos é o *Branch and Bound*.

Branch and Bound

Proposto por Land e Doig (1960) para problemas de programação discreta, *Branch and Bound* (*B&B*) se refere a um conjunto de métodos e algoritmos que explora toda a região viável do espaço de soluções. Combinados com boas heurísticas de estimação do valor de funções, principalmente de limites inferiores, esses algoritmos têm se mostrado muito eficientes na prática. O Algoritmo 1 expressa em termos gerais o método *B&B*.

Sem perda de generalidade, vamos assumir que queremos minimizar a função objetivo. Algoritmos *B&B* consistem em repartir recursivamente a região viável, ou uma região que contenha a região viável, em subconjuntos, preferencialmente convexos e disjuntos, mas não necessariamente, formando uma árvore. Esse passo é conhecido com *Branch*.

Antes de visitar cada subárvore procurando pela melhor solução, estimamos um limite inferior para o menor valor possível da função objetivo nesta região, f_k^L, e um limite superior para o valor ótimo, f^U. Esse passo é o *Bound*. Se o mínimo estimado para uma região for

maior que f^U, $F_k^L > f^U$, descartamos essa região. Desse modo evitamos visitar regiões que sabemos não conter o valor ótimo. Regiões também podem ser descartadas se não contêm solução viável.

Recursivamente, fazemos o *Branch* de novo e repetimos o procedimento até que não existam mais regiões para se visitar, quando finalmente selecionamos o valor mínimo encontrado nas regiões visitadas. A estratégia de seleção da ordem de análise das regiões, como por fila, pilha ou priorizando aqueles com menor limite inferior, pode influenciar a performance do algoritmo.

No pior caso, se não pudermos eliminar nenhuma região, o algoritmo se resume a uma busca exaustiva por toda a região viável. Dependendo da estratégia de *Branch* e a ordem de visitação de cada nó da árvore, isso pode levar a um número infinito de regiões a explorar. Se definirmos um limite de precisão ϵ, é possível garantir a parada do algoritmo, pois podemos descartar regiões menores que ϵ, e a cada repartição a região fica menor.

Porém, se conseguirmos obter boas estimações do limite inferior da função objetivo em uma região, podemos reduzir muito o espaço de busca e encontrar a solução ótima muito mais rapidamente.

Técnica de Reformulação e Linearização

Obter boas estimativas é fundamental para obter boa eficiência em algoritmos *Branch and Bound*. Em geral, essas estimativas devem ser criadas caso a caso, específicas para cada problema não convexo que queremos resolver. A *Técnica de Reformulação e Linearização* (*Reformulation Linearization Technique* (*RLT*), no original) é uma técnica que ajuda a guiar a criação de boas estimativas para problemas não convexos discretos e contínuos.

Apresentada em um conjunto de trabalhos, Sherali e Adams (1999, 1994a,b), a técnica consiste em duas fases. Primeiro se faz uma *reformulação* do problema adicionando explicitamente restrições que no problema original aparecem apenas implicitamente como resultado da combinação de outras restrições. Esse novo problema é equivalente ao original. Então se faz a *linearização* (ou possivelmente "convexão") do problema reformulado que substitui todos os termos não lineares/não convexos por novas variáveis, tornando-os lineares. O valor mínimo da função objetivo do novo problema linear é garantidamente menor ou igual ao valor mínimo da função objetivo original, e as restrições novas adicionadas na primeira fase deixam de ser simplesmente uma combinação das demais e passam a garantir que a estimativa seja a mais próxima possível do mínimo global procurado.

Considere, por exemplo, o caso de problemas polinomiais com expoentes inteiros (Sherali e Adams, 1999), também chamados *programas multilineares*, descritos em 2.25. Esta família de pro-

Entrada: Função objetivo f, região viável X, tolerância ϵ
Saída: x^* que está no máximo ϵ distante do valor ótimo da função, ou NULO caso não exista solução

Encontre um conjunto inicial C_1 tal que $X \subseteq C_1$
Encontre um limite inferior f_1^L de C_1
Encontre um ponto viável $x_1 \in C_1 \cap X$ qualquer
se *não existe ponto viável* **então retorna** *NULO*
$f^U \leftarrow f(x_1)$
$\Lambda \leftarrow \{C_1\}$
$k \leftarrow 1$
$fx^* \leftarrow \infty$
$x^* \leftarrow$ NULO
enquanto $\Lambda \neq \emptyset$ **faça**
    Escolhe um C de Λ e o divide nos subconjuntos $C_{k+1}, \ldots, C_{k+h}$
    Encontra os limites inferiores $f_{k+1}^L, \ldots, f_{k+h}^L$
    para cada p *entre* $k+1$ *e* $k+h$ **faça**
        se $C_p \cap X$ *não contém pontos viáveis ou* $f_p^L > f^U$ **então**
            $\Lambda \leftarrow \Lambda \backslash C_p$
        senão
            Encontra um ponto viável x_p
            se $f(x_p) < f^U$ **então**
                $f^U \leftarrow f(x_p)$
                Remove todos os C_l de Λ tal que $f_l^L > f^U$
            fim
            se $f(x_p) < fx^*$ **então**
                $fx^* \leftarrow f(x_p)$
                $x^* \leftarrow x_p$
            fim
            se `Tamanho`$(C_p) \geq \epsilon$ **então** $\Lambda \leftarrow \Lambda \cup \{C_p\}$
        fim
        $k \leftarrow k + h$
    fim
fim
retorna x^*

Algoritmo 1: Algoritmo *Branch and Bound* em linhas gerais (baseado em Hendrix e G.-Tóth (2010, Algoritmo 50))

blemas será útil para nós no capítulo 4.

$$\begin{aligned} &\text{minimize} && \phi_0(x) \\ &\text{sujeito a} && \phi_r(x) = \beta_r \text{ para } r = 1, \ldots, R \\ & && 0 \leq l_j \leq x_j \leq u_j < \infty \text{ para } j = 1, \ldots, n \\ &\text{onde} && \phi_r(x) = \sum_{t \in T_r} \alpha_{rt} \left[\prod_{j \in J_{rt}} x_j \right] \text{ para } r = 1, \ldots, R \end{aligned} \tag{2.25}$$

Cada ϕ_r é uma função polinomial de grau no máximo δ. O conjunto T_r é um conjunto de índices dos termos de ϕ_r, e α_{rt} são coeficientes dos termos $(\prod_{j \in J_{rt}} x_j)$.

Os conjuntos J_{rt} são conjuntos dos índices dos termos do monômio $(\prod_{j \in J_{rt}} x_j)$ e permitem repetição. Por exemplo, para $J_{rt} = \{1, 2, 2, 3\}$, temos o monômio $x_1 x_2^2 x_3$. Formalmente, seja $N = \{1, \ldots, n\}$ e $N^\delta = \{\underbrace{N, \ldots, N}_{\delta \text{ vezes}}\}$, cada $J_{rt} \subseteq N^\delta$, com $1 \leq |J_{rt}| \leq \delta$, para $t \in T_r$ e $r = 1, \ldots, R$.

Para resolver um problema Ω da forma 2.25, podemos usar um algoritmo de *Branch and Bound* e construímos as estimativas utilizando *RLT*, como veremos a seguir.

Observando que $(x_j - l_j) \geq 0$ e $(u_j - x_j) \geq 0, \forall j \in N$, podemos escrever as restrições 2.26.

$$\prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j) \geq 0 \quad J_1 \cup J_2 \subseteq N^\delta \text{ e } |J_1 \cup J_2| = \delta \tag{2.26}$$

Note-se que as restrições 2.26 são sempre verdadeiras na região viável do problema original e o problema Ω', formado com a adição de restrições da forma 2.26 a Ω, é equivalente a Ω. Essa é a fase de *reformulação*.

As restrições 2.26 são apenas algumas das possíveis restrições que podem ser adicionadas na fase de reformulação. Outro exemplo seria multiplicar todas as restrições $\phi_r(x) = \beta_r$ por produtos $\prod_{j \in J_1} x_j$, $J_1 \subseteq N^\delta$. Quanto mais restrições adicionadas na fase de reformulação, mais próximas serão as estimativas, porém maiores serão os problemas gerados. Apenas para representar todas as restrições do tipo 2.26 são necessárias $\sum_{k=0}^{\delta} \binom{n+k-1}{k} \binom{n+(\delta-k)-1}{(\delta-k)}$ restrições.

Para a fase de *linearização*, definimos o problema Ω^L substituindo em Ω

$$X_J = \prod_{j \in J} x_j \quad \forall J \subseteq N^\delta, |J| \geq 2 \tag{2.27}$$

Assumimos que J é ordenado de forma não decrescente para evitar repetições.

É trivial perceber que o problema Ω^L é linear. Qualquer termo não linear foi substituído por uma das novas variáveis X_J. Se tivéssemos feito o passo de linearização sem a reformulação, as variáveis novas poderiam receber qualquer valor, mas as restrições 2.26 limitam os valores que X_J pode receber ao intervalo de valores que o produto das variáveis x_j, $j \in J$ pode assumir no problema original.

Teorema 3 (Lema 7.1 de Sherali e Adams (1999))**.** *Seja $v[\cdot]$ o valor ótimo de um problema, $v[\Omega^L] \leq v[\Omega]$.*

Além disso, se uma solução ótima (x^*, X^*) do problema Ω^L satisfaz 2.27 $\forall J$, então x^* é a solução ótima de Ω.

Para ver que esse teorema é verdadeiro, basta notar que toda solução viável de Ω é uma solução viável de Ω^L. A região viável de Ω está contida na região viável de Ω^L. Além disso, se 2.27 é satisfeito, os valores das funções objetivo de Ω^L e Ω são iguais.

Com esse teorema sabemos que o problema Ω^L pode ser usado como estimação do limite inferior de uma região no algoritmo *BnB*. Além disso, se (x^*, X^*) de alguma região satisfaz 2.27 obtemos um limite superior para o valor $v[\Omega]$ e podemos eliminar qualquer região Ω' tal que $v[\Omega'^L] \geq x^*$.

Para fazer o *Branch*, escolhemos uma variável x_p segundo a Equação 2.28, ou seja, o x_p com maior diferença entre o valor de alguma nova variável $X_{p\cup J}$ e o produto associado $x_p \prod_{j\in J} x_j$, e dividimos a região atual em duas, com valores de x_p maiores e menores, respectivamente, ao valor x_p^* encontrado na última iteração.

$$p \in \text{argmax}_{j\in N} \left\{ \max_{t=1,\ldots,\delta-1} \quad \max_{J\subseteq N^\delta, |J|=t} \left\{ |\hat{X}_{J\cup j} - \hat{x}_j \hat{X}_J| \right\} \right\} \tag{2.28}$$

Na Equação 2.28, $(\hat{x}, \hat{X})$ são os valores ótimos encontrados na região atualmente analisada. Utilizando os estimadores *RLT* desenvolvidos nesta seção, o Algoritmo 1 pode ser reescrito com o Algoritmo 2.

O teorema 7.1 de Sherali e Adams (1999) garante que, se rodando com $\epsilon = 0$, o Algoritmo 2 ou para em tempo finito com a solução ótima de Ω, ou gera uma sequência infinita de regiões tal que x^* tende à solução ótima.

Entrada: Problema Ω, tolerância ϵ
Saída: x^* que está no máximo ϵ distante do valor ótimo da função, ou NULO caso não exista solução

$\Lambda \leftarrow \{1\}$
$fx^* \leftarrow \infty$
$f^U \leftarrow \infty$
$x^* \leftarrow$ NULO
$\Omega_1 \leftarrow \Omega$
$\Omega_1^L \leftarrow \texttt{RLT}(\Omega_1)$ `/* RLT cria o problema linear` Ω^L `*/`
$(\hat{x}, \hat{X}) \leftarrow \texttt{Simplex}(\Omega_1^L)$
$f_1^L \leftarrow v[\Omega_1^L]$
se $\hat{x}$ *respeita 2.27* **então retorna** $\hat{x}$
enquanto $\Lambda \neq \emptyset$ **faça**
  Escolhe um C de Λ com menor f_C^L
  Seleciona a variável x_p segundo a Equação 2.28
  Particiona C em duas regiões: C_1, onde $x_p \leq \hat{x}$, e C_2, onde $x_p \geq \hat{x}$
  para cada p *entre* 1 *e* 2 **faça**
    $C_p^L \leftarrow \texttt{RLT}(C_p)$
    $(\hat{x}_p, \hat{X}_p) \leftarrow \texttt{Simplex}(C_p^L)$
    $f_p^L \leftarrow v[C_p^L]$
    se $f_p^L > f^U$ **então**
      $\Lambda \leftarrow \Lambda \backslash C_p$
    senão
      se $\hat{x}_p$ *respeita 2.27* **então**
        se $f(\hat{x}_p) < f^U$ **então**
          $f^U \leftarrow f(\hat{x}_p)$
          Remove todos os C_l de Λ tal que $f_l^L > f^U$
        fim
        se $f(\hat{x}_p) < fx^*$ **então**
          $fx^* \leftarrow f(x_p)$
          $x^* \leftarrow x_p$
        fim
        se $f_p^L + \epsilon < fx^*$ **então** $\Lambda \leftarrow \Lambda \cup \{C_p\}$
      fim
    fim
  fim
fim
retorna x^*

Algoritmo 2: Algoritmo *Branch and Bound* com estimadores RLT

Capítulo 3

Etiquetagem morfossintática via PSAT

Métodos estatísticos em Processamento de Linguagem Natural (e outros problemas resolvidos com aprendizagem computacional) normalmente consideram apenas um contexto local limitado para evitar um número exponencial de variáveis ou restrições.

Apesar dessa limitação, os algoritmos da área conseguem obter ótimos resultados, chegando a mais de 97% de acurácia por palavra e 56% de acurácia por sentença no caso da etiquetagem morfossintática (Manning, 2011). Porém, sua eficácia tende a atingir um limite, pois, apesar de muitas vezes as dependências locais serem predominantes, alguns fenômenos linguísticos só podem ser capturados considerando-se contexto de distâncias ilimitadas.

Neste capítulo, cujos principais resultados foram publicados em de Morais *et al.* (2015), exploraremos meios de tratar contexto ilimitado em problemas de Processamento de Linguagem Natural com uma modelagem que utiliza lógica probabilística e desenvolveremos mecanismos de inferência sobre esses modelos.

PSAT, ao contrário de modelos markovianos normalmente usados, é livre de hipóteses de independência entre variáveis e, apesar de o processo de decisão da lógica probabilística proposicional ser NP-completo, na prática muitas instâncias podem ser resolvidas de forma relativamente rápida. Essas características sugerem que a lógica probabilística pode ser uma boa ferramenta para a tarefa.

Em particular, trabalharemos com o problema de classificação morfossintática (*part-of-speech tagging*) utilizando modelos baseados em Modelos Ocultos de Markov (Baum *et al.*, 1970). Estudar as limitações do PSAT ao ser aplicado na modelagem de Modelos Ocultos de Markov nos levará ao estudo de modelos com independência parcial no capítulo 4.

Na literatura, a classificação em classes gramaticais é chamada *etiquetagem morfossintática* ou *part-of-speech tagging* (*POS tagging*) (Jurafsky e Martin, 2000) e será discutida com mais detalhes na próxima seção.

3.1 Classes de palavras e etiquetagem morfossintática

Palavras de uma sentença são comumente agrupadas em classes de equivalência chamadas *classes gramaticais*, *classes morfossintáticas* ou, em inglês, *part-of-speech*. Essas classes dão informação quanto a uma palavra e seu contexto, esclarecendo o seu significado e papel na frase. Por exemplo, saber se, em uma sentença, a palavra "busca" aparece como substantivo (como em "A busca pela verdade") ou como verbo ("Ele busca suas chaves") dá dicas quanto à provável classe das próximas palavras e seu significado na frase.

Normalmente as classes gramaticais agrupam palavras com funções sintáticas e variações morfológicas semelhantes, mas os critérios específicos usados variam entre diferentes modelos de linguagem e gramática. A classificação mais tradicional para o português brasileiro (Mesquita e Martos, 1994) divide as palavras em dez classes: substantivo, verbo, artigo, adjetivo, numeral, pronome, advérbio, preposição, conjunção e interjeição. Porém, essa classificação é criticada por alguns (Bagno, 2011), e modelos mais especializados para propósitos computacionais podem possuir um número bem diferente de classes e variações. O Corpus Histórico do Português Tycho Brahe tem 377 classes contando com todas as variações de tempo verbal, gênero e número.

O pertencimento a uma classe pode ser uma propriedade ambígua. Não só as mesmas palavras podem pertencer a diferentes classes em diferentes contextos, contexto que às vezes depende de outras sentenças, como às vezes os critérios de classe podem ser ambíguos e pouco determinados, variando entre modelos. Bagno (2011), por exemplo, critica o critério tradicional para formação da classe de pronomes como sendo inconsistente.

Em Processamento de Linguagem Natural, a informação apresentada pelas classes morfossintáticas ajuda em tarefas de reconhecimento de voz, recuperação de informação e extração de estrutura sintática (Jurafsky e Martin, 2000).

3.1.1 Etiquetagem morfossintática

A *etiquetagem morfossintática* ou *part-of-speech tagging* é o processo de associar palavras de uma sentença a classes morfossintáticas. No contexto do processo de etiquetagem, é comum se referir às classes associadas como *etiquetas* ou *tags*. Por uma questão prática, normalmente também são associadas etiquetas à pontuação de um frase. A etiquetagem morfossintática pode ser considerada para as linguagens naturais como o análogo da *tokenização* para linguagens de programação.

Muitas palavras não apresentam ambiguidades. Essas palavras sempre, ou quase sempre, pertencem à mesma classe gramatical. Segundo Charniak (1996), ao associar apenas as classes mais comuns para cada palavra, conseguimos uma acurácia de até 90%. Porém, isso não é verdade para muitas das palavras, mesmo as mais comuns. Para resolver essas ambiguidades, os vários algoritmos utilizam diversas estratégias.

Os primeiros esforços em problemas de PLN se baseavam em regras e gramáticas criadas por especialistas em uma linguagem natural (Harris, 1964; Karlsson *et al.*, 1995; Klein e Simmons, 1963). Apesar de obter bons resultados para alguns textos, essas abordagens encontraram

problemas para ser generalizadas para grande número de textos com diferentes estilos e formas. O grande avanço obtido com métodos estocásticos levou a uma mudança de foco para abordagens baseadas em aprendizagem computacional.

As abordagens baseadas em aprendizagem contam com um *corpus* de textos anteriormente classificados e analisados por especialistas humanos e, a partir desses textos, retiram estatísticas e regras para tratar outros textos nunca vistos. Essas abordagens obtiveram grande sucesso, chegando a mais de 97% de acurácia por palavra e 56% de acurácia por sentença no caso da etiquetagem morfossintática (Manning, 2011).

3.1.2 HMMs aplicados à etiquetagem

Um dos métodos que obtiveram mais sucesso na tarefa de etiquetagem morfossintática é o *Modelo Oculto de Markov* (ver seção 2.3).

HMMs foram usados em tarefas de Processamento de Linguagem desde os anos 1970 (Baker, 1975) com aplicações em reconhecimento de fala e extração de informação, além, é claro, de etiquetagem morfossintática (Jurafsky e Martin, 2000).

No contexto da etiquetagem morfossintática, as etiquetas são representadas como os estados de um HMM, e as palavras como as observações. As probabilidades de transição de estados e de geração de palavras por estado (matrizes A e B respectivamente) são treinadas por simples contagem (possivelmente com suavização) a partir de um *corpus* de treinamento previamente etiquetado. Normalmente são utilizados HMMs de segunda ou terceira ordem (Kepler, 2010).

Esse modelo sugere um *modelo gerativista* para a linguagem, em que primeiro a classe de uma palavra é selecionada e depois as palavras dessa classe são escolhidas aleatoriamente sem levar em consideração as demais palavras (Propriedade da Independência). Além disso, são levadas em conta apenas as duas ou três etiquetas vizinhas para se transitar para outra etiqueta (Propriedade de Markov).

Apesar dessas hipóteses pouco razoáveis, na prática, etiquetadores baseados em HMM possuem acurácia superior a 96% (Jurafsky e Martin, 2000) em alguns casos.

3.2 Etiquetagem via PSAT

Considere, por exemplo, a sentença em inglês "*The horse raced past the barn*". Nessa sentença a palavra "*raced*" aparece como um verbo no passado (`VB-P`). Se adicionarmos uma palavra ao final da sentença, "*The horse raced past the barn fell*", o verbo se torna em verbo no particípio passado (`VB-PP`). Porém, as palavras "*raced*" e "*fell*" podem estar arbitrariamente distantes, por exemplo "*The horse raced past the old red barn fell*".

No caso do português, esse tipo de construção subordinativa é marcada pela palavra "que". Segundo Kepler (2010), uma das palavras mais comumente confundidas por etiquetadores morfossintáticos do português é a palavra "que". Ela também pode ser influenciada por palavras arbitrariamente distantes, por exemplo, se "que" seguir "mais" indica que "que"

é uma conjunção subordinativa. Porém, essas duas palavras podem estar bem distantes, como em "mais caro, complexo e avançado que".

Não está claro qual o impacto que mais contexto pode ter na acurácia da etiquetagem. Manning (2011) fez uma análise de uma amostragem de erros de classificação do estado da arte de *POS tagging* trabalhando sobre um *corpus* anotado do *Wall Street Journal* do repositório Penn Treebank[1]. Segundo sua análise, 9% dos erros se devem a palavras desconhecidas ou com uma etiqueta que não aparece no treinamento, $55,5\%$ são devidos a problemas na classificação feita por especialistas no *corpus* de treinamento[2], e, finalmente, $35,5\%$ dos erros poderiam, possivelmente, ser corrigidos com melhores etiquetadores que consideram mais contexto, tanto contexto de palavras quanto contexto entre frases e conhecimento semântico.

Mesmo tendo em vista as limitações dessa análise, como considerar uma amostragem relativamente pequena, de apenas 100 amostras, e o uso de apenas um *corpus* específico, $35,5\%$ pode ser considerado como uma primeira estimativa para o limite superior da melhora que pode ser esperada desta técnica no melhor dos casos.

Apresentaremos a seguir, um método para realizar etiquetagem morfossintática com dependências de longa distância.

3.3 Modelando HMMs com lógica probabilística

O seguinte método foi desenvolvido em conjunto com Glauber de Bona e Marcelo Finger, e publicado em de Morais *et al.* (2015).

Conforme discutido na seção 2.3, Modelos Ocultos de Markov (HMMs) são usados há muito tempo para a etiquetagem morfossintática. Nesse contexto, um HMM associa uma única probabilidade a cada elemento de um domínio formado pela sequência de palavras e de *tags*.

Definindo variáveis proposicionais para indicar que uma certa palavra/*tag* está em uma posição, podemos definir logicamente esse domínio, e a probabilidade associada pelo HMM pode ser definida como um conjunto de associações de probabilidades a fórmulas lógicas, ou seja, como uma base de conhecimento probabilística.

Uma vez modelado dessa maneira, para encontrar o conjunto de *tags* mais provável para um conjunto de palavras (o problemas de *decodificação* do HMM), basta encontrar o máximo *a posteriori* das conclusões da base de conhecimento, reduzindo o problema a um problema OPSAT.

Na subseção 3.3.1 veremos como modelar um HMM em uma base de conhecimento probabilística. Modificando o modelo lógico-probabilístico de HMMs, podemos remover a hipótese de Markov e permitir que seja considerado todo o contexto. Isso será visto na subseção 3.3.2.

[1] https://www.cis.upenn.edu/~treebank/

[2] Esses problemas se devem a etiquetas ambíguas, falta de padronização na etiquetagem e erros humanos.

3.3.1 Modelando HMMs

Considere um conjunto de estados $S = \{s_1 \dots s_N\}$, um de observações $O = \{o_1 \dots o_K\}$ e um HMM de primeira ordem $H = (A, B, q)$, tal como definido na seção 2.3. Dada uma sequência de palavras $W = (w_1, \dots, w_M)$, podemos calcular a probabilidade de que ela tenha sido gerada pela sequência de *tags* $T = (t_1, \dots, t_M)$ no HMM H. Nosso objetivo é gerar uma base de conhecimento probabilística Γ_H tal que, se uma distribuição π satisfaz Γ_H, a probabilidade $P_\pi(T|W)$ é equivalente à calculada pelo HMM.

Seja a sequência de palavras $W = (w_1, \dots, w_M) = (o_{w(1)}, \dots, o_{w(M)})$, nós não estamos interessados na probabilidade de outras palavras serem geradas. Por isso precisamos de apenas uma variável lógica para cada palavra: ou $w_i = o_{w(i)}$ ou $w_i \neq o_{w(i)}$. Abusando da notação, vamos definir o conjunto de variáveis lógicas $\{w_1, \dots, w_M\}$ e $\{t_{i,j} | 1 \leq i \leq M, 1 \leq j \leq N\}$. A variável w_i será verdadeira se, e somente se, $w_i = o_{w(i)}$, a palavra que nos interessa. A variável $t_{i,j}$ é verdadeira se, e somente se, $t_i = s_j$. Isso nos dá um total de $NM + M$ variáveis lógicas.

Primeiro vamos eliminar a probabilidade de haver duas *tags* para a mesma palavra ou então uma palavra sem *tag*. Isso pode ser feito pelas restrições:

$$P(\bigvee_{j=1}^{m} t_{i,j}) = 1, \qquad 1 \leq i \leq M\,; \tag{3.1}$$

$$P(t_{i,j} \wedge t_{i,k}) = 0, \quad 1 \leq i \leq M, 1 \leq j, k \leq N, j \neq k\,. \tag{3.2}$$

Agora precisamos codificar as propriedades de um HMM em Γ_H. Considerando a Equação 2.17 da matriz B de um HMM, que nos dá a probabilidade de produzir uma palavra w_i em um estado t_j, produzimos a restrição:

$$P(w_i|t_{i,j}) = b_{j,w(i)}, 1 \leq i \leq M, 1 \leq j \leq N\,. \tag{3.3}$$

De maneira similar para a matriz A de transição de estados (Equação 2.16) e de estados iniciais (Equação 2.18).

$$P(t_{i,j}|t_{i-1,k}) = a_{k,j}, \quad 2 \leq i \leq M, 1 \leq j, k \leq N\,. \tag{3.4}$$

$$P(t_{1,j}) = q_j, \qquad 1 \leq j \leq N\,. \tag{3.5}$$

As únicas propriedades que faltam para expressar um HMM são as Propriedades de Markov (Equação 2.20) e de Independência (Equação 2.21). Ambas as propriedades tratam de independência estatística, que não é uma propriedade fácil de se expressar com PSAT de maneira genérica, como veremos com mais detalhes no capítulo 4. O jeito mais direto de representar essas propriedades sem entrar em detalhes, os quais veremos mais adiante, é listar exaustivamente todas as combinações e fixar os valores das conjunções.

A Propriedade de Independência implica $P(w_1 \wedge \dots \wedge w_n | t_{1,h_1} \wedge \dots \wedge t_{n,h_n}) = \prod_{i=1}^{M} P(w_i|t_{i,h_i})$. Se chamarmos de T uma etiquetagem completa, formada pela conjunção de M *tags*, e a

sentença completa formada por M palavras de W, podemos escrever a Propriedade da Independência como:

$$P(W|t_{1,h_1} \wedge \cdots \wedge t_{M,h_M}) = \prod_{i=1}^{M} b_{h_i,w(i)}, \quad \text{para qualquer } T\,. \tag{3.6}$$

Note-se que é necessário repetir a restrição 3.6 para todas as possíveis etiquetagens T, o que leva a um total de N^M restrições. Para codificar completamente um HMM, seria necessário também repetir essa restrição para todas as conjunções de palavras, porém, como só estamos interessados na probabilidade da sentença inteira, as demais serão suprimidas.

A Propriedade de Markov, que afirma que uma *tag* depende apenas da *tag* anterior, implica a restrição:

$$P(t_{i,h_i}|t_{i-1,h_{i-1}} \wedge \cdots \wedge t_{1,h_1}) = P(t_{i,h_i}|t_{i-1,h_{i-1}}), \text{para qualquer } T\,. \tag{3.7}$$

Para melhor representação no PSAT, a restrição 3.7 pode ser combinada com as restrições 3.4 e 3.5, que, juntas, criam a restrição equivalente:

$$P(t_{1,h_1} \wedge \cdots \wedge t_{M,h_M}) = q_{h_1} \prod_{i=2}^{M} a_{h_{i-1},h_i}, \text{para qualquer } T\,, \tag{3.8}$$

A base de conhecimento probabilística Γ_H que procuramos é formada pelas restrições 3.1, 3.2, 3.3, 3.6 e 3.8.

Proposição 4. *Dada uma sequência de* tags *T, as distribuições de probabilidade π que satisfazem a base de conhecimento probabilística Γ_H geram uma única probabilidade para $P_\pi(T|W)$, e esta medida de probabilidade corresponde à codificada pelo Modelo Oculto de Markov H.*

Demonstração. Primeiro vamos considerar a afirmação de que a probabilidade de $P_\pi(T|W)$ é única e não um intervalo. Pelo teorema de Bayes, temos que $P_\pi(T|W) = \frac{P_\pi(W|T)P_\pi(W)}{P_\pi(T)}$. Sabemos que $P_\pi(W|T)$ é único para qualquer π, pois existe uma restrição da forma 3.6 que força que qualquer distribuição π associe o mesmo valor para $P_\pi(W|T)$. Exatamente a mesma coisa acontece com $P_\pi(T)$ e uma restrição da forma 3.8. Consequentemente, cada $P_\pi(W \wedge T)$ é unicamente determinado. Devido às restrições 3.1 e 3.2, apenas as sequências de *tags* T possuem alguma probabilidade associada, e, portanto, podemos calcular unicamente $P_\pi(W)$ por marginalização, ou seja, pela equação $P_\pi(W) = \sum_T P_\pi(W \wedge T)$. Como cada elemento usado para seu cálculo é único, independente da distribuição π, $P_\pi(T|W)$ também é único.

Por construção, uma distribuição de probabilidades que associa a cada sequência de *tags* e palavras a mesma probabilidade obtida pelo HMM H satisfaz todas as restrições de Γ_H, e, portanto, essa medida deve ser pelo menos um dos resultados possíveis para $P_\pi(T|W)$. Sabendo que $P_\pi(T|W)$ é único, podemos concluir que $P_\pi(T|W)$ possui o mesmo valor codificado por H. □

Com a base de conhecimento Γ_H podemos decodificar um HMM resolvendo um problema PSAT. Porém, isso por si só não é útil. O algoritmo de Viterbi realiza essa mesma tarefa em tempo polinomial, enquanto o problema PSAT é NP-completo. Além disso, Γ_H possui um número exponencial de restrições do tipo 3.6 e 3.8.

Entretanto, a utilidade desta representação se tornará clara na próxima seção, em que modificamos Γ_H para ir além das limitações dos Modelos Ocultos de Markov.

3.3.2 Modelos não markovianos

Abandonando a Propriedade de Markov

Agora que temos uma representação de um HMM na forma de uma instância PSAT que calcula os mesmos valores de probabilidade *a posteriori*, podemos estender este modelo e remover algumas de suas limitações.

Como nosso objetivo é tratar dependências de longa distância, o meio mais óbvio é eliminar a Propriedade de Markov (restrição 3.8) da base de conhecimento probabilística. Lembrando que as restrições 3.4 e 3.5 haviam sido representadas conjuntamente em 3.8, precisamos reintroduzi-las. A base de conhecimento resultante por enquanto é formada por 3.1, 3.2, 3.3, 3.4, 3.5 e 3.6.

A Propriedade da Independência (3.6) também é problemática por um questão prática: é necessário um número exponencial N^M de restrições para exprimi-la. Como o problema PSAT, por ser NP-difícil, já tem uma complexidade de pior caso exponencial com os algoritmos atuais, $\Omega(2^{|\Gamma|})$, isso significa que, para obter uma distribuição de probabilidade π, pode exigir, no pior caso, $\Omega(2^{N^M})$ passos.

Porém, como veremos adiante, suprimir toda a Propriedade da Independência pode levar a resultados ruins. Iremos utilizar aqui o conceito de independência k a k e alguns métodos de resolução que serão apresentados com mais detalhes na seção 4.1, na qual revisitaremos esses conceitos.

A Propriedade da Independência, escrita $P(W|T) = \prod_{i=1}^{M} P(w_i|t_{i,h_i})$, pode ser vista como a condição que, dado um estado t_i, a palavra w_i é condicionalmente independente de todas as outras variáveis. Como não podemos exprimir a independência condicional de todas as variáveis, iremos apenas limitar a aplicação da Propriedade para sequências de ℓ palavras, para algum valor de $\ell > 1$.

$$
\begin{aligned}
&P(w_i \wedge \cdots \wedge w_{i+\ell-1} | t_{i,h_i} \wedge \cdots \wedge t_{i+\ell-1,h_{i+\ell-1}}) = \\
&= \prod_{j=i}^{i+\ell-1} P(w_j|t_{j,h_j}), \text{para todo } T \text{ e } 1 \leq i \leq M-\ell+1 \\
&= \prod_{j=i}^{i+\ell-1} b_{h_j,w(j)}, \qquad \text{para todo } T \text{ e } 1 \leq i \leq M-\ell+1. \qquad (3.9)
\end{aligned}
$$

Para um ℓ fixo, ao substituir as restrições do tipo 3.6 por 3.9, o tamanho da base de conhecimento se torna polinomial com grau ℓ, pois teremos, no máximo, $(M-\ell+1)N^\ell$

restrições do tipo 3.9.

Seja Γ^{ℓ}_{H} a base de conhecimento probabilística formada pelas restrições 3.1, 3.2, 3.3, 3.4, 3.5 e 3.9, apesar de a medida de probabilidade codificada pelo Modelo de Markov H satisfazer todas as restrições de Γ^{ℓ}_{H}, a Proposição 4 não é mais válida. Ou seja, podem existir distribuições π que satisfazem Γ^{ℓ}_{H} mas não Γ_H, e o valor das probabilidades de $P_\pi(T|W)$ é um intervalo ao invés de um único valor. Por causa disso, para encontrar uma etiquetagem, iremos selecionar uma única distribuição π.

Se pensarmos em cada π que satisfaz Γ^{ℓ}_{H} como um diferente modelo gerador que associa probabilidades para cada sequência de etiquetas e palavras observadas, podemos escolher o modelo π mais provável dadas as palavras observadas. Assumindo uma distribuição *a priori* uniforme sobre os possíveis π, esse critério se transforma em encontrar os valores de π onde $P_\pi(W)$ é máximo.

Para computar um π que maximize $P_\pi(W)$ e satisfaça Γ^{ℓ}_{H}, iremos resolver um problema OPSAT. É possível que exista mais de um π que maximize $P_\pi(W)$, porém, por uma questão prática, selecionaremos o primeiro π^* que seja solução para este problema OPSAT.

A distribuição de probabilidade π^* associa uma única probabilidade *a posteriori* para cada etiquetagem, $P_{\pi^*}(T|W)$. A solução do problema OPSAT vai associar probabilidades para apenas $|\Gamma^{\ell}_{H}| + 1$ etiquetagens, por isso, encontrar a etiquetagem com maior probabilidade leva apenas tempo linear no tamanho da base de conhecimento.

Tendo tudo isso em mente, podemos justificar por que devemos manter ao menos uma versão restrita da Propriedade da Independência, ao invés de eliminá-la completamente. Lembrando que a Propriedade da Independência pode ser entendida como a independência condicional das palavras dadas suas *tags*, sem essa independência em algumas distribuições π a probabilidade conjunta das palavras $P_\pi(W)$, ao invés de ser o produto da probabilidade da palavras, pode ser tão alta como $\min_i\{P_\pi(w_i)\}$. Nesse caso estaríamos levando em conta apenas uma palavra e em muitos casos esse seria o maior $P_\pi(W)$. As restrições 3.9 devem, ao menos, amenizar isso.

O Algoritmo 3 resume todo o processo descrito, partindo de um Modelo Oculto de Markov. Assuma-se que a função $OPSAT(\Gamma, P_\pi(\varphi))$ retorna uma dupla $(\pi^*, P_{\pi^*}(\varphi))$, onde π^* é a distribuição de probabilidades que maximiza $P_\pi(\varphi)$ e satisfaz Γ.

Entrada: Um HMM H e uma lista de palavras observadas $(w_1, \ldots, w_m)$
Saída: Uma etiquetagem morfossintática T^*
1 $\Gamma^{\ell}_{H} \leftarrow$ {associações de probabilidade 3.1, 3.2, 3.3, 3.4, 3.5 e 3.9}
2 $(\pi^*, P_W) \leftarrow$ `OPSAT` $(\Gamma^{\ell}_{H}, P_\pi(W))$
3 $\mathsf{T}^* \leftarrow$ `argmax`$_T$ $(P_{\pi^*}(T \wedge W))$
4 **retorna** T^*

Algoritmo 3: Realizando inferência em um HMM relaxado

Restrições extras considerando contexto ilimitado

Uma vez que transformamos os Modelos Ocultos de Markov para modelos não markovianos (que não possuem a propriedade de Markov), podemos considerar adicionar mais informações ao modelo. Se durante o treinamento com o *corpus*, além de obter as estatísticas necessárias ao HMM, também contarmos a frequência de *tags* que aparecem à esquerda de outras, independente da distância, podemos usar essa informação para refinar a busca por etiquetagens.

Essa informação pode ser codificada de muitas formas. Por exemplo, se levarmos em consideração a distância entre as palavras ao considerar sua influência, podemos ter:

$$P(t_{i,h_i}|t_{j,h_j}) = r_{h_i,h_j,i-j}, 1 \leq h_i, h_j \leq N, 1 \leq j < i \leq M\,. \tag{3.10}$$

Nessa fórmula, a associação de probabilidade $r_{h_i,h_j,i-j}$ depende da distância $i-j$ entre as palavras. Se ignorarmos essa distância, podemos escrever a restrição como:

$$P(t_{i,h_i}|t_{j,h_j}) = r_{h_i,h_j}, 1 \leq h_i, h_j \leq N, 1 \leq j < i \leq M\,. \tag{3.11}$$

Para diminuir o número de associações de probabilidade de $\mathcal{O}(N^2M^2)$ para $\mathcal{O}(N^2M)$, podemos usar a disjunção:

$$P(t_{i,h_i}|\bigvee_{j<i} t_{j,k}) = r'_{h_i,k}, 1 \leq h_i, k \leq N, 2 \leq i \leq M\,. \tag{3.12}$$

Porém, ao aumentar a base de conhecimento Γ_H^ℓ com a informação das fórmulas 3.10, 3.11 ou 3.12, não podemos mais ter certeza se a base de conhecimento resultante (a qual chamaremos de $\Gamma_H^{\ell+}$) será satisfazível. Antes usávamos o fato de que a solução do HMM satisfazia todas as restrições, porém esse não é mais o caso e nada podemos afirmar quanto à satisfazibilidade da base de conhecimento. Se ela for satisfazível, basta aplicar o mesmo procedimento indicado no Algoritmo 3. Caso contrário, precisamos de um procedimento diferente.

Tendo uma base de conhecimento inconsistente, podemos tentar encontrar a distribuição de probabilidade π que é, de acordo com alguma medida, a mais próxima de satisfazer as associações de probabilidade. Potyka e Thimm (2014) fizeram algo parecido com o objetivo de tentar consertar a base de conhecimento.

Como vimos na subseção 2.2.3, Equação 2.9, a probabilidade de uma fórmula condicional é definida como $P_\pi(\phi_i|\psi_i) = p_i \iff P_\pi(\phi_i \wedge \psi_i) = p_i P_\pi(\psi_i)$, ou equivalentemente, $P_\pi(\phi_i \wedge \psi_i) - p_i P_\pi(\psi_i) = 0$. Uma vez que a distribuição de probabilidade π pode não satisfazer essa equação, podemos medir o quão distante π está de satisfazê-la definindo variáveis de erro

$$\varepsilon_i = P_\pi(\varphi_i \wedge \psi_i) - p_i P_\pi(\psi_i)\,. \tag{3.13}$$

Basta então tentar minimizar essas variáveis de erro.

Seja $\varepsilon_i = P_\pi(\varphi_i \wedge \psi_i) - p_i P_\pi(\psi_i)$, queremos encontrar um π que minimize a p-norma do vetor $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_{|\Gamma_H^{\ell+}|})$, definida como:

$$\|\varepsilon\|_p = \sqrt[p]{\sum_{i=1}^{m} |\varepsilon_i|^p},$$

para $p \geq 1$. Quando $p \to \infty$, o limite da expressão leva a $\|\varepsilon\|_\infty = \max_i \varepsilon_i$, também conhecida como norma de Chebyshev. Potyka (2014) mostrou como $\|\varepsilon\|_p$ pode ser minimizado usando programação linear quando $p = 1$ ou $p = \infty$. Por isso usaremos essas normas como critério para encontrar os valores de π mais próximos de satisfazer $\Gamma_H^{\ell+}$.

Para encontrar uma etiquetagem em $\Gamma_H^{\ell+}$, primeiro escrevemos todas as associações de probabilidade da forma 3.13 e então minimizamos o erro $\|\varepsilon\|_p$. Seja o erro mínimo E, com $E = 0$ ou não, ainda assim $P\pi(T|W)$ pode ser um intervalo e por isso precisamos encontrar π^* que maximiza $P\pi^*(W)$, tal qual fizemos no Algoritmo 3. Por isso utilizamos novamente o OPSAT, mas desta vez adicionamos a restrição $\|\varepsilon\|_p = E$ para encontrar apenas modelos com o erro mínimo.

O Algoritmo 4 descreve todo o processo. A função $\text{OPSAT}_\varepsilon(\Gamma, \|\varepsilon\|_p)$ resolve uma instância PSAT Γ escrita com variáveis de erro minimizando o valor de $\|\varepsilon\|_p$.

Entrada: Um HMM H, uma lista de palavras observadas $(w_1, \ldots, w_m)$ e restrições extras Ψ da forma 3.10, 3.11 ou 3.12
Saída: Uma etiquetagem morfossintática T^*
1 $\Gamma_H^{\ell+} \leftarrow$ {associações de probabilidade 3.1, 3.2, 3.3, 3.4, 3.5 e 3.9} $\cup \Psi$
2 $(\pi_\varepsilon, E) \leftarrow \texttt{OPSAT}_\varepsilon(\Gamma_H^{\ell+}, \|\varepsilon\|_p)$
3 $\Gamma_H^{\ell+} \leftarrow \Gamma_H^{\ell+} \cup \{\|\varepsilon\|_p = E\}$
4 $(\pi^*, P_W) \leftarrow \texttt{OPSAT}(\Gamma_H^{\ell+}, P_\pi(W))$
5 $\mathsf{T}^* \leftarrow \texttt{argmax}_T(P_{\pi^*}(T \wedge W))$
6 **retorna** T^*

Algoritmo 4: Realizando inferência em um HMM possivelmente inconsistente

As bases de conhecimento Γ_H^{ℓ} e $\Gamma_H^{\ell+}$ são bases de conhecimento que derivam probabilidades para etiquetagens de palavras considerando mais do que seus contextos locais. No caso de $\Gamma_H^{\ell+}$, informação sobre a influência de longa distância é considerada diretamente, enquanto Γ_H^{ℓ} apenas remove a restrição que impede que valorações em que existe uma influência de longa distância sejam consideradas.

3.4 Implementação

Implementamos os algoritmos 3 e 4 em um projeto disponível publicamente, sobre licença GPLv3, em https://github.com/EduardoMMorais/HMM-PSAT. A implementação,

feita em `C++11`, depende das bibliotecas `Yices`[3] para o resolvedor SAT, `QSOpt_ex`[4] para resolução exata de programas lineares, e `GMP`[5] para aritmética exata de precisão arbitrária. Escolhemos usar aritmética exata com números racionais no lugar de números de ponto flutuante para evitar problemas de arredondamento.

Esses modelos não markovianos de etiquetagem morfossintática, teoricamente, são capazes de escolher etiquetagens que os métodos tradicionais teriam muita dificuldade em fazer. Porém, nossos experimentos com o método revelaram um ganho ínfimo de acurácia e em muitos casos uma perda.

O método do Algoritmo 3 revelou-se equivalente em acertos ao método dos Modelos Ocultos de Markov com 88,7242% de acerto em tags e 25,122% de acertos em sentenças completas no Córpus Histórico do Português Tycho Brahe. O Algoritmo 4 mostrou uma ligeira piora. Considerando que muitas dependências são, de fato, locais, a hipótese que levantamos é que esses métodos acabam "estragando" os resultados locais.

Como algumas das principais dificuldades envolvidas em sua implementação partem da representação da independência estatística, numa tentativa de melhor compreender as limitações dessas modelagens, partimos agora justamente para o estudo da independência estatística no PSAT.

[3] Disponível em: http://yices.csl.sri.com/
[4] Disponível em: https://github.com/jonls/qsopt-ex
[5] Disponível em: https://gmplib.org/

Capítulo 4

Modelos com independência parcial

Quando queremos modelar um modelo estatístico no PSAT, é importante que seja possível expressar independência probabilística, ou seja, selecionar modelos em que a probabilidade de um conjunto de variáveis não impacte a probabilidade de outra.

Mais precisamente, duas variáveis A e B são ditas *independentes* (denotado $A \perp\!\!\!\perp B$) se, e somente se,

$$P(A \wedge B) = P(A)P(B) \tag{4.1}$$

ou, equivalentemente,

$$P(B) = P(B|A) \text{ e } P(A) = P(A|B)$$

O conceito de *independência condicional* (denotado $A \perp\!\!\!\perp B|C$) pode ser obtido como uma simples generalização da independência.

$$P(A \wedge B|C) = P(A|C)P(B|C) \tag{4.2}$$

ou, equivalentemente,

$$P(B|C) = P(B|A, C) \text{ e } P(A|C) = P(A|B, C)$$

Um exemplo da utilidade de modelos estatisticamente independentes pode ser visto no capítulo 3, no qual estendemos *Modelos Ocultos de Markov* como uma instância PSAT. Porém, sem um mecanismo para expressar independência estatística, o modelo é dominado por uma única variável.

Uma propriedade interessante do PSAT é que, enquanto em técnicas como Redes Bayesianas, por exemplo, quanto mais variáveis independentes, e consequentemente menos variáveis dependentes, mais eficiente o algorítimo, no caso do PSAT, representar independência estatística é computacionalmente difícil, como veremos a seguir.

4.1 Independência mútua e independência k a k

Para um conjunto finito de variáveis $\chi = \{x_1, \ldots, x_n\}$, os eventos são *independentes dois a dois* se, e somente se,

$$P(x_j \wedge x_k) = P(x_j)P(x_k), \quad 1 \leq j \leq n, 1 \leq k \leq n, j \neq k \tag{4.3}$$

e as variáveis são ditas *mutualmente independentes* se, e somente se,

$$P(\bigwedge_{x \in \alpha} x) = \prod_{x \in \alpha} P(x), \quad \forall \alpha \subseteq \chi \tag{4.4}$$

Note-se que, como existe um número exponencial de subconjuntos α, expressar cada associação de probabilidade individualmente também requer um número exponencial de igualdades.

Como todo par é um subconjunto de χ, é claro que independência mútua implica independência par a par. Porém, o contrário não é verdadeiro. Em Hogg (2005) encontramos um exemplo, atribuído a Sergei Bernstein, de um conjunto de eventos independentes dois a dois porém não independentes mutuamente.

Exemplo 3. *Jogamos duas moedas honestas. Evento A ocorre se a primeira moeda é "cara", evento B ocorre se a segunda moeda é "cara", e o evento C ocorre se um, e apenas um dos eventos, é "cara".*

Os eventos A, B e C são independentes dois a dois mas não mutuamente independentes. Como as moedas são honestas, temos:

$$\begin{aligned} P(A) &= 0.5 \\ P(B) &= 0.5 \\ P(C) &= 0.5 \\ P(A \wedge B) &= 0.25 \\ P(A \wedge C) &= 0.25 \\ P(B \wedge C) &= 0.25 \end{aligned}$$

Porém, $P(A \wedge B \wedge C) = 0 \neq 0.125 = P(A)P(B)P(C)$.

O conceito de independência dois a dois pode ser estendido para *independência k a k* (*k-wise independence*), se, ao invés de considerar par de variáveis, considerarmos tuplas de tamanho k.

$$P(\bigwedge_{x \in \alpha} x) = \prod_{x \in \alpha} P(x), \quad \forall \alpha \subseteq \chi, |\alpha| = k \tag{4.5}$$

Variáveis de um conjunto de tamanho n são mutuamente independentes se elas são k-wise independentes para todo $k \leq n$. Neste trabalho, chamamos de *independência parcial*

quando um conjunto de variáveis é independente k a k mas não necessariamente mutuamente independente.

A independência estatística é uma propriedade impossível de ser expressada diretamente pela linguagem do PSAT, por exemplo, com uma fórmula proposicional como a Proposição 5 indica.

Proposição 5. *Sejam a e b duas variáveis proposicionais, não existe uma fórmula proposicional $f(a,b)$ tal que, para qualquer instância PSAT Π e qualquer distribuição de probabilidade π que satisfaz Π, $P_\pi(f(a,b)) = P_\pi(a)P_\pi(b)$.*

Demonstração. Considere uma instância PSAT Π com apenas duas variáveis, a e b e nenhuma associação de probabilidade, ou seja, qualquer distribuição a satisfaz.

Essa instância possui as valorações $v_1 = \{a = 0; b = 0\}, v_2 = \{a = 0; b = 1\}, v_3 = \{a = 1; b = 0\}, v_4 = \{a = 1; b = 1\}$.

Sejam π_1, π_2 e π_3 distribuições de probabilidade tal que $P_{\pi_1}(b) = 0$, $P_{\pi_2}(a) = 1$ e $P_{\pi_3}(b) = 1, P_{\pi_3}(a) = 0$ e assuma-se que existe $f(a,b)$ tal que $P_\pi(f(a,b)) = P_\pi(a)P_\pi(b)$ para todo π que satisfaz Π.

Isso significa que $P_{\pi_1}(f(a,b)) = 0$ e $P_{\pi_2}(f(a,b)) = P_{\pi_2}(b)$, o que significa que a fórmula proposicional $f(a,b)$ é verdadeira somente nas valorações em que b é verdadeiro.

Porém, $P_{\pi_3}(f(a,b)) = 0 \neq 1 = P_{\pi_3}(b)$. Logo não pode existir $f(a,b)$. □

A Proposição 5 restringe como podemos buscar modelos com independência estatística, pois nos garante que a independência só pode ser obtida como uma metapropriedade que utiliza um procedimento de resolução diferente das técnicas normalmente usadas para resolver o PSAT[1], ou seja, uma propriedade que gera uma classe de problema diferente. Por isso, podemos definir formalmente o problema PSAT com independência.

Definição 8. *O problema PSAT com independência parcial (INDEPSAT) consiste da tupla (Γ, Ψ, Υ), onde Γ é um conjunto de fórmulas proposicionais, Ψ é uma associação de probabilidades a variáveis de Γ, e Υ é um conjunto de conjuntos de variáveis.*

Uma distribuição de probabilidade sobre as variáveis de Γ é dita consistente com um problema INDEPSAT se todas as valorações com probabilidade maior que zero satisfazem todas as fórmulas de Γ, satisfazem as associações de probabilidade de Ψ e, para cada conjunto α de variáveis em Υ, satisfazem a Equação 4.5.

Observe-se que é possível exigir a independência mútua de um conjunto de variáveis se Υ contiver todos os subconjuntos do conjunto de variáveis da qual se quer a independência.

O caso trivial para conseguir distribuições de probabilidade independentes ocorre se as variáveis já estão restritas a um único valor probabilístico, isto é, se temos a probabilidade marginal da variável em questão. Nesse caso, se $P(a) = p_a$, $P(b) = p_b$ e queremos obter uma

[1] Apesar de o problema PSAT ser NP-difícil, o que significa que deve ser possível reduzir o problema de encontrar modelos com independência ao PSAT, a Proposição 5 sugere que qualquer tentativa de encontrar modelos independentes com PSAT deve envolver a manipulação das probabilidades por fórmulas proposicionais, ou seja, deve "simular" o procedimento de resolução do PSAT para trazer uma metapropriedade para a linguagem.

distribuição onde essas variáveis são independentes, basta adicionar a associação $P(a \wedge b) = p_a p_b$ à instância PSAT para garantir a independência de a e b. Note-se que temos os valores p_a e p_b neste caso.

Felizmente, essa é a versão que temos no problema da seção 3.3.

Porém, a desvantagem dessa representação aparece no caso da independência mútua de um conjunto de n variáveis. Precisamos de uma associação de probabilidade para cada subconjunto de variáveis, isto é, precisamos de 2^n associações para representar independência mútua.

Se nos contentarmos com independência k-wise, precisamos apenas de $O(n^k)$ associações de probabilidade, o que é polinomial no tamanho de n.

Se não temos a probabilidade marginal, o problema fica mais complicado. Veremos este caso na próxima seção.

4.2 Encontrando distribuições independentes

A Equação 4.5, que, por definição, garante que as variáveis de um conjunto são independentes, não pode ser usada diretamente como uma associação de probabilidade de um problema PSAT. A Equação 4.5 é claramente não linear, e, segundo o critério estabelecido no Proposição 1, a região viável de um problema de otimização formado por essa equação com certeza será não convexa e suscetível a problemas com mínimos locais.

Cozman *et al.* (2008) propõem o uso de programas multilineares, que podem tratar esse tipo de restrição não linear. Um exemplo de técnica de programação multilinear é justamente o problema 2.25, mostrado na parte sobre *Reformulation Linearization Technique* da subseção 2.4.1.

Programação multilinear no caso geral é muito ineficiente e no pior caso converge apenas no infinito. Porém, é possível encontrar distribuições independentes k a k mais rapidamente no caso especial de $k = 2$, ou seja, se procurarmos apenas distribuições dois a dois independentes. O Algoritmo 5 mostra como podemos encontrar tais distribuições, com uma margem de erro.

As funções `PSAT` e `OPSAT` utilizadas no Algoritmo 5 resolvem, respectivamente, o problema PSAT e OPSAT e retornam a distribuição de probabilidade que satisfaz ou otimiza a função objetivo. Essas funções retornam *Falso* caso não haja solução. A função `Heuristica` é uma aproximação "por baixo" do erro, ou seja, $\texttt{Heuristica}(\Psi) \leq \sum_{\alpha \in \Upsilon} \varepsilon_\alpha$. Mais detalhes adiante.

Intuitivamente, o Algoritmo 5 parte de uma solução inicial e, para cada variável de cada par independente, fixa todas as variáveis deixando apenas uma variável livre (linhas 12 e 20). Então minimizamos o erro da solução, ou seja, modificamos a variável livre de modo a deixar a probabilidade conjunta de um par o mais próximo possível do produto das probabilidades desse par (linhas 16 e 24). Eventualmente, o erro da solução chegará a um mínimo local (proposições 7 e 8). Como esse problema é não convexo, é possível que esse valor não seja um mínimo global. Utilizamos uma estratégia de *Branch and Bound* com

Entrada: Problema INDEPSAT (Γ, Ψ, Υ) onde $|\alpha| = 2, \forall \alpha \in \Upsilon$, margem de erro ϵ
Saída: Uma distribuição de probabilidade Π tal que, para todos os pares $\{a, b\} \in \Upsilon$, $|P(a \wedge b) - P(a)P(b)| \leq \epsilon$, ou FALSO caso não haja distribuição assim

1 $\Pi_0 \leftarrow \texttt{PSAT}(\Gamma, \Psi)$
2 **se** $\Pi_0 = Falso$ **então retorna** *Falso*
3 $i \leftarrow 0$
4 **para todo** $(a, b) \in \Upsilon$ **faça**
5 $\quad min_{a,b} \leftarrow |P_{\Pi_i}(a \wedge b) - P_{\Pi_i}(a)P_{\Pi_i}(b)|$
6 **fim**
7 $t \leftarrow$Verdadeiro
8 **enquanto** $t =$*Verdadeiro* **faça**
9 $\quad t \leftarrow$Falso
10 $\quad$ **para todo** $(a, b) \in \Upsilon$ **faça**
11 $\quad\quad i \leftarrow i + 1$
12 $\quad\quad \Psi_i \leftarrow \Psi \cup \{P(a \wedge b) + \varepsilon^+_{a,b} - \varepsilon^-_{a,b} = P_{\Pi_{i-1}}(a)P(b)\} \cup \{\varepsilon^+_{c,d} \geq 0, \varepsilon^-_{c,d} \geq 0 | \forall (c, d) \in \Upsilon\}$
13 $\quad\quad$ **para cada** *par* (c, d) *em* Υ, $c, d \neq b$ **faça**
14 $\quad\quad\quad \Psi_i \leftarrow \Psi_i \cup \{P(c) = P_{\Pi_{i-1}}(c)\} \cup \{P(d) = P_{\Pi_{i-1}}(d)\} \cup \{P(c \wedge d) + \varepsilon^+_{c,d} - \varepsilon^-_{c,d} = P_{\Pi_{i-1}}(c)P_{\Pi_{i-1}}(b)\}$
15 $\quad\quad$ **fim**
16 $\quad\quad \Pi_i \leftarrow \texttt{OPSAT}(\Gamma, \Psi_i, \text{MIN } \sum_{(a,b)\in\Upsilon} \varepsilon^+_{a,b} + \varepsilon^-_{a,b})$
17 $\quad\quad$ Repete procedimento das linhas 4–6
18 $\quad\quad$ **se** $min_{c,d} \leq \epsilon$ *para todo* $(c, d) \in \Upsilon$ **então retorna** Π_i
19 $\quad\quad i \leftarrow i + 1$
20 $\quad\quad \Psi_i \leftarrow \Psi \cup \{P(a \wedge b) + \varepsilon^+_{a,b} - \varepsilon^-_{a,b} = P(a)P_{\Pi_{i-1}}(b)\} \cup \{\varepsilon^+_{c,d} \geq 0, \varepsilon^-_{c,d} \geq 0 | \forall (c, d) \in \Upsilon\}$
21 $\quad\quad$ **para cada** *par* (c, d) *em* Υ, $c, d \neq a$ **faça**
22 $\quad\quad\quad \Psi_i \leftarrow \Psi_i \cup \{P(c) = P_{\Pi_{i-1}}(c)\} \cup \{P(d) = P_{\Pi_{i-1}}(d)\} \cup \{P(c \wedge d) + \varepsilon^+_{c,d} - \varepsilon^-_{c,d} = P_{\Pi_{i-1}}(c)P_{\Pi_{i-1}}(b)\}$
23 $\quad\quad$ **fim**
24 $\quad\quad \Pi_i \leftarrow \texttt{OPSAT}(\Gamma, \Psi_i, \text{MIN } \sum_{(a,b)\in\Upsilon} \varepsilon^+_{a,b} + \varepsilon^-_{a,b})$
25 $\quad\quad$ Repete procedimento das linhas 4–6
26 $\quad\quad$ **se** $min_{c,d} \leq \epsilon$ *para todo* $(c, d) \in \Upsilon$ **então retorna** Π_i
27 $\quad\quad$ **se** $P_{\Pi_i}(a) \neq P_{\Pi_{i-1}}(a)$ *ou* $P_{\Pi_{i-1}}(b) \neq P_{\Pi_{i-2}}(b)$ **então** $t \leftarrow$Verdadeiro
28 $\quad$ **fim**
29 **fim**
30 **se** $min_{a,b} \leq \epsilon$ *para todo* $(a, b) \in \Upsilon$ **então**
31 $\quad$ **retorna** Π_i
32 **senão**
33 $\quad$ Seleciona algum $(a, b) \in \Upsilon$ onde $min_{a,b} > \epsilon$
34 $\quad$ **se** $P_{\Pi_i}(a \wedge b) \geq P_{\Pi_i}(a)P_{\Pi_i}(b)$ **então**
35 $\quad\quad \Psi_{i+1} \leftarrow \Psi \cup \{P(a) > P_{\Pi_i}(a), P(b) > P_{\Pi_i}(b)\}$
36 $\quad\quad \Psi_{i+2} \leftarrow \Psi \cup \{P(a) < P_{\Pi_i}(a), P(b) < P_{\Pi_i}(b)\}$
37 $\quad$ **senão**
38 $\quad\quad \Psi_{i+1} \leftarrow \Psi \cup \{P(a) > P_{\Pi_i}(a), P(b) < P_{\Pi_i}(b)\}$
39 $\quad\quad \Psi_{i+2} \leftarrow \Psi \cup \{P(a) < P_{\Pi_i}(a), P(b) > P_{\Pi_i}(b)\}$
40 $\quad$ **fim**
41 $\quad$ **para cada** j *em* $\{i + 1, i + 2\}$ **faça**
42 $\quad\quad$ **se** $\texttt{Heuristica}(\Psi_j) \leq min_{a,b}$ *e a mudança entre as restrições de* Ψ_j *e* Ψ *é maior que* ϵ **então**
43 $\quad\quad\quad \Pi_j \leftarrow \texttt{Algoritmo5}((\Gamma, \Psi_j, \Upsilon), \epsilon)$
44 $\quad\quad\quad$ **se** $\Pi_j \neq Falso$ **então retorna** Π_j
45 $\quad$ **fim**
46 $\quad$ **retorna** *Falso*
47 **fim**

Algoritmo 5: Algoritmo para encontrar distribuições de probabilidade independentes

chamadas recursivas (linhas 42–44). À primeira vista, isso não tornaria o algoritmo muito mais atraente que o RLT. Porém, utilizamos uma propriedade do problema (proposição 11) para garantir que cada chamada recursiva precise analisar apenas 1/4 do espaço de busca da chamada anterior. Isso nos dá garantia de convergência exponencialmente rápida.

A seguir vamos mostrar que o Algoritmo 5 é correto (proposições 6 e 14) e completo (proposições 15 e 13), dentro de uma margem de erro ϵ.

Proposição 6. *Se o Algoritmo 5 não retorna Falso, ele retorna uma distribuição de probabilidade* Π *tal que* $|P_\Pi(a \wedge b) - P_\Pi(a)P_\Pi(b)| \leq \epsilon$ *para todo par* $(a, b) \in \Upsilon$.

Demonstração. As linhas que retornam uma distribuição de probabilidade são as linhas 18, 26 e 44.

As linhas 18 e 26 só podem ser alcançadas após um teste explícito que verifica se $min_{c,d} \leq \epsilon$ para todo par $(c, d) \in \Upsilon$. Cada $min_{c,d}$ é atualizado com $|P_\Pi(a \wedge b) - P_\Pi(a)P_\Pi(d)|$ antes de cada teste. Logo, as linhas 18 e 26 só podem retornar uma distribuição de probabilidade Π tal que $|P_\Pi(a \wedge b) - P_\Pi(a)P_\Pi(b)| \leq \epsilon$ para todo par $(a, b) \in \Upsilon$.

A linha 44 utiliza o resultado de uma chamada recursiva do algoritmo. Como todas as demais linhas que retornam distribuições de probabilidade satisfazem a propriedade que buscamos, por indução sobre o número de chamadas recursivas, podemos concluir que sempre que o Algoritmo 5 não retorna Falso, ele retorna uma distribuição de probabilidade que satisfaz a propriedade desejada. □

Proposição 7. *A distribuição de probabilidade* Π_i *obtida até a linha 29 do Algoritmo 5 é um mínimo local da função*

$$e(\Pi_i) = \sum_{(a,b)\in\Upsilon} P_{\Pi_i}(\varepsilon^+_{a,b}) + P_{\Pi_i}(\varepsilon^-_{a,b})$$

Demonstração. Quando chegamos à linha 29 e saímos do laço não existe nenhuma variável em nenhum par de Υ que pode ser alterado para minimizar a função $e(\Pi)$, pois as chamadas das linhas 16 e 24 não alteraram os valores relevantes de Π_i.

Além disso, a função $e(\Pi)$ é uma função linear e portanto muda uniformemente em uma mesma direção. Por isso, se existisse uma direção viável que diminuiria o valor de $e(\Pi)$, algumas das chamadas das linhas 16 e 24 alterariam algum valor de Π_i e não estaríamos na linha 29. □

Proposição 8. *Uma distribuição de probabilidade que é um mínimo local da função*

$$\sum_{(a,b)\in\Upsilon} P_{\Pi_i}(\varepsilon^+_{a,b}) + P_{\Pi_i}(\varepsilon^-_{a,b})$$

também é mínimo local da função

$$\sum_{(a,b)\in\Upsilon} |P_\Pi(a \wedge b) - P_\Pi(a)P_\Pi(b)|$$

Demonstração. A prova dessa proposição pode ser vista em Bertsimas e Tsitsiklis (1997, página 18), aplicando-se o argumento apresentado às nossas funções e restrições. □

Corolário 9. *A distribuição de probabilidade* Π_i *obtida até a linha 29 do Algoritmo 5 é um mínimo local da função*

$$\sum_{(a,b)\in\Upsilon} |P_\Pi(a \wedge b) - P_\Pi(a)P_\Pi(b)|.$$

Em muitos dos testes que fizemos do Algoritmo 5, apenas chegando na linha 29, ou seja, sem chamadas recursivas, já obtivemos uma distribuição de probabilidade independente, quando esta existia. Porém, não temos garantia de que no caso geral isso aconteça. Precisamos garantir que, além de mínimo local, Π_i seja o mínimo global. Com o mínimo global sabemos que, se Π_i não for a solução para o problema INDEPSAT, então não existe nenhuma.

Com uma estratégia de *Branch and Bound* podemos procurar mínimos globais. A função `Heuristica` é usada para saber se vale a pena explorar um subespaço de busca, porém, não é estritamente necessária, e uma função que sempre retorna ϵ também deve funcionar, mesmo que seja menos eficiente.

Uma boa candidata à função `Heuristica` é a linearização (Equação 2.27) do problema Ψ_{i+1}, acrescida da Equação 4.5 para cada par de Υ. O Proposição 3 garante que ela funcione para este caso.

Como veremos a seguir, uma propriedade desse problema permite que adicionemos restrições quanto ao valor de $P(a)$ e $P(b)$ à nossa instância, reduzindo assim o espaço de busca em 1/4 do anterior, e ainda assim encontremos o mínimo global.

Primeiro precisamos esclarecer como são as distribuições de probabilidade que fazem parte de uma corda entre duas distribuições de probabilidade.

Definição 9. *Seja* Π *um conjunto de distribuições de probabilidade sobre um conjunto de variáveis* X*,* $\pi_1, \pi_2 \in \Pi$*. A probabilidade de cada variável da distribuição de probabilidade* $\pi_3 = \lambda\pi_1 + (1-\lambda)\pi_2$ *para algum* $\lambda \in [0,1]$ *é*

$$P_{\lambda\pi_1+(1-\lambda)\pi_2}(a) = \lambda P_{\pi_1}(a) + (1-\lambda)P_{\pi_2}(a) \quad a \in X$$

Considere a seguinte função definida sobre o conjunto de distribuições de probabilidade um sobre conjunto de variáveis X.

$$g_{a,b}(\pi) = P_\pi(a \wedge b) - P_\pi(a)P_\pi(b) \quad a, b \in X \tag{4.6}$$

A Equação 4.5 é equivalente a $g(\pi) = 0$ com $a, b \in \alpha$. As restrições adicionadas ao problema Ψ_i no algoritmo espelham esse fato e pelo Corolário 9, o nosso algoritmo minimiza a somatória de $|g_{a,b}(\pi)|$ para todo (a, b). Vamos estudar critérios para obter condições próximas a condição de convexidade na função g.

Lema 10. *Sejam π_1 e π_2 distribuições de probabilidade, $\lambda \in [0,1]$ e $g(\pi)$ a Equação 4.6. Temos que*

$$g(\lambda\pi_1 + (1-\lambda)\pi_2) \geq \lambda g(\pi_1) + (1-\lambda)g(\pi_2) \quad \forall \pi_1, \pi_2, \forall \lambda \in [0,1]$$

se, e somente se,

$$P_{\lambda\pi_1+(1-\lambda)\pi_2}(a)P_{\lambda\pi_1+(1-\lambda)\pi_2}(b) \leq \lambda P_{\pi_1}(a)P_{\pi_1}(b) + (1-\lambda)P_{\pi_2}(a)P_{\pi_2}(b)$$

Analogamente,

$$g(\lambda\pi_1 + (1-\lambda)\pi_2) \leq \lambda g(\pi_1) + (1-\lambda)g(\pi_2) \quad \forall \pi_1, \pi_2, \forall \lambda \in [0,1]$$

se, e somente se,

$$P_{\lambda\pi_1+(1-\lambda)\pi_2}(a)P_{\lambda\pi_1+(1-\lambda)\pi_2}(b) \geq \lambda P_{\pi_1}(a)P_{\pi_1}(b) + (1-\lambda)P_{\pi_2}(a)P_{\pi_2}(b)$$

Demonstração. Ambas as proposições podem ser obtidas simplesmente expandindo a expressão original.

$$\begin{array}{rcl}
g(\lambda\pi_1 + (1-\lambda)\pi_2) & \leq & \lambda g(\pi_1) + (1-\lambda)g(\pi_2) \\
P_{\lambda\pi_1+(1-\lambda)\pi_2}(a \wedge b) & \leq & \lambda[P_{\pi_1}(a \wedge b) - P_{\pi_1}(a)P_{\pi_1}(b)] \\
-P_{\lambda\pi_1+(1-\lambda)\pi_2}(a)P_{\lambda\pi_1+(1-\lambda)\pi_2}(b) & & +(1-\lambda)[P_{\pi_2}(a \wedge b) - P_{\pi_2}(a)P_{\pi_2}(b)] \\
\lambda P_{\pi_1}(a \wedge b) + (1-\lambda)P_{\pi_2}(a \wedge b) & \leq & P_{\lambda\pi_1+(1-\lambda)\pi_2}(a)P_{\lambda\pi_1+(1-\lambda)\pi_2}(b) \\
-\lambda P_{\pi_1}(a \wedge b) - (1-\lambda)[P_{\pi_2}(a \wedge b) & & -\lambda P_{\pi_1}(a)P_{\pi_1}(b) - (1-\lambda)P_{\pi_2}(a)P_{\pi_2}(b) \\
0 & \leq & P_{\lambda\pi_1+(1-\lambda)\pi_2}(a)P_{\lambda\pi_1+(1-\lambda)\pi_2}(b) \\
 & & -\lambda P_{\pi_1}(a)P_{\pi_1}(b) - (1-\lambda)P_{\pi_2}(a)P_{\pi_2}(b) \\
\lambda P_{\pi_1}(a)P_{\pi_1}(b) + (1-\lambda)P_{\pi_2}(a)P_{\pi_2}(b) & \leq & P_{\lambda\pi_1+(1-\lambda)\pi_2}(a)P_{\lambda\pi_1+(1-\lambda)\pi_2}(b)
\end{array}$$

O caso $\geq$ é análogo. □

O Lema 10 apenas elimina a parte referente a $P_\pi(a \wedge b)$. Como esse termo é linear, era de se esperar que não fizesse diferença em relação à convexidade de $g(\pi)$.

Proposição 11. *Seja Π um conjunto de distribuições de probabilidade sobre um conjunto de variáveis X, $\pi_1, \pi_2 \in \Pi$ e $a, b \in X$, se $(P_{\pi_1}(a) - P_{\pi_2}(a))(P_{\pi_1}(b) - P_{\pi_2}(b)) \geq 0$, então*

$$g(\lambda\pi_1 + (1-\lambda)\pi_2) \geq \lambda g(\pi_1) + (1-\lambda)g(\pi_2) \quad \forall \pi_1, \pi_2, \forall \lambda \in [0,1]$$

Caso contrário,

$$g(\lambda\pi_1 + (1-\lambda)\pi_2) \leq \lambda g(\pi_1) + (1-\lambda)g(\pi_2) \quad \forall \pi_1, \pi_2, \forall \lambda \in [0,1]$$

Demonstração. Segundo o Lema 10, precisamos apenas analisar as expressões: $P_{\lambda\pi_1+(1-\lambda)\pi_2}(a)P_{\lambda\pi_1+(1-\lambda)\pi_2}(b)$ e $\lambda P_{\pi_1}(a)P_{\pi_1}(b) + (1-\lambda)P_{\pi_2}(a)P_{\pi_2}(b)$. Expandindo a primeira expressão temos:

$$\begin{aligned} P_{\lambda\pi_1+(1-\lambda)\pi_2}(a)P_{\lambda\pi_1+(1-\lambda)\pi_2}(b) &= [\lambda P_{\pi_1}(a) + (1-\lambda)P_{\pi_2}(a)][\lambda P_{\pi_1}(b) + (1-\lambda)P_{\pi_2}(b)] \\ &= \lambda^2 P_{\pi_1}(a)P_{\pi_1}(b) + (1-\lambda)^2 P_{\pi_2}(a)P_{\pi_2}(b) \\ &\quad +\lambda(1-\lambda)[P_{\pi_1}(a)P_{\pi_2}(b) + P_{\pi_2}(a)P_{\pi_1}(b)] \end{aligned}$$

Primeiro, vamos assumir que $(P_{\pi_1}(a) - P_{\pi_2}(a))(P_{\pi_1}(b) - P_{\pi_2}(b)) \geq 0$. Temos:

$$\begin{aligned} (P_{\pi_1}(a) - P_{\pi_2}(a))(P_{\pi_1}(b) - P_{\pi_2}(b)) &\geq 0 \\ P_{\pi_1}(a)P_{\pi_1}(b) + P_{\pi_2}(a)P_{\pi_2}(b) - P_{\pi_1}(a)P_{\pi_2}(b) - P_{\pi_2}(a)P_{\pi_1}(b) &\geq 0 \\ \lambda(1-\lambda)[P_{\pi_1}(a)P_{\pi_1}(b) + P_{\pi_2}(a)P_{\pi_2}(b) - P_{\pi_1}(a)P_{\pi_2}(b) - P_{\pi_2}(a)P_{\pi_1}(b)] &\geq 0 \\ \lambda(1-\lambda)P_{\pi_1}(a)P_{\pi_1}(b) + \lambda(1-\lambda)P_{\pi_2}(a)P_{\pi_2}(b) & \\ -\lambda(1-\lambda)[P_{\pi_1}(a)P_{\pi_2}(b) + P_{\pi_2}(a)P_{\pi_1}(b)] &\geq 0 \\ \lambda[(\lambda-1)P_{\pi_1}(a)P_{\pi_1}(b)] + (1-\lambda)[(-\lambda)P_{\pi_2}(a)P_{\pi_2}(b)] & \\ +\lambda(1-\lambda)[P_{\pi_1}(a)P_{\pi_2}(b) + P_{\pi_2}(a)P_{\pi_1}(b)] &\leq 0 \\ \lambda[\lambda P_{\pi_1}(a)P_{\pi_1}(b) - P_{\pi_1}(a)P_{\pi_1}(b)] + (1-\lambda)[(1-\lambda)P_{\pi_2}(a)P_{\pi_2}(b) - P_{\pi_2}(a)P_{\pi_2}(b)] & \\ +\lambda(1-\lambda)[P_{\pi_1}(a)P_{\pi_2}(b) + P_{\pi_2}(a)P_{\pi_1}(b)] &\leq 0 \\ P_{\lambda\pi_1+(1-\lambda)\pi_2}(a)P_{\lambda\pi_1+(1-\lambda)\pi_2}(b) - \lambda P_{\pi_1}(a)P_{\pi_1}(b) + (1-\lambda)P_{\pi_2}(a)P_{\pi_2}(b) &\leq 0 \end{aligned}$$

Pelo Lema 10, temos que $g(\lambda\pi_1 + (1-\lambda)\pi_2) \geq \lambda g(\pi_1) + (1-\lambda)g(\pi_2) \quad \forall\pi_1$. O caso $(P_{\pi_1}(a) - P_{\pi_2}(a))(P_{\pi_1}(b) - P_{\pi_2}(b)) \leq 0$ é análogo. □

A intuição da propriedade provada pela Proposição 11 pode ser vista no gráfico da função $P(a)P(b)$, representada na Figura 4.1. Observe-se que, nos casos em que $P(a)$ e $P(b)$ têm o mesmo sinal (ou seja, $(P_{\pi_1}(a) - P_{\pi_2}(a))(P_{\pi_1}(b) - P_{\pi_2}(b)) \geq 0$), o gráfico tem um formato côncavo e, no caso em que $P(a)$ e $P(b)$ têm os sinais diferentes, o gráfico tem formato convexo.

Segundo a Proposição 11, quando $(P_{\pi_1}(a) - P_{\pi_2}(a))(P_{\pi_1}(b) - P_{\pi_2}(b)) \geq 0$, g tem um formato côncavo, apesar de a função não ser realmente nem côncava nem convexa. Porém, essa propriedade é o suficiente para mostrar que não existem valores menores nessa região, de maneira semelhante ao feito no Proposição 2.

Proposição 12. *Seja π^* um mínimo local de g. Se $P_{\Pi^*}(a \wedge b) \geq P_{\Pi^*}(a)P_{\Pi^*}(b)$, então não existe π' tal que $g(\pi') < g(\pi^*)$ e $(P_{\pi^*}(a) - P_{\pi'}(a))(P_{\pi^*}(b) - P_{\pi'}(b)) \leq 0$.*

Seja π^* um máximo local de g. Se $P_{\Pi^*}(a \wedge b) < P_{\Pi^*}(a)P_{\Pi^*}(b)$, então não existe π' tal que $g(\pi') > g(\pi^*)$ e $(P_{\pi^*}(a) - P_{\pi'}(a))(P_{\pi^*}(b) - P_{\pi'}(b)) > 0$

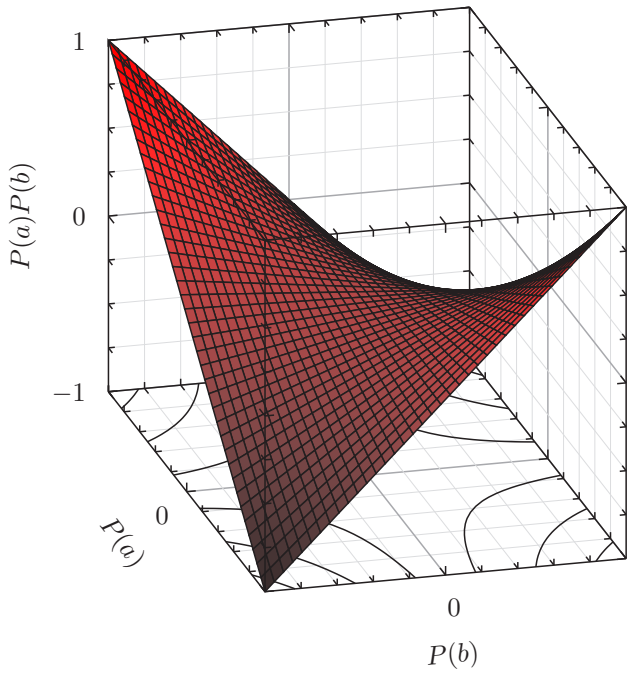

Figura 4.1: *Gráfico da função $P(a)P(b)$.*

Demonstração. De maneira análoga ao Teorema 2, consideramos que, se existe um π' tal que $g(\pi') < g(\pi^*)$ e $(P_{\pi^*}(a) - P_{\pi'}(a))(P_{\pi^*}(b) - P_{\pi'}(b)) \leq 0$, então, $g(\lambda\pi' + (1-\lambda)\pi^*) \leq \lambda g(\pi') + (1-\lambda)g(\pi^*) \quad \forall\lambda \in [0,1]$. No entanto, isso é uma contradição com a definição de mínimo local, pois implica que deve existir alguém na vizinhança de π^* menor ou igual a ele. A prova para o máximo local é análoga. □

Estamos querendo minimizar $|g(x)|$. Um mínimo local de $|g(\pi)|$ ou é mínimo de $g(\pi)$, ou é máximo de $-g(\pi)$, ou é 0 (e nesse caso, encontramos a resposta). Considerando a Proposição 12, percebemos que não precisamos pesquisar por valores do mínimo global quando $g(\pi) > 0$ e $(P_{\pi_1}(a) - P_{\pi_2}(a))(P_{\pi_1}(b) - P_{\pi_2}(b)) \geq 0$, pois, se fosse possível minimizar g, levando seu valor mais próximo a zero, seria possível minimizar $\sum_{(a,b)\in\Upsilon} P_{\Pi_i}(\varepsilon^+_{a,b}) + P_{\Pi_i}(\varepsilon^-_{a,b})$ e não estaríamos em um mínimo local. Analogamente, quando $g(\pi) < 0$ não precisamos procurar valores que minimizam $|g(\pi)|$ (assim maximizando $g(\pi)$) nas regiões onde $(P_{\pi_1}(a) - P_{\pi_2}(a))(P_{\pi_1}(b) - P_{\pi_2}(b)) \leq 0$. Este é o objetivo do bloco na linha 34 do algoritmo.

Corolário 13. *Seja Π^* o mínimo global da função $\sum_{(a,b)\in\Upsilon} P_{\Pi_i}(\varepsilon^+_{a,b}) + P_{\Pi_i}(\varepsilon^-_{a,b})$, o Algoritmo 5 encontra um Π tal que $|\Pi - \Pi^*| = \epsilon$.*

Corolário 14. *Se o Algoritmo 5 retorna Falso, não existe distribuição de probabilidade Π tal que $|P_\Pi(a \wedge b) - P_\Pi(a)P_\Pi(b)| \leq \epsilon$ para todo par $(a,b) \in \Upsilon$.*

Proposição 15. *Se $\epsilon > 0$, o Algoritmo 5 eventualmente para.*

Demonstração. O argumento é o mesmo usado para a parada de algoritmos *Branch and Bound* gerais. Após diversas chamadas recursivas, eventualmente o espaço de busca fica

menor que ϵ. Nesse caso a busca pode ser encerrada. A linha 42 do algoritmo verifica se o espaço de busca se tornou menor que ϵ. □

Considerando os resultados das proposições 6, 14, 15 e 13, podemos finalmente mostrar que o Algoritmo 5 é correto e completo para a resolução do problema INDEPSAT e uma alternativa mais eficiente para encontrar distribuições de probabilidade com propriedade de independência.

Teorema 16. *O Algoritmo 5 resolve o problema INDEPSAT, com margem de erro ϵ.*

Capítulo 5

Conclusões

Durante nosso trabalho com a aplicação de PSAT ao problema da etiquetagem morfossintática percebemos que o PSAT precisa tratar relações de independência. Nesta tese apresentamos as dificuldades envolvidas com o uso de relações de independência com lógica probabilística, propusemos um relaxamento e apresentamos um algoritmo que pode tratar relações de independência em casos especiais.

Uma linguagem sem independência probabilística acaba sofrendo de uma vacuidade inferencial (Cozman *et al.*, 2008), ou seja, é muito fácil fazer inferências vácuas e assim não obter resultados significativos. Um exemplo claro disso é o que encontramos em nossa modelagem PSAT de Modelos Ocultos de Markov. Sem independência entre as variáveis, a probabilidade de uma frase pode ser completamente dominada pela probabilidade de uma palavra, pois esse cenário é um modelo que satisfaz todas as restrições, mesmo não sendo um modelo que queremos.

Existe um balanço entre flexibilidade e expressividade em linguagens lógicas, e a independência estatística é um exemplo disso. Por um lado, assumir a independência de todas as variáveis inviabiliza inferências úteis. Por outro lado, não poder expressar independência, como era o caso do PSAT, nos leva a inferências vácuas e modelos matematicamente corretos, mas sem sentido. A independência k a k, como alternativa à independência mútua, entra como um passo intermediário nessa escala, aumentando os possíveis modelos de um problema enquanto descarta algumas inferências vácuas.

Considerando esses aspectos citados, podemos ver a importância do nosso novo algoritmo para encontrar distribuições de probabilidade independentes.

Um outro lado do balanço entre flexibilidade e expressividade refere-se à eficiência computacional. Nesse aspecto, apresentamos um diferencial: enquanto a maioria dos algoritmos probabilísticos torna-se mais ineficiente quanto maior o número de variáveis dependentes, no nosso caso, quanto menos pares independentes tivermos, melhor será a eficiência do nosso algoritmo. De certa forma, complementamos os algoritmos de Cozman *et al.* (2008) na busca da independência. Vale a pena destacar que, apesar de, no pior caso, com $\epsilon = 0$, haver o risco de o algoritmo não parar, na prática essa precisão raramente é necessária e, como o espaço de busca diminui exponencialmente a cada iteração, no caso médio o algoritmo

converge rapidamente.

Apesar de a nossa modelagem de HMMs sem a hipótese de Markov não oferecer uma melhora significativa de performance em relação a métodos anteriores, acreditamos que o processo de modelagem por si só tem utilidade. Podemos espelhar esse processo para outros modelos probabilísticos, eventualmente modificando hipóteses e associações de probabilidade, e usar o arcabouço já desenvolvido do PSAT para resolver esses novos modelos. A viabilidade dessa linha de pesquisa é um assunto para trabalhos futuros.

Referências bibliográficas

Bagno(2011) Marcos Bagno. *Gramática Pedagógica do Português Brasileiro*. Parábola Editorial, São Paulo. Citado na pág. 22

Baker(1975) James Baker. The dragon system–an overview. *Acoustics, speech and signal processing, IEEE transactions on*, 23(1):24–29. Citado na pág. 23

Baum *et al.***(1970)** Leonard E Baum, Ted Petrie, George Soules, e Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, páginas 164–171. Citado na pág. 10, 21

Bertsimas e Tsitsiklis(1997) Dimitris Bertsimas e John N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, Belmont, Massachusetts, EUA. ISBN 1-886529-19-1. Citado na pág. 39

Blunsom(2004) Phil Blunsom. Hidden markov models. *Lecture notes, August*, 15:18–19. Citado na pág. 12

Boole(1854) G. Boole. *An investigation of the laws of thought: on which are founded the mathematical theories of logic and probabilities*. Walton and Maberly. URL http://books.google.com/books?id=SWgLVT0otY8C. Citado na pág. 6, 9

Bubeck(2015) Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3–4):231–357. doi: 10.1561/2200000050. Citado na pág. 14

Bühlmann e Wyner(1999) Peter Bühlmann e Abraham J Wyner. Variable length markov chains. *The Annals of Statistics*, 27(2):480–513. Citado na pág. 13

Charnes e Cooper(1962) A. Charnes e W.W. Cooper. Programming with linear fractional functionals. *Naval Research Logistics Quarterly*, 9:181–186. Citado na pág. 9

Charniak(1996) Eugene Charniak. *Statistical language learning*. MIT press. Citado na pág. 22

Cozman *et al.***(2008)** Fabio Gagliardi Cozman, Cassio Polpo de Campos, e José Carlos Ferreira da Rocha. Probabilistic logic with independence. *International Journal of Approximate Reasoning*, 49(1):3–17. Citado na pág. 2, 36, 45

de Bona(2011) Glauber de Bona. Satisfazibilidade probabilística. Dissertação de Mestrado, Instituto de Matemática e Estatística da Universidade de São Paulo, 2011. Citado na pág. 7, 8

de Morais *et al.***(2015)** Eduardo Menezes de Morais, Glauber de Bona, e Marcelo Finger. Non-markovian logic-probabilistic modeling and inference. Em *IJCAI-2015 Workshop on Weighted Logics for Artificial Intelligence (WL4AI-2015)*. Citado na pág. 2, 21, 24

Finger e De Bona(2010) Marcelo Finger e Glauber De Bona. A logic based algorithm for solving probabilistic satisfiability. Em *IBERAMIA*, páginas 453–462. Springer. Citado na pág. 7

Finger e De Bona(2011) Marcelo Finger e Glauber De Bona. Probabilistic satisfiability: Logic-based algorithms and phase transition. Em *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*, páginas 528–533. Citado na pág. 7

Hansen e Jaumard(2000) P. Hansen e B. Jaumard. Probabilistic satisfiability. *Handbook of Defeasible Reasoning and Uncertainty Management Systems: Algorithms for uncertainty and defeasible reasoning*, página 321. Citado na pág. 9

Harris(1964) Zellig Sabbettai Harris. *String analysis of sentence structure*. Number 1 in Paper on formal linguistics. Mouton Publishers. Citado na pág. 22

Hendrix e G.-Tóth(2010) Eligius M. T. Hendrix e Boglárka G.-Tóth. *Introduction to Nonlinear and Global Optimization*, volume 37 de *Springer Optimization and Its Applications*. Springer. ISBN 978-0-387-88669-5. Citado na pág. 14, 15, 17

Hogg(2005) Robert Hogg. *Introduction to mathematical statistics*. Pearson Education, Upper Saddle River, N.J. ISBN 0130085073. Citado na pág. 34

Jaumard *et al.***(1991)** B. Jaumard, P. Hansen, e M. Poggi de Aragão. Column generation methods for probabilistic logic. *INFORMS Journal on Computing*, 3(2):135. Citado na pág. 9

Jurafsky e Martin(2000) Daniel S. Jurafsky e James H. Martin. *Speech and Language Processing*. Prentice Hall. Citado na pág. 21, 22, 23

Karlsson *et al.***(1995)** Fred Karlsson, Atro Voutilainen, Juha Heikkilae, e Arto Anttila. *Constraint Grammar: a language-independent system for parsing unrestricted text*, volume 4. Mouton de Gruyter, Berlin. Citado na pág. 22

Kepler(2010) Fábio Natanael Kepler. *Modelagem de contextos para aprendizado automático aplicado à Análise Morfossintática*. Tese de Doutorado, Instituto de Matemática e Estatística da Universidade de São Paulo. Citado na pág. 23

Klein e Simmons(1963) Sheldon Klein e Robert F Simmons. A computational approach to grammatical coding of english words. *Journal of the Association for Computing Machinery*, 10(3):334–347. Citado na pág. 22

Land e Doig(1960) A. H. Land e A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520. ISSN 00129682, 14680262. Citado na pág. 15

Mächler e Bühlmann(2012) Martin Mächler e Peter Bühlmann. Variable length markov chains: methodology, computing, and software. *Journal of Computational and Graphical Statistics*. Citado na pág. 13

Manning(2011) Christopher D. Manning. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? Em Alexander Gelbukh, editor, *Lecture Notes in Computer Science*, páginas 171–189. Springer-Verlag. Citado na pág. 21, 23, 24

Markov(1913) Andrey Andreyevich Markov. An example of statistical investigation in the text of eugene onyegin, illustrating coupling of tests in chains. Em *Proceedings of the Academy of Sciences of St. Petersburg*. Citado na pág. 10

Mesquita e Martos(1994) Roberto Melo Mesquita e Cloder Rivas Martos. *Português - Linguagem & Realidade*, volume 1. Editora Saraiva, São Paulo, 3 edição. Citado na pág. 22

Nilsson(1986) N.J. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28(1):71–87. Citado na pág. 6, 8

Potyka(2014) Nico Potyka. Linear programs for measuring inconsistency in probabilistic logics. Em *Fourteenth International Conference on Principles of Knowledge Representation and Reasoning (KR-14)*. AAAI. Citado na pág. 30

Potyka e Thimm(2014) Nico Potyka e Matthias Thimm. Consolidation of probabilistic knowledge bases by inconsistency minimization. Em *ECAI 2014*. Citado na pág. 29

Prasolov e Tikhomirov(2001) V.V. Prasolov e V.M. Tikhomirov. *Geometry*. American Mathematical Society. Citado na pág. 7

Sherali e Adams(1999) Hanif D. Sherali e Warren P. Adams. *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*, volume 31 de *Nonconvex Optimization and Its Applications*. Kluwer Academic Publishers. ISBN 0-7923-5487-7. Citado na pág. 16, 19

Sherali e Adams(1994a) Hanif D. Sherali e Warren P. Adams. A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems. *Discrete Applied Mathematics*, (52):83–106. Citado na pág. 16

Sherali e Adams(1994b) Hanif D. Sherali e Warren P. Adams. A reformulation-linearization technique for solving discrete and continuous nonconvex problems. *Mathematics Today*, XII-A:61–78. Citado na pág. 16

Viterbi(1967) A.J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269. ISSN 0018-9448. Citado na pág. 12