Modelagem de contextos para aprendizado automático aplicado à análise morfossintática

Fábio Natanael Kepler

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

Programa: Ciência da Computação
Orientador: Marcelo Finger

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro da CAPES por meio de bolsa de doutorado e de bolsa de doutorado sanduíche.

São Paulo, julho de 2010.

Fábio Natanael Kepler: *Modelagem de contextos para aprendizado automático aplicado à análise morfossintática*, Tese de Doutorado, © julho de 2010.

HOMEPAGE:
http://www.ime.usp.br/~kepler

E-MAIL:
kepler@ime.usp.br

Modelagem de contextos para aprendizado automático aplicado à análise morfossintática

Este exemplar corresponde à redação
final da tese devidamente corrigida
e defendida por Fábio Natanael Kepler
e aprovada pela Comissão Julgadora.

Comissão Julgadora

Prof. Dr. Marcelo Finger (orientador) – IME-USP

Prof. Dr. Ruy Luiz Milidiú – PUC-Rio

Profa. Dra. Sandra Maria Aluísio – ICMC-USP

Profa. Dra. Maria Clara Paixão de Sousa – FFLCH-USP

Prof. Dr. Flávio Soares Corrêa da Silva – IME-USP

Aos meus pais, que me fizeram como sou,
e à Michelle, que gostou de mim desse jeito.

ABSTRACT

Part-of-speech tagging involves assigning to words in a sentence their part-of-speech class based on the contexts they appear in. Variable-Length Markov Chains (VLMCs) offer a way of modeling contexts longer than trigrams without suffering too much from data sparsity and state space complexity. Even so, two words in Portuguese show a high degree of ambiguity: *que* and *a*. The number of errors tagging these words corresponds to a quarter of the total errors made by a VLMC-based tagger. Moreover, these words seem to show two different types of ambiguity: one depending on non-local context and one on right context. We searched ways of expanding the VLMC-based model with a number of different models and methods in order to tackle these issues. The approaches showed variable degrees of success, with one particular method (Guided Learning) solving much of the ambiguity of *a*. We explore reasons why this happened. Regarding *que*, throughout this thesis we propose and test various methods for learning contextual information in order to try to disambiguate it. We show how, in all of them, the level of ambiguity shown by *que* remains practically constant.

RESUMO

A etiquetagem morfossintática envolve atribuir às palavras de uma sentença suas classes morfossintáticas de acordo com os contextos em que elas aparecem. Cadeias de Markov de Tamanho Variável (VLMCs, do inglês *Variable-Length Markov Chains*) oferecem uma forma de modelar contextos maiores que trigramas sem sofrer demais com a esparsidade de dados e a complexidade do espaço de estados. Mesmo assim, duas palavras do português apresentam um alto grau de ambiguidade: *que* e *a*. O número de erros na etiquetagem dessas palavras corresponde a um quarto do total de erros cometidos por um etiquetador baseado em VLMCs. Além disso, essas palavras parecem apresentar dois diferentes tipos de ambiguidade: um dependendo de contexto não local e outro de contexto direito. Exploramos maneiras de expandir o modelo baseado em VLMCs através do uso de diferentes modelos e métodos, a fim de atacar esses problemas. As abordagens mostraram variado grau de sucesso, com um método em particular (aprendizado guiado) se mostrando capaz de resolver boa parte da ambiguidade de *a*. Discutimos razões para isso acontecer. Com relação a *que*, ao longo desta tese propusemos e testamos diversos métodos de aprendizado de informação contextual para tentar desambiguá-lo. Mostramos como, em todos eles, o nível de ambiguidade de *que* permanece praticamente constante.

AGRADECIMENTOS

Há muitas pessoas a quem agradecer. Logo de começo, quero agradecer à minha esposa e à toda minha família pelo apoio e vida que me deram neste período de desenvolvimento deste trabalho. À Michelle, a mulher que, quando comecei este trabalho, era minha namorada, depois se tornou minha noiva e hoje é minha esposa, que aguentou os seis meses que passei fora no período sanduíche, que aceitou dormir ao meu lado enquanto escrevia esta tese, que me incentiva na minha vida profissional mas não me deixa esquecer da minha vida pessoal, a ela, meu muito obrigado por todo amor incondicional. Aos meus pais, Dieter e Maria Luiza, que, desde que lembro ter consciência, me apoiaram e incentivaram em minhas iniciativas, inclusive com “puxões de orelha”, agradeço de coração o amor também incondicional. Aos meus irmãos, sogros, cunhados e cunhadas, tios e primos, muito obrigado por todas conversas, visitas, passeios e horas divertidas. É bom ter todos vocês como família.

Meu orientador, Marcelo Finger, merece todos meus agradecimentos. Muito obrigado pela força, ânimo e "aulas"sobre como ser um melhor pesquisador. Desculpe as dores de cabeças que causei :).

Quero também agradecer ao Mitchell Marcus que me recebeu na Universidade da Pensilvânia como coorientador durante meu doutorado sanduíche, e me mostrou uma parte da vida acadêmica muito importante e interessante. Aproveitando que falei sobre esse tempo nos EUA, quero agradecer a todo pessoal que encontrei lá e com quem fiz amizade, que fizeram esse período valer realmente a pena (e a pena não foi fácil, né Mi?).

De volta ao Brasil, falta agradecer, e continuam sendo importantes, todos meus amigos que convivem comigo em São Paulo e todos que estão em Panambi, Porto Alegre e outros lugares. Muito obrigado. Obrigado também ao pessoal da Brasiliana USP, que conheci faz menos tempo, mas que já me mostrou diversas coisas sobre a vida profissional e também já me divertiu muito.

Finalmente, ao Salvador da minha vida que conheci pessoalmente e que está comigo desde então, em qualquer circunstância.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

ACRÔNIMOS

CCM Modelo de Constituintes e Contextos[1] 35
EM Maximização de Expectativa[2] 34
HMM Modelo de Markov Oculto[3] 13
ML Máxima Verossimilhança[4] 9
PLN Processamento de Linguagem Natural 3
VLMC Cadeia de Markov de Tamanho Variável[5] 14
VLMM Modelo de Markov de Tamanho Variável[6] 23

1 CCM, do inglês *Constituent-Context Model*.
2 EM, do inglês *Expectation Maximization*.
3 HMM, do inglês *Hidden Markov Model*.
4 ML, do inglês *Maximum Likelihood*.
5 VLMC, do inglês *Variable-Length Markov Chain*.
6 VLMM, do inglês *Variable-Length Markov Model*.

Parte I

INTRODUÇÃO

1 INTRODUÇÃO

O objetivo de longo prazo da área de Processamento de Linguagem Natural (PLN) é alcançar o entendimento da linguagem pela máquina. Um pré-requisito para isto é extrair estrutura de um texto. Uma única sentença possui mais de um tipo de estrutura implícita, e algumas estruturas dependem de outras. Por exemplo, a estrutura sintática depende das classes gramaticais das palavras para determinar os componentes da sentença. Já estruturas de discurso dependem de mais de uma sentença.

Até duas décadas atrás os esforços em recuperar diferentes estruturas de um texto se concentraram em abordagens baseadas em regras, buscando utilizar gramáticas elaboradas por especialistas em linguagem natural. O resultado negativo encontrado, e que motivou a mudança de abordagem, foi que os sistemas desenvolvidos só funcionaram bem para domínios restritos da língua, e os especialistas necessários eram caros, principalmente em relação à disponibilidade de tempo.

A abordagem que passou a ser cada vez mais usada buscava empregar modelos estatísticos para calcular, inferir e predizer relações entre palavras e estruturas. Depois de um tempo, modelos combinando evidências positivas do uso da linguagem passaram a obter resultados promissores, o que motivou ainda mais a concentração de foco na abordagem estatística.

Nosso foco será em modelos estatísticos baseados em evidências.

1.1 OBJETIVO

Desenvolvemos, implementamos e testamos novas abordagens para a análise morfossintática do português, buscando resolver especificamente o problema das palavras com alto nível de ambiguidade.

1.2 DECLARAÇÃO

Esta tese apresenta apenas texto, implementações e resultados originalmente escritos, desenvolvidos e executados por mim, Fábio Natanael Kepler, sob orientação de Marcelo Finger, ao longo do período do doutorado, a não ser que seja dito o contrário ao longo do texto, caso em que haverá ao menos uma citação dos autores originais. Há três exceções que merecem destaque: uma parte que foi originalmente desenvolvida no meu mestrado; o modelo CCM, descrito no capítulo 4, que não foi inventado por mim mas cuja implementação e resultados são meus; e o etiquetador bidirecional usando aprendizado guiado, que foi criado e implementado por seus autores originais, e ao qual eu tive acesso permitido e pude testar nos recursos usados nesta tese.

1.3 PUBLICAÇÕES

A parte inicial deste trabalho foi publicada após o término do mestrado em:

- PROPOR – *7th International Workshop on Computational Processing of Written and Spoken Portuguese*, 2006;
- IBERAMIA-SBIA – *2nd International Joint Conference, 10th Ibero-American Conference on AI, 18th Brazilian AI Symposium*, 2006;
- CTD – Concurso de Teses e Dissertações da Sociedade Brasileira de Computação (SBC), 2006.

Um artigo contendo vários resultados desta tese foi apresentado e publicado no NAACL-HLT 2010[1]: *Young Investigators Workshop on Computational Approaches to Languages of the Americas*, realizado em junho de 2010.

1.4 ORGANIZAÇÃO DA TESE

Continuando a Parte i, no Capítulo 2 apresentamos a tarefa da área de PLN na qual destacaremos nosso trabalho, dando fundamentos usados no restante da tese e explicando o uso de cadeias de Markov e os resultados iniciais obtidos na etiquetagem morfossintática do português. No Capítulo 3 então descrevemos os problemas encontrados e refinamos os objetivos desta tese.

A Parte ii contém as abordagens utilizados para atacar os problemas descritos: no Capítulo 4 descrevemos o uso de informação sintática explícita na etiquetagem morfossintática; no Capítulo 5 mostramos uma abordagem que utiliza informação sintática de forma indireta; no Capítulo 6 descrevemos o uso do algoritmo de treinamento Perceptron; no Capítulo 7 apresentamos uma abordagem bidirecional simples para etiquetagem; no Capítulo 8 mostramos o uso de uma abordagem bidirecional mais sofisticada; e no Capítulo 9 descrevemos mais alguns experimentos realizados.

Na Parte iii, por fim, comparamos, no Capítulo 10, os resultados obtidos por essas abordagens e, no Capítulo 11, tecemos as conclusões finais, listando problemas persistentes e idéias de trabalho futuro.

1 *Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, http://naaclhlt2010.isi.edu.

2 ANÁLISE MORFOSSINTÁTICA

Os linguistas agrupam as palavras de uma língua em classes que mostram comportamento sintático semelhante. Estas classes de palavras também são chamadas de categorias gramaticais ou morfossintáticas. A correta classificação de palavras é útil para outros estudos da linguística, como tradução automática, sumarização automática e extração de informações (Manning e Schütze, 1999).

Considere, entretanto, uma palavra muito comum no português: a palavra *a*. Qual seria sua categoria morfossintática? Ela poderia ser:

UM DETERMINANTE como em *"é a escolha"*;

UMA PREPOSIÇÃO como em *"que leva a um nível"*;

UM PRONOME CLÍTICO como em *"já a tomei"*.

Ou seja, existe um problema de ambiguidade, no qual uma palavra pode ter mais de um significado ou função, e portanto mais de uma classe gramatical. Quando olhamos para o contexto da palavra, entretanto, seu significado fica muito mais claro. Por exemplo, considere a sentença

> *"A bebida aqui é muito cara."*

Considerando apenas a palavra *bebida*, ela é ambígua porque pode ser classificada como substantivo e como verbo. Mas considerando seu contexto, facilmente detectamos que é o primeiro caso que se aplica. O segundo caso aconteceria, por exemplo, em *"Toda água foi bebida em questão de minutos"*.

A *Análise Morfossintática* consiste em classificar gramaticalmente cada palavra de uma frase dentro do seu contexto. A análise morfossintática correta das duas sentenças acima, usando uma notação de etiquetas que explicaremos mais adiante, é:

> *"A/D bebida/N aqui/ADV é/VB muito/ADV cara/ADJ ./."* e
>
> *"Toda/ADV água/N foi/VB bebida/VB em/P questão/N de/P minutos/N ./."*,

onde `D` representa a classe dos determinantes (artigos), `VB` dos verbos, `P` das preposições, `N` dos nomes (substantivos), `ADV` dos advérbios, `ADJ` dos adjetivos e `.` dos símbolos de pontuação final.

Para um ser humano, detectar as diferenças de contexto é uma tarefa trivial, e geralmente nem feita conscientemente. Entretanto, não há um conjunto finito de regras para se distinguir os significados de todas as palavras em seus diversos usos na linguagem. Isto torna a realização automática dessa análise via computador, e também dos demais níveis de análise da linguagem natural, uma tarefa complexa e desafiadora.

Outros tipos de ambiguidade existem na linguagem, como ambiguidade sintática na ligação de sintagmas, ambiguidade em coordenação, ambiguidade semântica, ambiguidade de tradução etc. Sempre que as ações de um sistema dependam do significado do texto sendo processado, resolver a ambiguidade é necessário.

Um programa de computador feito para realizar a análise morfossintática é chamado de *etiquetador morfossintático*, e seu desenvolvimento é, há anos, alvo de diversas pesquisas da comunidade de PLN, tanto a do português (Aires, 2000; Alves e

Finger, 1999) quanto a do inglês (Brants, 2000; Brill, 1995; Cutting, Kupiec, Pedersen e Sibun, 1991; Ratnaparkhi, 1996). As categorias morfossintáticas também podem ser chamadas de etiquetas morfossintáticas, daí o nome *etiquetador*.

Desenvolver um etiquetador que resolva o máximo de ambiguidade possível é um dos focos desse trabalho. Sempre que não for dito o contrário, usaremos o termo *etiqueta* para nos referirmos à categoria morfossintática.

2.1 FUNDAMENTOS

Vamos usar modelos estatísticos de etiquetagem (Charniak, 1993; Manning e Schütze, 1999; Samuelsson e Voutilainen, 1997). Mais especificamente, modelos baseados em cadeias de Markov. Além disso, estamos particularmente interessados no método de aprendizado supervisionado. Iremos explicar esses conceitos mais adiante.

Modelos estatísticos modelam probabilidades em seus parâmetros, que podem ser de diferentes tipos. Utilizamos a *Teoria da Probabilidade* quando queremos prever ou medir as chances de algo acontecer. Usando termos simples, essa probabilidade geralmente é denotada por $P(X)$, ou seja, a probabilidade de X acontecer. Se quisermos saber a probabilidade de Y acontecer juntamente com X, usamos a probabilidade conjunta, denotada por $P(X, Y)$. Podemos, também, condicionar as chances de um resultado a algo que já aconteceu: $P(Y|X)$ representa a probabilidade de Y ocorrer dado que X ocorreu.

Em um modelo para etiquetagem, podemos usar $P(t|w)$, que é a probabilidade da palavra w ter a etiqueta t, por exemplo.

Nas seções seguintes descreveremos noções básicas sobre análise morfossintática, modelos e métodos estatísticos e cadeias de Markov. Quando relevante, e se não dito o contrário, usaremos a notação normalmente utilizada em textos sobre processamento estatístico de linguagem natural, mostrada na Tabela 1.

w_i	a palavra na posição i no córpus ou na sentença
t_i	a etiqueta de w_i
$w_{i,i+m}$	a sequência de palavras de w_i até w_{i+m}
$t_{i,i+m}$	as etiquetas de $w_{i,i+m}$
w^k	a k-ésima ocorrência de w no léxico
t^j	a j-ésima etiqueta no conjunto de etiquetas
n	tamanho da sentença
T	conjunto de etiquetas
$\|T\|$	cardinal do conjunto de etiquetas

Tabela 1: Notação utilizada.

2.1.1 Fundamentos da Análise Morfossintática

O processo de análise morfossintática estatística, assim como outros processos em PLN, é separado em dois componentes:

MODELO é a função que define o espaço de eventos e os parâmetros a serem associados a cada evento;

ANALISADOR propriamente dito, é o algoritmo que implementa a busca pela melhor sequência de etiquetas para cada sentença.

Nesse cenário, se o espaço de eventos possíveis for associado aos mapeamentos entre as sentenças (S) e suas respectivas etiquetas (A), o aprendizado passa a ser a indução de uma função probabilística

$$\textit{Pontuação} : a \times s \rightarrow [0,1],$$

$a \in A$, $s \in S$, em que $\textit{Pontuação}(a, s)$ pode ser tanto uma probabilidade conjunta, $P(a, s)$, quanto uma probabilidade condicional, $P(a|s)$. A sequência de etiquetas mais provável para uma sentença de entrada s é então definida como

$$a_{\text{melhor}}(s) = \arg\max_{a \in A} \textit{Pontuação}(a, s) \tag{2.1}$$

Deste modo, os dois componentes da análise podem ser definidos como:

MODELO é a função que define a probabilidade $\textit{Pontuação}(a, s)$ para cada par (a, s);

ANALISADOR é o algoritmo que implementa a busca por a_{melhor} para qualquer sentença de entrada s.

Para que o modelo tenha um número tratável de parâmetros, ou seja, para que o espaço de estados não tenha complexidade excessiva, um par (a, s) precisa ser quebrado em um conjunto de eventos independentes

$$\langle \textit{Evento}_1 \ldots \textit{Evento}_n \rangle.$$

$\textit{Pontuação}(a, s)$ é então calculada como um produto de termos, cada *Evento* tendo sua probabilidade correspondente:

$$\textit{Pontuação}(a, s) = \prod_{i=1 \ldots n} \textit{Pontuação}(\textit{Evento}_i). \tag{2.2}$$

A escolha da *parametrização* é a escolha de quais eventos associar a parâmetros. Existem muitas maneiras possíveis de se fazer isso. Os métodos estatísticos na literatura diferem pelo modo como o fazem, ou seja, como determinam quais objetos linguísticos são considerados para estimar os parâmetros de seu modelo.

A escolha da parametrização é central para o sucesso de um modelo de análise. Uma boa parametrização deve representar concisamente os dados. Segundo Collins (1999), dois critérios são importantes:

PODER DISCRIMINATIVO: os parâmetros devem incluir informação contextual necessária para decisões de desambiguação. Os modelos mais simples falham nesse sentido, por serem insensíveis demais a informações lexicais e preferências estruturais. Por exemplo: as preferências estruturais de ligações entre os sintagmas dependem de informações lexicais (como as palavras que estão sendo ligadas) para serem realizadas;

PODER DE COMPACTAÇÃO: o modelo deve ter o menor número possível de parâmetros. O número de parâmetros do modelo determina a quantidade de dados de treinamento necessária para treiná-lo. Isso significa que o poder de compactação do modelo determinará o quão perto ele chega do seu potencial máximo, dada a quantidade (quase certamente) limitada de dados para treinamento.

Esses dois critérios evoluem em sentidos opostos. Quanto mais parâmetros um modelo possui, maior é, em teoria, seu poder discriminativo, e menor seu poder de compactação. Em um nível extremo, por exemplo, a geração de uma sequência de

etiquetas poderia ser composta por um único passo, o da própria geração de toda sequência. Este caso pecaria pela compactação, já que o número de parâmetros a ser estimado seria do tamanho do domínio de sentenças possíveis da linguagem, ou seja, extremamente grande ou mesmo infinito. A quantidade de dados de treinamento precisaria ser igualmente extensa, e portanto inviável. Desse modo, um bom modelo deve ser obtido pelo equilíbrio dos dois critérios, sempre visando eliminar parâmetros redundantes, ou seja, que diferenciam as mesmas ambiguidades.

Para isto, um modelo parametrizável possui um argumento adicional na função *Pontuação*, dado por um vetor de parâmetros Θ. A função é então escrita como $\mathit{Pontuação}(a, s|\Theta)$, e o problema do aprendizado consiste em se ajustar a estimativa dos valores contidos em Θ a partir do conjunto de exemplos de treinamento. Isso geralmente é feito via dois métodos diferentes: aprendizado supervisionado e aprendizado não-supervisionado. Explicaremos esses métodos na Seção 2.1.3.

A tarefa de modelagem da análise morfossintática é, portanto, dividida em três etapas:

A. Definição do modelo de estrutura: a função $\mathit{Pontuação}(a, s|\Theta)$;

B. Definição do modelo parametrizável;

C. Definição do algoritmo de busca para encontrar

$$a_{\text{melhor}}(s) = \arg\max_{a} \mathit{Pontuação}(a, s|\Theta). \tag{2.3}$$

O algoritmo de busca geralmente utilizado em etiquetagem morfossintática é o *algoritmo de Viterbi* (Viterbi, 1967). Os parâmetros de um modelo são definidos de formas diferentes para cada abordagem diferente. Mostraremos várias delas ao longo desse trabalho. Já os modelos de estrutura utilizados em etiquetadores morfossintáticos podem ser vários, mas eles geralmente são especializações de duas classes diferentes de modelos estatísticos. Apresentaremos essas duas classes na próxima seção.

2.1.2 Classes de Modelos Estatísticos

Duas classes de modelos estatísticos amplamente utilizadas em PLN são a dos *modelos gerativos* e a dos *modelos discriminativos*. A diferença básica está na forma como cada modelo especifica as distribuições de probabilidade.

Um modelo gerativo gera aleatoriamente informações observáveis dados parâmetros ocultos. Isso é feito especificando-se distribuições de probabilidade conjunta sobre observações s e sequências de etiquetas a, $P(a, s)$.

Um modelo discriminativo modela a dependência de uma variável não observável a a uma variável observável s. Isso é feito especificando-se distribuições de probabilidade condicional, $P(a|s)$.

Modelos gerativos são modelos de probabilidade completa de todas variáveis, enquanto modelos discriminativos são modelos apenas das variáveis não observáveis condicionadas às variáveis observáveis. Modelos gerativos podem ser usados para formar distribuições condicionais através do uso da *Regra de Bayes*. A regra (ou teorema) de Bayes diz que a probabilidade *a posteriori* $P(a|s)$ pode ser calculada a partir da verossimilhança $P(s|a)$, da probabilidade *a priori* $P(a)$, e da evidência $P(s)$:

$$P(a|s) = \frac{P(s|a)P(a)}{P(s)}.$$

A verossimilhança pode ser obtida, pela definição de probabilidade conjunta, por

$$P(s|a) = \frac{P(a,s)}{P(a)}.$$

Exemplos de modelos gerativos são: distribuição Gaussiana, distribuição multinomial, modelos ocultos de Markov (HMM), *naive* Bayes etc. Exemplos de modelos discriminativos são: máquinas de suporte vetorial (SVM), campos aleatórios condicionais (CRF), regressão logística, redes neurais etc.

2.1.3 Métodos de Aprendizado

Uma tarefa básica para ferramentas estatísticas de processamento de linguagem natural é o ajuste dos parâmetros de seu modelo. Este ajuste ou estimativa é feito por uma fase de processamento chamada *treinamento* ou *aprendizado*. A forma como esta fase é feita varia em três tipos: supervisionada, não-supervisionada e parcialmente supervisionada.

Sistemas que utilizam *aprendizado supervisionado* duplicam análises corretas a partir de dados de treinamento. Utilizam córpus pré-anotado e estimam probabilidades de diversas relações, os parâmetros, através de Máxima Verossimilhança[1] (ML) (Manning e Schütze, 1999). O problema é que a anotação manual de dados é custosa, cara, difícil de adaptar para novos objetivos – depende de linguagens, domínios etc – e faz com as pesquisas sejam guiadas pela disponibilidade de córpus, e não por tarefas. Por exemplo, o banco de árvores sintáticas *Penn Treebank* (Marcus, Santorini e Marcinkiewicz, 1994) possui 50 mil sentenças que foram anotadas manualmente durante vários anos. Além disso esses sistemas precisam se preocupar com questões de suavização de probabilidades, porque dados não vistos no treinamento não necessariamente devem possuir probabilidade zero de acontecer.

Sistemas que utilizam *aprendizado não-supervisionado* tomam dados puros e buscam detectar padrões automaticamente, induzindo estruturas ou conjuntos. Não assumem nem exploram pré-conhecimentos ou fundamentos. Os principais incentivos são que mais dados crus estão disponíveis do que anotações, e conhecimentos de outras áreas como aprendizado de máquina e aglomeração podem ser compartilhados. Entretanto, geralmente é reconhecido que esse é um problema muito mais difícil, pelo menos para o objetivo de aprender estruturas linguísticas plausíveis.

Sistemas que utilizam *aprendizado parcialmente supervisionado* tentam equilibram as vantagens e as desvantagens dos dois métodos: utilizar poucos dados pré-anotados para guiar a indução não-supervisionada. O problema desse método é que a definição de seu modelo e seus parâmetros permanece difícil.

2.1.4 Recursos

Iremos utilizar aprendizado supervisionado para estimar os parâmetros dos modelos de etiquetagem descritos nesse trabalho. Para tanto, precisamos de exemplos positivos, obtidos de um *córpus* – um conjunto de textos – previamente etiquetado por linguistas. É interessante haver, além de um córpus etiquetado para treinamento do modelo, um córpus etiquetado para teste e avaliação do etiquetador. Idealmente esse córpus de teste deve ser um conjunto disjunto do córpus de treinamento, de modo que o desempenho do modelo seja melhor avaliado em situações genéricas e, portanto, em aplicações no mundo real. A forma de avaliação é descrita na próxima seção.

1 ML, do inglês *Maximum Likelihood*.

O conjunto formado pelas palavras do córpus e por suas categorias morfossintáticas é chamado de *léxico*. Quando nos referirmos a "palavras de um córpus" estaremos nos referindo a todos os símbolos léxicos desse córpus, como por exemplo palavras, números e sinais de pontuação.

O córpus que utilizamos é o *Tycho Brahe* (IEL-UNICAMP e IME-USP, 2010), que contém vários textos do Português histórico etiquetados manualmente no formato $\langle palavra \rangle / \langle etiqueta \rangle$, e que utiliza um conjunto de 383 etiquetas[2] (Britto e Finger, 1999; Britto, Galves, Ribeiro, Augusto e Scher, 1999; Galves e Britto, 1999). Os textos selecionados estão listados no Apêndice A. Juntando-os, obtemos um total de 1.035.593 palavras e etiquetas. As etiquetas presentes nos textos selecionados são listadas no Apêndice B.

Para gerar os córpus de treinamento e de teste separamos aleatoriamente, de cada texto, três quartos das sentenças para o córpus de treinamento e o quarto restante para o córpus de teste. Assim, 75% das sentenças dos textos selecionados do Tycho Brahe formam o córpus de treinamento e os 25% restantes formam o córpus de teste. O córpus de treinamento contém 775.602 ocorrências de palavras/etiquetas, e o de teste 259.991. Há 262 diferentes etiquetas no córpus de treinamento.

Além do córpus de treinamento e teste acima, geramos novas versões deles através da segmentação das palavras e etiquetas contraídas. Palavras contraídas são palavras como *da*, que leva a etiqueta `P+D-F` e é a contração da preposição *de* (etiqueta `P`) com o determinante feminino *a* (etiqueta `D-F`). Essa segmentação é importante para nossos experimentos porque queremos utilizar também os textos do córpus Tycho Brahe que são anotados com estrutura sintática, que já estão segmentados. Para realizar a segmentação sobre o córpus normal de treinamento e de teste, desenvolvemos uma ferramenta em Python que utiliza regras – a maioria baseada em expressões regulares – que possuem diferentes prioridades de aplicação[3]. Chamaremos os córpus de treinamento e de teste resultantes de córpus segmentado. Como não há mais etiquetas compostas, há agora 161 diferentes etiquetas no córpus segmentado de treinamento.

2.1.5 Métricas

A eficiência de um etiquetador morfossintático é medida principalmente pela *taxa de acerto*:

$$\textit{Taxa de acerto} = \frac{\textit{Número de etiquetas coincidentes}}{\textit{Número total de etiquetas}},$$

ou seja, pela proporção de palavras do córpus de teste às quais o etiquetador atribui a etiqueta correta comparada ao córpus original.

Outra avaliação que pode ser efetuada é quanto ao tempo de processamento do etiquetador, tanto na fase de treinamento quanto na de teste. Esta medida é pouco encontrada na literatura da área, porque é bastante relativa à linguagem de programação utilizada e ao poder de processamento da máquina em que é executada. Nós a incluímos nos nossos resultados porque utilizamos uma máquina com configurações médias para a atualidade e porque achamos importante saber o tempo necessário para se efetuar o treinamento e a etiquetagem utilizando conjuntos grandes de texto.

2 Veja detalhes em http://www.tycho.iel.unicamp.br/~tycho/corpus/manual/tags.html.

3 Primeiro através da detecção de etiquetas compostas, e depois de palavras contraídas comuns, a ferramenta aplica as regras repetidamente, podendo assim segmentar mais de uma vez partes de uma palavra contraída. A ferramenta possui interface gráfica e pode ser facilmente adaptada a outras regras. Mais informações sobre ela podem ser obtidas por email ou em http://www.ime.usp.br/~kepler.

2.2 CADEIAS DE MARKOV: FOCO COMPUTACIONAL

Nesta seção explicamos a base da teoria sobre cadeias de Markov, para, na seção seguinte explicar, uma variante delas. Além do uso na etiquetagem morfossintática, os modelos de Markov também têm sido usados, em PLN, para modelar sequências fonéticas válidas em reconhecimento de voz (Jurafsky e Martin, 2000) e, em biologia, para modelar problemas como o do alinhamento de sequências de DNA (Brito, 2003).

Cadeias de Markov (Cinlar, 1975; Ross, 1970, 1987) foram primeiramente desenvolvidas por Andrei A. Markov para modelar as sequências de letras em trabalhos de literatura Russa, e desde então se desenvolveram como uma ferramenta estatística geral. Suponha que $X = (X_1, \ldots, X_T)$ seja uma sequência de variáveis aleatórias com valores num conjunto finito $S = (s_1, \ldots, s_N)$, o espaço de estados. X é dita ser uma cadeia de Markov se tiver a propriedade do *horizonte limitado*,

$$P(X_{t+1} = s_k | X_1, \ldots, X_t) = P(X_{t+1} = s_k | X_t), \tag{2.4}$$

e da *invariante de tempo*,

$$P(X_{t+1} = s_k | X_t) = P(X_2 = s_k | X_1),$$

Essas propriedades especificam que, numa cadeia assim, a probabilidade de uma variável aleatória ter um determinado valor depende apenas do valor da variável imediatamente anterior, e que essa dependência não se altera em razão do tempo. Ou em outras palavras, que a probabilidade de estarmos em um determinado estado, depois de termos passado por vários outros estados, depende apenas do estado em que estávamos imediatamente antes de virmos para o estado atual.

Podemos descrever uma cadeia de Markov por uma matriz de probabilidades de transição M:

$$m_{ij} = P(X_{t+1} = s_j | X_t = s_i)$$

Aqui, $m_{ij} \geqslant 0, \forall i, j$ e $\sum_{j=1}^{N} m_{ij} = 1, \ \forall i$.

Dadas as definições acima, podemos descrever um modelo de Markov como uma tripla $\mu = (S, M, \Pi)$, onde S é o espaço de estados, M a matriz de transição e Π um vetor de probabilidades iniciais.

Para fixar a idéia, considere um modelo de Markov de 3 estados do clima. Assumimos que uma vez por dia o tempo climático é observado como sendo um desses:

Estado 1: chuvoso

Estado 2: nublado

Estado 3: ensolarado

Postulamos que o clima no dia t é caracterizado por um único dos três estados acima, e que a matriz M de probabilidades de transição é dada por

$$M = \{m_{ij}\} = \begin{bmatrix} 0{,}4 & 0{,}3 & 0{,}3 \\ 0{,}2 & 0{,}6 & 0{,}2 \\ 0{,}1 & 0{,}1 & 0{,}8 \end{bmatrix}$$

Dado que o tempo no dia 1 ($t = 1$) está ensolarado (estado 3), podemos perguntar: Qual é a probabilidade (de acordo com o modelo) de que o tempo para os

próximos 7 dias seja "ensolarado-ensolarado-chuvoso-chuvoso-ensolarado-nublado-ensolarado"? Definindo esta sequência de estados como O,

$$O = (S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3),$$

queremos determinar a probabilidade de O, dado que o estado inicial é S_3. Esta probabilidade pode ser expressa como

$$\begin{aligned} P(O|Modelo) &= P[S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3|Modelo] \\ &= P[S_3] \cdot P[S_3|S_3] \cdot P[S_3|S_3] \cdot P[S_1|S_3] \cdot P[S_1|S_1] \\ &\quad \cdot P[S_3|S_1] \cdot P[S_2|S_3] \cdot P[S_3|S_2] \\ &= 1 \cdot a_{33} \cdot a_{33} \cdot a_{31} \cdot a_{11} \cdot a_{13} \cdot a_{32} \cdot a_{23} \\ &= 1 \cdot (0{,}8)(0{,}8)(0{,}1)(0{,}4)(0{,}3)(0{,}1)(0{,}2) \\ &= 1{,}535 \times 10^{-4} \end{aligned}$$

Podemos, também, representar uma cadeia de Markov por um diagrama de estados, como na Figura 1. Os estados são mostrados como círculos ao redor de seu

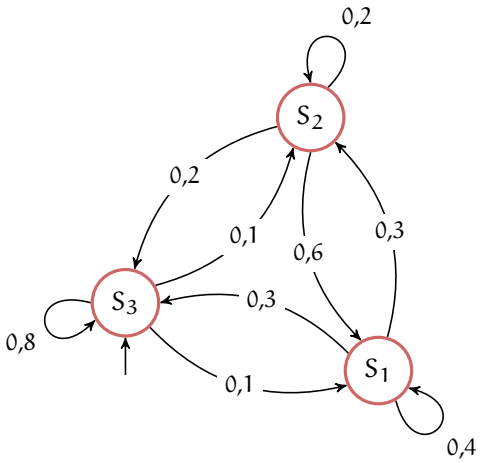


Figura 1: Um modelo de Markov.

nome, e o estado inicial é indicado por uma seta de entrada. As transições possíveis são mostradas por arcos dirigidos conectando estados, e estes arcos são rotulados com a probabilidade desta transição ser tomada. Transições com probabilidade zero são omitidas. As probabilidades dos arcos de saída de cada estado somam 1. A partir desta representação, podemos perceber que um modelo de Markov pode ser visto como um autômato de estados finitos não-determinístico com probabilidades em cada arco.

Em um etiquetador que usa modelos de Markov olhamos para a sequência de etiquetas em um texto como uma cadeia de Markov. Pela propriedade do horizonte limitado assumimos que a etiqueta de uma palavra só depende da etiqueta anterior. Usando a notação da Tabela 1, podemos escrever a propriedade do Horizonte Limitado assim:

$$P(t_{i+1}|t_{1,i}) = P(t_{i+1}|t_i).$$

Apesar dessa propriedade, podemos considerar que um estado é composto por duas etiquetas, e assim estarmos olhando duas etiquetas para trás. Uma cadeia de Markov dessa forma é dita ser de ordem dois. O mesmo vale para ordens n.

Cadeias de Markov como explicamos são chamadas de visíveis, porque ao fim de uma sequência de eventos sabemos exatamente por quais estados passamos, e a probabilidade final da sequência é dada pela multiplicação das probabilidades dos estados pelos quais passamos. Entretanto, esse tipo de modelagem é restritivo demais para ser aplicável a problemas mais complexos. Uma variação menos restritiva é dada pelos Modelos de Markov Ocultos (HMMs). Neles, eventos profundos geram probabilisticamente eventos de superfície. Dessa forma, ao passar de um estado para outro com uma certa probabilidade, emitimos um determinado símbolo, também com certa probabilidade, e ao final o que temos é apenas a sequência de símbolos emitidos e sua probabilidade total. Não sabemos qual é a sequência de estados que o modelo percorre, apenas alguma função probabilística dela. Algoritmos existem para calcular a probabilidade de um modelo ter gerado uma dada sequência de símbolos, para calcular a sequência mais provável de estados percorridos na emissão desta sequência, e para treinar o modelo de modo a maximizar a probabilidade desta sequência.

Esse tipo de modelo é o mais largamente usado, principalmente em etiquetadores morfossintáticos estatísticos. Nele, os estados correspondem às etiquetas e as palavras aos símbolos emitidos. Embora tenha alcançado resultados satisfatórios, convergindo em um nível de acerto em torno de 97% para o inglês e pouco menos que isso para o português, ainda são restritivos para esta tarefa: tentam distinguir a classe de uma palavra geralmente considerando no máximo as duas etiquetas anteriores – ou seja, usam cadeias de ordem 2 (i.e., trigramas) – e às vezes as duas posteriores.

Cadeias de ordens maiores não são usadas porque acarretam uma explosão no tamanho do espaço de estados, o que aumenta a complexidade computacional e a esparsidade dos dados: mais dados são necessários para que probabilidades consistentes sejam encontradas – caso contrário muitas sequências terão probabilidade zero – e consequentemente ainda mais poder computacional é necessário.

Entretanto, diversas palavras só podem ter seus significados distinguidos se etiquetas anteriores às duas últimas forem consideradas. Mais ainda, diversas palavras dependem de palavras que ocorrem bastante antes na sentença, como é o caso do *"mais que"*: muitas outras palavras podem existir entre *mais* e *que*, sem alterar o sentido de *que*. Por exemplo, *mais* e *que* tem a mesma classificação morfossintática em qualquer dos casos abaixos:

- *... comeu* ***mais que*** *podia ...*
- *... era* ***mais*** *esperto* ***que*** *inteligente ...*
- *... muito* ***mais*** *exausto, abatido e desanimado* ***que*** *nunca ...*
- ***Mais*** *vale um asno que me carregue* ***que*** *um cavalo que me derrube.*

Embora a classificação correta de *mais* e *que* nas duas primeiras sentenças seja mais fácil de ser realizada – *que* não está a mais de duas palavras de *mais*, portanto um etiquetador clássico utilizando cadeias de Markov de ordem 2 pode detectar isto –, nas duas últimas sentenças ela não é tão trivial. Em especial, note na última sentença os *que*'s intermediários.

Como a taxa de acerto dos etiquetadores convergiu a um valor inferior a 100% ($\sim$ 96%), intuitivamente uma taxa de acerto extra poderia ser alcançada se fosse considerado um contexto maior na etiquetagem destas palavras ambíguas. Procurando

sanar esta restrição de tamanho e detectar dependências longas entre palavras, apresentamos na próxima seção outro modelo de Markov, que implementamos e testamos para o português histórico do Brasil em Kepler (2005).

2.3 CADEIAS DE MARKOV DE TAMANHO VARIÁVEL

Cadeias de Markov de Tamanho Variável (VLMCs) permitem que a memória de uma cadeia de Markov tenha tamanho (ou alcance) variável dependendo dos valores passados observados. Explicaremos o conceito básico a seguir. Uma descrição formal e com provas matemáticas de sua convergência pode ser encontrada em Bühlmann e Wyner, 1999; e Mächler e Bühlmann, 2002.

Para começar, considere uma cadeia de Markov de ordem finita k. Então, pela propriedade do horizonte limitado, vista na Equação 2.4,

$$P(t_i|t_{i-\infty,i-1}) = P(t_i|t_{i-k,i-1}), \forall t_{i-\infty,i}.$$

Uma memória de tamanho variável pode ser vista como um corte de estados irrelevantes do histórico $t_{i-k,i-1}$. O conjunto de estados remanescentes é chamado de *contexto* de t_i.

Seja X um processo estacionário – assim como as cadeias de Markov de ordem fixa – com valores $t_i \in \mathcal{T}$, $|\mathcal{T}| < \infty$, $\mathcal{T}$ o conjunto de etiquetas possíveis. Seja $c : \mathcal{T} \to \mathcal{T}$ uma função que mapeia

$$c : t_{i-\infty,i-1} \mapsto t_{i-h,i-1},$$

onde h é definido por

$$h = h(t_{i-\infty,i-1}) = \min\{k|P(t_i|t_{i-\infty,i-1}) = P(t_i|t_{i-k,i-1})\ \forall t_i \in \mathcal{T}\},$$

onde $h \equiv 0$ corresponde à independência. Então, $c(\cdot)$ é chamada de uma função contexto e para qualquer $i < |\mathcal{T}|$, $c(t_{i-\infty,i-1})$ é chamada de função contexto para a variável t_i.

Seja $0 \leqslant k \leqslant \infty$ o menor inteiro tal que

$$|c(t_{i-\infty,i-1})| = h(t_{i-\infty,i-1}) \leqslant k, \forall t_{i-\infty,i-1} \in \mathcal{T}. \tag{2.5}$$

Então $c(\cdot)$ é dita uma função contexto de ordem k, e se $k < \infty$, X é dita uma cadeia de Markov de tamanho variável de ordem k. A probabilidade $P(t_i|c(t_{i-k,i-1})$ então fornece a probabilidade de t_i dado seu contexto relevante de tamanho até k.

Geralmente, o espaço da função contexto $c(\cdot)$ não é o espaço completo $\mathcal{T}^k$. Neste caso a VLMC de ordem k é uma cadeia de Markov de ordem k, mas que tem *memória de tamanho variável* h. Do contrário, a VLMC é uma cadeia de Markov completa de ordem k.

2.3.1 Representação do Espaço de Estados

Os estados que determinam as probabilidades de transição da VLMC são dados pelos valores da função contexto $c(\cdot)$. Representamos estes estados – o espaço de estados minimal da VLMC – como uma árvore.

Uma *árvore de contexto* é uma árvore com um nó raiz no topo, de onde descem ramificações, tal que cada nó interno da árvore possui no máximo $|\mathcal{T}|$ filhos. Veja a Figura 2. Cada valor de uma função contexto $c(\cdot)$ pode ser representado como um

galho (i.e., um caminho que vai da raiz até uma folha) de tal árvore. O contexto $c(t_{i-\infty,i-1})$ é representado por um galho, cujo sub-galho no topo é determinado por t_{i-1}, o próximo sub-galho por t_{i-2} e assim por diante, até a folha, determinada por $t_{i-h(t_{i-\infty,i-1})}$.

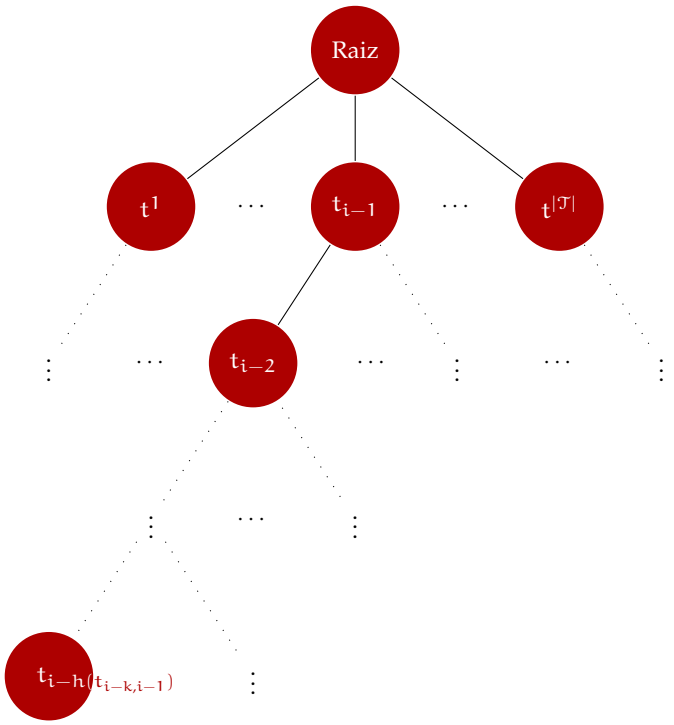


Figura 2: Definição de uma *Árvore de Contexto*.

Um exemplo de uma árvore de contexto contendo etiquetas morfossintáticas é dado na Figura 3. A partir da raiz estão as etiquetas mais recentes de um contexto, descendo pelos nós para as mais antigas, até as últimas consideradas em um contexto. Cada nó fornece a probabilidade de uma etiqueta t dado que o contexto ocorrido é a seqüência formada por este nó mais os nós acima dele. Assim, se tivermos a frase

> *A criança correu rapidamente para casa quando começou a chover*.

e já tivermos etiquetado até *casa*, em

> *A*/`D` *criança*/`N` *correu*/`VB` *rapidamente*/`ADV` *para*/`P` *casa*/`N` *quando começou a chover*.

a probabilidade de que a etiqueta t venha em seguida é dada pelo nó `VB` mais abaixo na árvore: P(t|`VB,ADV,P,N`)[4]. O restante do histórico, `D,N`, não é considerado relevante, por isso não está na árvore.

Na próxima seção introduzimos um algoritmo de contexto que constrói uma destas árvores para uma VLMC. Por isto, é conveniente darmos a noção de uma árvore de contexto terminal. Seja τ uma árvore de contexto como definida acima. Então uma árvore de contexto terminal τ^T é uma árvore que contém somente nós terminais, i.e. folhas, de τ.

4 Para efeitos de exemplo não estamos considerando a palavra a ser etiquetada, *quando*.

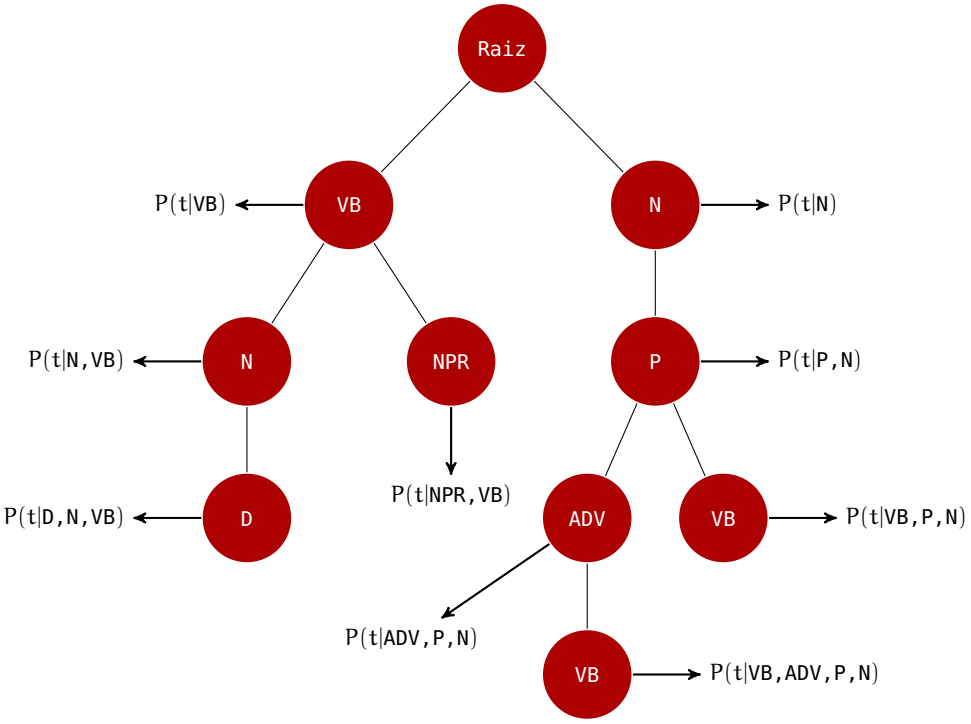


Figura 3: Exemplificação de uma árvore de contexto.

2.3.2 Algoritmo de Contexto

Dada uma VLMC, o objetivo é encontrar a função contexto $c(\cdot)$ subjacente e suas probabilidades. Para resolver essa questão usamos uma versão do algoritmo de contexto de Rissanen (1983). Primeiro, construímos uma grande árvore de contexto, utilizando para isso o córpus de treinamento. Depois, o algoritmo utiliza uma função reversa de poda de árvore considerando um critério de decisão local.

Convencionamos que quantidades envolvendo índices de tempo fora de $\{1, \ldots, n\}$ são iguais a zero (ou irrelevantes). Seja $C(v)$ o número de ocorrências da etiqueta v na seqüência $t_{1,n}$. Ainda, seja

$$\hat{P}(v) = C(v)/n, \ \hat{P}(u|v) = \frac{C(uv)}{C(v)}, \ v, u \in \mathcal{T}^+, \tag{2.6}$$

$\mathcal{T}^+$ o conjunto de todas seqüências não nulas de etiquetas de $\mathcal{T}$. O algoritmo de contexto para VLMCs é mostrado no Algoritmo 1.

Para obter o contexto de uma palavra, primeiro construímos uma árvore com todo passado possível, e depois vamos efetuando podas nesta árvore. Fazemos isto comparando a probabilidade de um contexto com sua probabilidade sem a última etiqueta. Se a diferença não é significativa, podamos do contexto esta última etiqueta. A relevância desta diferença nas probabilidades é dada pelo valor de corte K, determinado empiricamente.

2.3.3 Etiquetagem Morfossintática Baseada em VLMCs

Em Kepler (2005) implementamos um etiquetador morfossintático, que chamaremos de VLMC TAGGER, que utiliza cadeias de Markov de tamanho variável. O número

Entrada: d: densidade da árvore de contexto.
Entrada: K: valor de corte.

PASSO 1 Dadas as etiquetas $t_1, \ldots, t_n$ em $\mathcal{T}$, busque a função contexto $c_{max}(\cdot)$ com árvore de contexto terminal τ^T_{max}, onde τ^T_{max} é a maior árvore tal que toda folha em τ^T_{max} tenha sido observada pelo menos d vezes em $t_1, \ldots, t_n$. Faça $\tau^T_i = \tau^T_{max}$, i o número da iteração sendo executada.

PASSO 2 Examine cada elemento (folha) de τ^T_i como segue. Seja $c(\cdot)$ a função contexto correspondente de τ^T_i e seja

$$vu = t_{i-h+1,i} = c(t_{i-\infty,i}),\ u = t_{i-h+1},\ v = t_{i-h+2,i},$$

onde vu é um elemento (folha) de τ^T_i, o qual comparamos com sua versão podada $v = t_{i-h+2,i}$. Se $h = 1$, a versão podada é o ramo vazio, isto é, a raiz.

Faça a poda de $vu = t_{i-h+1,i}$ para $v = t_{i-h+2,i}$ se

$$\Delta_{vu} = \sum_{t \in \mathcal{T}} P(t|vu) \log\left(\frac{P(t|vu)}{P(t|v)}\right) C(vu) < K, \tag{2.7}$$

onde $P(\cdot|\cdot)$ é como em 2.6. A decisão sobre podar para cada elemento leva possivelmente a uma árvore menor τ^T_{i+1}. Construa tal árvore de contexto terminal.

PASSO 3 Repita o **PASSO 2** para $i = 1,2,\ldots,$ até que mais nenhuma poda seja possível. Denote esta árvore de contexto podada maximal por $\hat{\tau}$, e sua função contexto correspondente por $\hat{c}(\cdot)$.

PASSO 4 Estime as probabilidades de transição $P(t_i|c(t_{i-\infty,i-1}))$ por $\hat{P}(t_i|\hat{c}(t_{i-\infty,i-1}))$.

Algoritmo 1: Algoritmo de Contexto.

considerado de etiquetas à esquerda de uma palavra varia de palavra para palavra, e de contexto para contexto. Com isso não há um aumento tão expressivo de complexidade, já que o custo de analisar contextos longos é equilibrado com o de contextos curtos, que são mais freqüentes.

Para uma VLMC de ordem k, a probabilidade de uma etiqueta, dada a palavra atual e o contexto, é obtida usando a seguinte equação:

$$P(t_i|w_i, c(t_{i-k,i-1})) = P(w_i|t_i)P(t_i|c(t_{i-k,i-1})). \tag{2.8}$$

A probabilidade de uma sequência de etiquetas associada a uma sequência de palavras, reescrevendo a Equação 2.2, é obtida por:

$$P(w_{1,n}, t_{1,n}) = \prod_{i=1}^{n} P(t_i|c(t_{i-k,i-1}))P(w_i|t_i). \tag{2.9}$$

Para encontrar a melhor sequências de etiquetas $t_{1,n}$ para uma dada sequência de palavras $w_{1,n}$ de tamanho n, usamos como algoritmo de busca o *algoritmo de*

Viterbi (Viterbi, 1967). A equação de melhor sequência, mostrada em 2.3, é redefinida com base na Equação 2.9:

$$\arg\max_{t_{1,n}} \prod_{i=1}^{n} P(t_i|c(t_{i-k,i-1}))P(w_i|t_i).$$

2.3.4 Resultados Anteriores

O VLMC TAGGER, desenvolvido em Kepler (2005)[5], foi implementado em C++ utilizando a STL (*Standard Template Library*), e compilado com o g++ versão 3.3.4. Os testes foram feitos em uma máquina com processador Intel de 3.0 GHz e com 1 GB de memória RAM. Os córpus de treinamento e de teste normais foram obtidos do córpus Tycho Brahe, conforme descrito na Seção 2.1.4.

Executamos conjuntos de testes variando o tamanho do córpus de treinamento, escolhendo 5%, 10%, ..., 95%, 100% de suas sentenças e executando 10 vezes com cada um desses tamanhos, escolhendo aleatoriamente as sentenças em cada vez. O córpus de teste foi sempre usado sem modificações.

O gráfico da Figura 4 mostra a taxa de acerto em cada iteração de teste conforme descrito acima. Cada ponto ao redor da linha indica a taxa de acerto de uma iteração de teste, e a linha passa pela taxa média de acerto de cada conjunto de iterações. A taxa de acerto final, usando 100% do córpus de treinamento, foi igual a 95,51% A taxa de acerto para palavras desconhecidas, não mostrada no gráfico, foi de 69,53%. Vemos que, quanto maior o número de palavras do córpus de treinamento, menor é

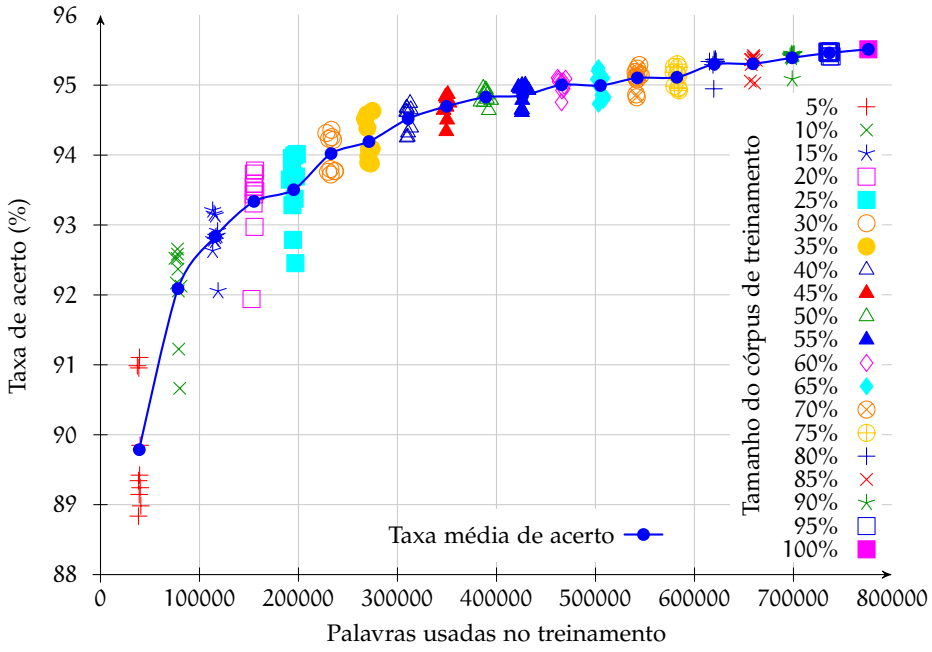


Figura 4: Distribuição das precisões dos testes do VLMC TAGGER.

a diferença de resultado entre iterações de teste de um mesmo conjunto. Entretanto, embora os resultados possam convergir, isto também mostra que o etiquetador é

5 Disponível sob licença GPL em http://www.ime.usp.br/~kepler/vlmmtagger.

sensível não apenas à quantidade de palavras de treinamento, mas também à escolha das sentenças. Algumas iterações de teste alcançaram resultados muito melhores ou muito piores que a média, indicando a existência de sentenças que melhoram o aprendizado do etiquetador ou que o prejudicam.

O gráfico da Figura 5 mostra três curvas dos tempos tomados pelo VLMC TAGGER em relação ao número de palavras usadas para treinamento: o tempo total de execução, o tempo de treinamento e o tempo de etiquetagem (teste)[6]. O tempo total de execução usando 100% do córpus de treinamento é de 157 segundos. Além disso, as três curvas apresentam comportamento linear, e de fato a curva do tempo médio de execução possui correlação igual a 0,9969.

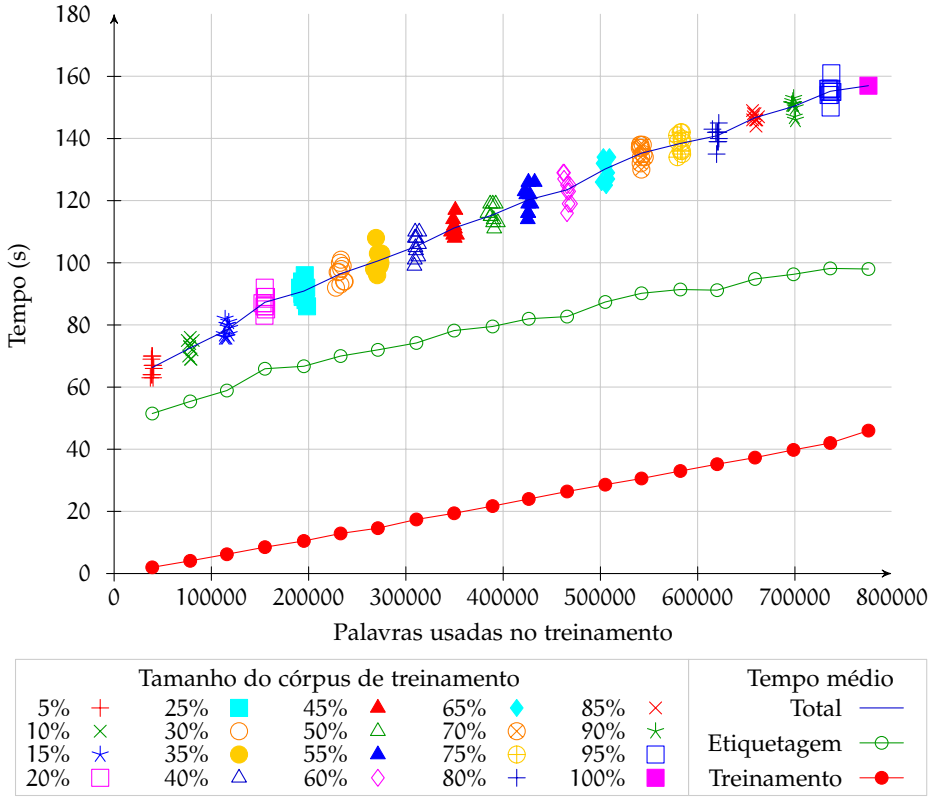


Figura 5: Distribuição dos tempos de execução do VLMC TAGGER.

A Figura 6 mostra, para os diversos tamanhos do córpus de treino, o número de galhos de tamanhos distintos contidos na árvore de contexto depois da execução do algoritmo de contexto (Algoritmo 1). Nessa figura, vemos o fato interessante de que a maioria dos galhos da árvore de contexto tem tamanho dois, mesmo que o número de galhos maiores também aumente em relação ao tamanho do córpus de treino. Considerando o córpus inteiro, a árvore de contexto do etiquetador armazena por volta de 675 galhos de tamanho dois.

Outro resultado obtido foi o número de erros cometidos para cada palavra do córpus de teste. Ordenando-se esses números em ordem decrescente, as duas primeiras

6 Note que o tempo total de execução é maior do que a soma dos tempos de treinamento e etiquetagem porque inclui o tempo tomado por outras operações como a leitura dos arquivos do córpus e o cálculo da taxa de acerto.

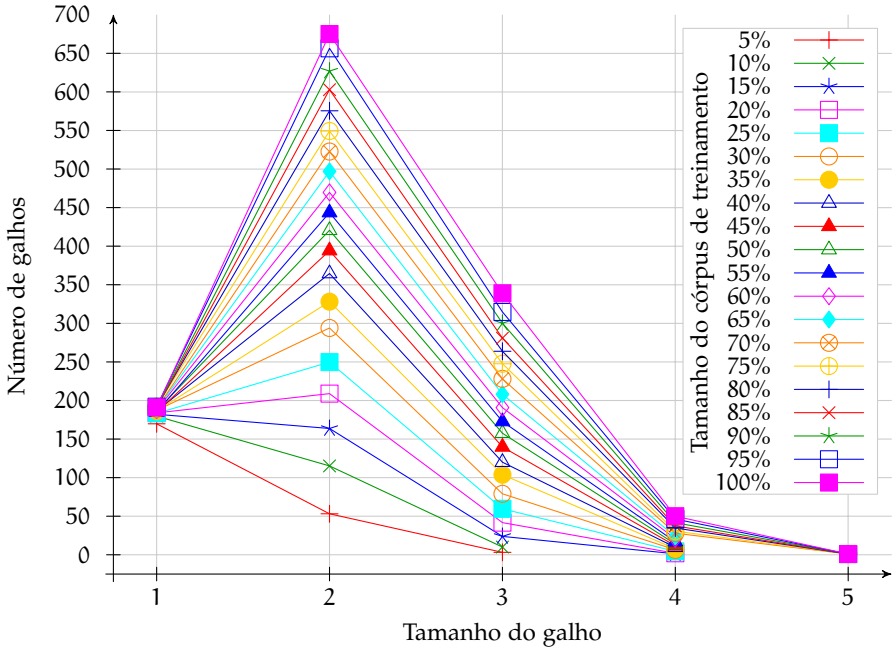


Figura 6: Proporção de galhos da árvore de contexto do VLMC TAGGER.

palavras em número de erros são *que* e *a*. Dos 10658 erros cometidos na etiquetagem, *que* foi errada 2266 vezes, e *a* 1115 vezes. Os erros cometidos com essas duas palavras correspondem a quase um terço do total de erros. Descrevemos esses erros mais adiante.

2.3.5 Conclusões

A taxa de acerto obtida pelo VLMC TAGGER se aproxima dos melhores resultados reportados para o português (Aires, 2000; Finger, 2000; Santos, Milidiú e Rentería, 2008). Já o tempo de execução usando mais de 775.000 palavras para treinamento e quase 260.000 para teste, num total de 1.035.593 palavras, é muito inferior aos existentes na literatura, embora nem todos reportem seus tempos e eles sejam relativos à máquina usada.

Quando o etiquetador foi modelado para considerar contextos mais longos, ele não foi capaz de detectar muitas dependências de longa distância. Mais ainda, instruí-lo a considerar contextos não tão longos melhorou a taxa de acerto. Provavelmente porque, quando possui menos contextos longos disponíveis, o etiquetador escolhe contextos curtos e recentes, o que mostra que existem contextos longos que não auxiliam a escolha correta das etiquetas, mas ao invés disto atrapalham essa escolha. Talvez isto se deva à esparsidade dos dados: dependendo dos parâmetros de configuração do etiquetador, como o valor de corte K, alguns contextos longos que não são muito consistentes permanecem na árvore de contexto. Nesse caso mais dados para treinamento poderiam suavizar este problema. Entretanto, dados de treinamento podem nunca ser suficientes, e portanto precisamos considerar isso como uma limitação do etiquetador.

Dessa forma, baseados nos resultados, podemos observar vantagens e limitações da aplicação à análise morfossintática de modelos estatísticos baseados em VLMCs: eles aprendem vários contextos de curta e média distância ($d \leqslant 5$), que já são maiores que os tradicionais contextos de tamanho 2, mas não possuem capacidade de generalização para aprender fenômenos linguísticos que ocorrem em contextos variáveis e talvez distâncias maiores. Dependências como *"mais ... que"* não são capturadas, porque o contexto entre as palavras é muito variável tanto em tamanho quanto em composição. Sendo os dados de treinamento limitados, o problema da esparsidade parece ser agravado para esse tipo de contexto. Portanto, parâmetros usando VLMCs para modelar esse tipo de dependência não parecem ser os mais adequados. Outras abordagens precisam ser consideradas. Faremos isso na Parte ii. Antes, mostraremos alguns casos que dificultam a tarefa da etiquetagem.

3 CASOS DIFÍCEIS NA ETIQUETAGEM

Usando como base o etiquetador desenvolvido em Kepler (2005), o VLMC TAGGER, alteramos e reimplementamos diversos parâmetros, estruturas e algoritmos para possibilitar a exploração de novos modelos baseados em VLMCs. Chamaremos esse modelo atualizado de Modelo de Markov de Tamanho Variável[1] (VLMM), já que ele apenas usa VLMCs como parte de seus parâmetros. A implementação desse modelo é dada pelo VLMM TAGGER.

Alguns dos textos do córpus Tycho Brahe foram atualizados, ou seja, foram revistos manualmente. Atualizamos o córpus normal de treinamento e de teste com esses textos revisados, e a partir dele geramos o córpus segmentado. Essas versões atualizadas são as utilizadas no restante deste trabalho.

Usando o modelo VLMM, a taxa de acerto sobre o córpus normal aumentou dos 95,51%, obtidos pelo VLMC TAGGER, para 96,40%. Uma das principais diferenças foi em relação à taxa de acerto sobre palavras desconhecidas, que subiu de 69,53% para 74,66%, o que representa aproximadamente 17% de redução de erro. A Tabela 2 mostra outros números obtidos.

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	84,7775	15,2225	1683	11056
a	90,9726	9,0274	659	7300
Desconhecidas	74,6631	25,3369	2256	8904
Conhecidas	97,1787	2,8213	7084	251087
Total	96,4076	3,5924	9340	259991

Tabela 2: Resultados do modelo VLMM sobre o córpus normal.

Mesmo com uma taxa de acerto maior, os resultados mostram que *a* e *que* são novamente responsáveis por uma grande parcela dos erros: 25%. A Tabela 3 lista as dez palavras com o maior número de erros cometidos no córpus de teste normal. Tirando *que* e *a*, os erros somados das outras oito palavras correspondem a apenas 10% do total de erros.

A Tabela 4(a) lista as 15 palavras mais frequentes no córpus normal. Dessas 15, apenas 4 aparecem na lista das 10 palavras com mais erros: *que*, *a*, *se* e *o*. As outras não são ambíguas ou possuem grau baixo de ambiguidade, como *os*.

A Tabela 5 mostra a matriz de confusão das etiquetas de *que*. Uma matriz de confusão indica, de forma geral, quais valores são os corretos e quais foram preditos. No nosso caso, as linhas indicam as etiquetas corretas, e as colunas as etiquetas que foram atribuídas pelo etiquetador. Os principais erros cometidos com *que* são a troca entre as etiquetas `WPRO`, que denota pronome relativo, e `C`, que representa conjunção subordinativa. As etiquetas minoritárias, `FW`, `D` e `WD-P`, são casos especiais. A etiqueta `FW`, que indica palavra estrangeira, ocorre em *que* dentro de um contexto de outras etiquetas `FW`, em sentenças em francês. Já `D` e `WD-P` claramente são erros de anotação do córpus:

1 VLMM, do inglês *Variable-Length Markov Model*.

PALAVRA	ERROS	OCORRÊNCIAS	% ERROS
que	1683	11056	15,22
a	659	7300	9,03
se	260	3342	7,78
porque	128	894	14,32
como	115	1351	8,51
nem	106	619	17,12
o	90	5983	1,50
foi	58	393	14,76
até	56	231	24,24
as	55	1923	2,86
tal	48	157	30,57
Que	47	135	34,81

Tabela 3: Palavras com maior número de erros no córpus de teste normal.

ORDEM	OCORR.	PALAVRA
1	21454	,
2	11056	*que*
3	8932	*e*
4	7961	*de*
5	7300	*a*
6	6720	.
7	5983	*o*
8	3342	*se*
9	3072	*não*
10	2582	;
11	2565	*em*
12	2503	*os*
13	2465	*com*
14	2226	*do*
15	2198	*da*

(a) No córpus normal.

ORDEM	OCORR.	PALAVRA
1	21454	,
2	15720	*de*
3	13597	*a*
4	11063	*que*
5	10706	*o*
6	8932	*e*
7	6720	.
8	5738	*em*
9	4419	*os*
10	4188	*se*
11	3231	*as*
12	3072	*não*
13	2582	;
14	2555	*com*
15	2466	*por*

(b) No córpus segmentado.

Tabela 4: Palavras com maior número de ocorrências nos córpus de teste.

- "*O*/`D` *mesmo*/`ADJ` *que*/`D` *era*/`SR-D` ".
- "*ter*/`TR` *presente*/`ADJ-G` *que*/`D` *a*/`D-F` *vil*/`ADJ-G` ".
- "*que*/`WD-P` *ofícios*/`N-P` *fizeram*/`VB-D` ".

Para *a*, os principais erros são entre `D-F` e `P`, determinante feminino e preposição, conforme mostra a matriz de confusão das etiquetas de *a* na Tabela 6.

O resultado do modelo VLMM sobre o córpus segmentado é mostrado na Tabela 7. O número de ocorrências de palavras desconhecidas cai ao redor de 10%, o que ajuda a fazer a taxa de acerto sobre elas subir 0,25%. Há, obviamente, um aumento no número de palavras, já que palavras como *da*, por exemplo, quando segmentadas, se tornam duas, *de* e *a*. No total, mesmo com 16550 palavras a mais, a taxa de acerto sobre o córpus segmentado aumenta de 96,40% para 96,62%.

	WPRO	C	WD	CONJ	FW	D	WD-P	% PREC.
WPRO	<5569>	561	13	9	.	.	.	90,5234
C	833	<3724>	6	31	.	.	.	81,0623
WD	14	19	<68>	4	.	.	.	64,7619
CONJ	99	91	.	<8>	.	.	.	4,0404
FW	.	.	.	.	<4>	.	.	100,0000
D	.	2	.	.	.	<.>	.	0,0000
WD-P	.	.	1	.	.	.	<.>	0,0000

Tabela 5: Matriz de confusão para a palavra *que* no córpus normal (linha: referência; coluna: teste).

	D-F	P	CL	FW	D	N	% PREC.
D-F	<4023>	214	3	.	.	1	94,8597
P	395	<2317>	7	.	.	.	85,2152
CL	19	14	<296>	.	.	.	89,9696
FW	2	.	.	<5>	.	.	71,4285
D	3	.	.	.	<.>	.	0,0000
N	1	.	.	.	.	<.>	0,0000

Tabela 6: Matriz de confusão para a palavra *a* no córpus normal (linha: referência; coluna: teste).

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	84,5250	15,4750	1712	11063
a	94,5576	5,4424	740	13597
Desconhecidas	74,9214	25,0786	1995	7955
Conhecidas	97,2642	2,7358	7348	268586
Total	96,6215	3,3785	9343	276541

Tabela 7: Resultados do modelo VLMM no córpus segmentado.

O número de ocorrências de *a* quase duplica em relação ao córpus normal: 13597 contra 7300. É interessante notar que, mesmo com esse aumento de ocorrências, o número de erros sobre *a* sobe de 659 no córpus normal para apenas 740 no córpus segmentado, o que implica em uma melhora na taxa de acerto sobre *a*. O número de ocorrências de *de* e de *o* também aumenta bastante, como pode ser visto na Tabela 4(b). Mas *de* praticamente não apresenta ambiguidade e *o* tem a taxa de erro aumentada para apenas 1,88%. O número de ocorrências de *que* permanece praticamente o mesmo, com a taxa de acerto piorando um pouco, mas não significativamente.

As Tabelas 8 e 9 mostram as matrizes de confusão das etiquetas de *que* e *a*, respectivamente. A confusão sobre as etiquetas de *que* no córpus segmentado não muda muito em relação ao córpus normal. Já para *a* há um aumento mais significativo nos erros em relação à etiqueta CL (clítico), que aumentaram bastante em número após a segmentação.

	WPRO	C	WD	CONJ	FW	WD-P	D	% ACERTO
WPRO	<5562>	575	14	8	.	.	.	90,3069
C	850	<3712>	6	26	.	.	.	80,8010
WD	13	20	<68>	4	.	.	.	64,7619
CONJ	101	92	.	<5>	.	.	.	2,5253
FW	.	.	.	.	<4>	.	.	100,0000
WD-P	.	.	1	.	.	<.>	.	0,0000
D	.	2	.	.	.	.	<.>	0,0000

Tabela 8: Matriz de confusão da palavra *que* no córpus segmentado (linha: referência; coluna: teste).

	D-F	P	CL	FW	D	D-P	N	% ACERTO
D-F	<8056>	206	8	.	.	.	.	97,4123
P	433	<4411>	17	.	.	.	.	90,7426
CL	29	39	<386>	.	.	.	.	85,0220
FW	2	1	.	<4>	.	.	.	57,1428
D	3	.	.	.	<.>	.	.	0,0000
D-P	1	.	.	.	.	<.>	.	0,0000
N	1	.	.	.	.	.	<.>	0,0000

Tabela 9: Matriz de confusão da palavra *a* no córpus segmentado (linha: referência; coluna: teste).

3.1 REFINAMENTO DOS OBJETIVOS DESTE TRABALHO

As palavras *que* e *a* são, nos nossos testes, as palavras com o maior número de erros cometidos pelo VLMM TAGGER, somando juntas um quarto de todos os erros. Embora não sejam as palavras que possuem a maior porcentagem de erro, são duas palavras que ocorrem com muita frequência no português, e portanto classificá-las corretamente ajudaria tanto na taxa de acerto geral quanto possivelmente na taxa de acerto de outras palavras. Claramente, uma frequência maior de uma palavra ambígua ajuda a explicar um maior número de erros sobre essa palavra. Mas também ajuda a descartar o problema da esparsidade como causa desses erros. Dessa forma, o problema não está nas estimativas, mas sim nos parâmetros usados no modelo para prever *que* e *a*. Isso implica que o contexto local esquerdo, mesmo que variavelmente longo, não é suficiente para classificar *que* e *a* corretamente.

A maior confusão para cada uma dessas palavras é entre duas etiquetas diferentes:

- A palavra *que* é, na maioria das vezes, ou um pronome relativo – denotado pela etiqueta `WPRO` – ou uma conjunção subordinativa – denotada pela etiqueta `C`;
- A palavra *a* é, geralmente, ou um determinante feminino – representado pela etiqueta `D-F` – ou uma preposição – denotada pela etiqueta `P`.

Considerando casos de uso de *que*, parece que palavras que ocorrem não imediatamente antes, ou seja, não locais a *que*, podem acrescentar informação importante. Por exemplo, se *que* seguir *mais*, é bem provável que *que* tenha etiqueta `C`. Entretanto, é possível que haja várias diferentes palavras entre *mais* e *que*, como por exemplo:

"*mais provável que*";
"*mais caro e complexo que*";

e assim por diante. Assim, melhores resultados poderiam ser produzidos se o contexto não local à esquerda, ou seja, o contexto não adjacente, pudesse ser eficientemente modelado.

Já a palavra *a* poderia ser melhor classificada olhando-se as etiquetas que a seguem, ou seja, à direita. Por exemplo, se seguida por um verbo, *a* é muito mais provavelmente uma preposição.

Portanto, essas duas palavras mostram dois tipos diferentes de ambiguidade: um que precisa de contexto à direita, e outro que precisa de contexto não local. O modelo VLMM não possui parâmetros para esses contextos, já que ele faz a etiquetagem da esquerda para a direita usando o contexto imediatamente à esquerda. Para explorar essas idéias sobre as dependências de *que* e *a*, e prová-las verdadeiras ou falsas, nesse trabalho exploraremos maneiras de expandir o etiquetador VLMM com diferentes modelos e métodos que potencialmente ajudem a resolver essas questões.

Parte II

ABORDAGENS

4 GENERALIZAÇÃO DE CONTEXTO VIA ESTRUTURA SINTÁTICA

As VLMCs usam apenas contexto local esquerdo, e apesar de serem capazes de utilizar contextos longos, algumas dependências não são capturadas. Uma idéia para tentar modelar dependências não-locais é utilizar informações sobre a estrutura sintática da sentença. Essa estrutura poderia ajudar a generalizar certas sequências de etiquetas, agrupando sequências diferentes que executam a mesma função como um único grupo sintático.

Para testarmos essa idéia é necessário recuperar as estruturas sintáticas das sentenças. Isso pode ser feito de várias formas, com requisitos e resultados diferentes. Na próxima seção introduzimos a área de estudo de PLN chamada *Análise Sintática*, para darmos a base para a explicação sobre duas diferentes formas de obtermos estrutura sintática para o etiquetador. Na Seção 4.3 então mostramos como implementamos essas duas abordagens em conjunto com o etiquetador, e apresentamos os resultados obtidos. Problemas encontrados e conclusões são por fim mostrados na Seção 4.4.

4.1 ANÁLISE SINTÁTICA

Sintaxe é o estudo das regularidades e restrições da ordem das palavras e da estrutura das sentenças (Manning e Schütze, 1999). Um dos requisitos fundamentais para o bom funcionamento de sistemas de processamento de linguagem, especialmente quando é necessária a interpretação – como em sistemas de Extração de Informação, Tradução Automática, ou Reconhecimento de Fala –, é a eficiência na recuperação da estrutura sintática das sentenças. Em sua forma mais simples, o problema da análise sintática envolve a definição de um algoritmo que associa a uma sentença de entrada sua correspondente estrutura de árvore sintática. Assumimos uma definição do problema da análise sintática na qual cada sentença deve ser associada a uma única árvore sintática, mesmo que isso necessite de desambiguação usando fontes de conhecimento além da gramática. Isso contrasta com outra definição comum do problema, na qual a tarefa é recuperar todas árvores bem-formadas para a sentença, sem nenhuma desambiguação.

A análise sintática por meio do computador é realizada pelo *analisador sintático*[1]. Ele pode recuperar a estrutura sintática com a ajuda de uma gramática ou através de aprendizagem automática, tanto supervisionada quanto não-supervisionada. Para a supervisionada utiliza-se *bancos de árvores*[2], que são córpus anotados sintaticamente, como por exemplo o *Penn Treebank* (Marcus, Santorini e Marcinkiewicz, 1994) para o inglês e também o *Tycho Brahe* (2010) para o português.

1 *Parser*, em inglês.
2 Do inglês *treebanks*.

4.1.1 Estruturas e Representações

A estrutura geralmente usada para codificar os vários níveis de informação sintática de uma sentença é chamada de *árvore sintática*. Um exemplo de árvore sintática pode ser visto na Figura 7. Nessa figura, as folhas da árvore são as palavras de uma

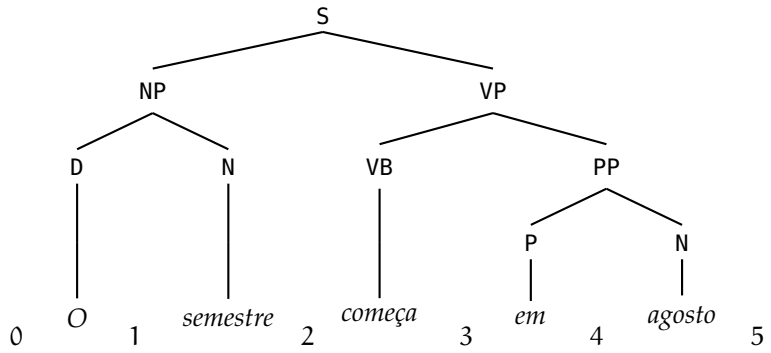


Figura 7: Um exemplo de árvore sintática.

sentença, e os nós imediatamente acima de cada palavra são suas categorias morfossintáticas. Assim, as palavras '*semestre*' e '*agosto*' são *nomes* (`N`), '*O*' é um *determinante* (`D`), '*começa*' é um *verbo* (`VB`) e '*em*' é uma *preposição* (`P`). Também temos a descrição sintática de agrupamentos de palavras, como "*O semestre*", que é um *sintagma nominal* (`NP`, do inglês *Noun Phrase*), "*em agosto*", que é um *sintagma preposicional* (`PP`, *Prepositional Phrase*), e mesmo `S`, a sentença. O *sintagma verbal* é representado pela etiqueta `VP` (do inglês *Verbal Phrase*).

Podemos também nos referir às folhas da árvore como nós *terminais*, e aos nós internos da árvore como nós *não-terminais*. Os nós não-terminais imediatamente acima do nós terminais também podem ser chamados de *pré-terminais*, como é o caso das etiquetas morfossintáticas no exemplo anterior.

Uma árvore sintática também pode ser representada através de uma *parentetização*, que basicamente significa envolver entre parênteses as palavras dominadas por cada nó da árvore. Uma parentetização ou árvore pode ser rotulada ou não com as etiquetas sintáticas e morfossintáticas. Usando a sentença do exemplo anterior, a parentetização não rotulada fica como na Figura 8 e a parentetização rotulada como na Figura 9.

((*O semestre*) (*começa* (*em agosto*)))

Figura 8: Um exemplo de parentetização não rotulada.

(S (NP (D *O*) (D *semestre*)) (VP (VB *começa*) (PP (P *em*) (N *agosto*))))

Figura 9: Um exemplo de parentetização rotulada.

4.1.2 Constituintes

Algumas palavras e grupos de palavras funcionam como uma única unidade na estrutura de uma sentença, se comportando como *constituintes*. Constituintes podem

ser detectados, por exemplo, se puderem ser substituídos por um pronome, expandidos, ou se puderem ocorrer em várias posições dentro de uma cláusula (Manning e Schütze, 1999, pág. 93). Por exemplo, um constituinte que agrupa nomes e seus modificadores, como "*o semestre*". Todos elementos que podem ser substituídos um pelo outro em uma certa posição sintática são membros de um *paradigma*. Duas palavras possuem uma *relação sintagmática* se podem formar um *sintagma*, como "*o semestre*", que é um sintagma nominal.

4.1.3 Abordagem Estatística

Uma abordagem padrão ao problema da análise sintática (Allen, 1995) é construir uma gramática manualmente, com algum tipo de formalismo, ou com uma grande quantidade de informação específica lexicalizada. A ambigüidade é resolvida através de *restrições seletivas*: por exemplo, um léxico pode especificar que *comer* deve tomar um objeto com a característica *+comida*, e especificar quais nomes no léxico possuem a característica *+comida*. Apesar de viáveis em um domínio limitado, o uso de restrições seletivas enfrenta vários problemas quando aplicado a tarefas mais gerais. O tamanho do vocabulário se torna grande demais e o tamanho de uma gramática de ampla cobertura também é restritivo. Enquanto esses problemas podem não aparecer em domínios restritos, aparecem frequentemente em domínios mais amplos.

Essas dificuldades foram apresentadas por praticamente todas tarefas de PLN que tentaram utilizar abordagens baseadas em regras. Em resposta a elas, as abordagens mais recentes passaram a se basear em aprendizado de máquina, principalmente através de métodos estatísticos.

Fundamentos da Análise Sintática Estatística

Similarmente ao descrito na Seção 2.1.1, sobre os fundamentos da análise morfossintática, o processo da análise sintática apresenta as mesmas caracterísitcas. É divido em dois componentes: *modelo* e *analisador*. A diferença principal é que desta vez, ao invés de uma busca pela melhor sequência de etiquetas, a busca é pela melhor árvore sintática para cada sentença. O conjunto A é agora o conjunto de árvores sintáticas, e $a_{\text{melhor}}(s)$ fornece a árvore mais provável para uma sentença $s \in S$, onde S é ainda o conjunto de sentenças. A definição de a_{melhor} continua a mesma que a da Equação 2.1, que incluindo o vetor de parâmetros Θ fica:

$$a_{\text{melhor}}(s) = \arg\max_{a \in A} \textit{Pontuação}(a, s|\Theta),$$

onde $\textit{Pontuação} : a \times s \rightarrow [0,1]$.

Como na análise morfossintática, a tarefa de modelagem da análise sintática é dividida em três etapas:

A. Definição do modelo de estrutura: a função $\textit{Pontuação}(a, s|\Theta)$;

B. Definição do modelo parametrizável;

C. Definição do algoritmo de busca para encontrar

$$a_{\text{melhor}}(s) = \arg\max_{a \in A} \textit{Pontuação}(a, s|\Theta).$$

Estruturas de modelo que definem distribuições de probabilidade conjunta $P(a, s|\Theta)$ podem ser treinadas de maneira não-supervisionada usando o algoritmo Maximização de Expectativa[3] (EM) (Dempster, Laird e Rubin, 1977).

A parametrização, agora, é a escolha de como quebrar uma árvore, de maneira que o poder discriminativo e o de compactação sejam equilibrados. Ela é feita de formas diferentes para cada abordagem diferente. Um exemplo de um algoritmo de busca da melhor árvore para uma dada sentença é o *Algoritmo de Earley* (Earley, 1970).

4.1.4 Dificuldades para o Português

Bons resultados são alcançados atualmente para o inglês por analisadores supervisionados. O desenvolvimento de boas ferramentas sintáticas para o português sofre por causa da pouca disponibilidade de córpus sintáticos pré-anotados, apesar de existirem alguns bastante consolidados na comunidade de língua portuguesa, como o MAC-MORPHO (NILC, 2010) e o Floresta Sintá(c)tica (Bick, Santos, Afonso e Marchi, 2007). Há alguns conjuntos de textos anotados e ferramentas para processamento do português disponíveis em locais como a Linguateca (2010) e o NILC (2010), mas ainda assim há ferramentas para o inglês que já estão bem estabelecidas e que não possuem muitos similares com o mesmo desempenho para o português.

Esse estado atual é um grande incentivo ao desenvolvimento de analisadores não-supervisionados. Embora a complexidade da linguagem talvez dificulte mais a tarefa para o português, no momento em que algum sistema não-supervisionado alcançar um certo grau de eficiência a criação de córpus anotados poderá ser alavancada. Atualmente não conhecemos nenhuma sistema assim, o que nos motivou ainda mais a implementar um analisador não-supervisionado.

Modelos de análise sintática que utilizam aprendizado não-supervisionado buscam induzir um conjunto de regras sintáticas, uma *gramática*, a partir de sentenças de etiquetas morfossintáticas, ou às vezes de palavras. Os melhores resultados atualmente para a análise sintática não-supervisionada do inglês são obtidos por Klein e Manning (2004), e nos baseamos neles para implementar um sistema semelhante para o português. Eles apresentam em seus trabalhos dois modelos para a análise sintática não-supervisionada (Klein, 2005; Klein e Manning, 2001, 2002, 2004), e também um modelo híbrido obtido da combinação destes dois. Vamos nos focar no primeiro modelo, que busca identificar os constituintes de uma sentença, utilizando para isto algumas técnicas de *aglomeração*[4] e a idéia do fenômeno linguístico de que constituintes aparecem em contextos semelhantes. Exploramos esse modelo na próxima seção.

4.2 INDUÇÃO GRAMATICAL DO PORTUGUÊS

Métodos supervisionados de indução gramatical dependem de grandes córpus anotados manualmente, o que consome tempo e precisa de especialistas em linguística. Muitas línguas não possuem os recursos necessários e por isso possuem deficiência em dados anotados à mão que estejam largamente disponíveis. Isso inclui o português, apesar do crescente aumento de recursos e pessoal trabalhando em linguística computacional. Portanto, bons métodos para indução gramatical não-supervisionada são desejados.

3 EM, do inglês *Expectation Maximization*.
4 *Clustering*, em inglês.

Alguns modelos baseados em *constituência* estão alcançando resultados promissores para o inglês. Exploraremos um deles, reimplementando-o e mostrando os resultados atingidos para o português usando poucos recursos. Com uma medida-f de 62,3% em aproximadamente 2400 sentenças do português histórico, esse é o melhor resultado que conhecemos para análise sintática não-supervisionada do português. As estruturas aprendidas também mostram boa qualidade, com constituintes e destituintes comuns sendo corretamente detectados. Nossa expectativa é que isso incentive novos trabalhos em indução gramatical não-supervisionada, e assim que mais córpus anotados sejam desenvolvidos.

4.2.1 Introdução

Análise sintática automática envolve fazer um programa analisador tentar aprender a sintaxe da língua e então atribuir a correta estrutura a novas sentenças. As abordagens com mais sucesso são baseadas em métodos de aprendizado supervisionado, geralmente usando algum tipo de modelo probabilístico. Usando córpus anotado para aprender estrutura, atualmente alcançam mais de 90% de precisão (Charniak e Johnson, 2005; McClosky, Charniak e Johnson, 2006; Petrov e Klein, 2007) em texto jornalístico em inglês (Marcus, Santorini e Marcinkiewicz, 1994). Entretanto, uma desvantagem importante é que essas abordagens requerem um grande córpus manualmente anotado com estrutura sintática. Outra desvantagem é que elas são muitos sensíves ao domínio, aplicação e gênero do córpus utilizado.

Um método de aprendizado que recentemente começou a apresentar resultados promissores é o *não-supervisionado*. Ele tenta induzir estrutura a partir de dados crus ou etiquetados, sem a necessidade de texto com marcação sintática. Há um conjunto muito maior de texto cru ou etiquetado do que de texto sintaticamente marcado. Embora os resultados com esse método ainda estejam relativamente bem abaixo dos alcançados por modelos supervisionados – para línguas onde há recursos que possibilitam essa comparação –, o uso de modelos não-supervisionados ainda é atrativo dada uma língua ou domínio sem recursos.

Uma das classes de algoritmos de aprendizado não-supervisionado mais promissoras usa evidências distribucionais para identificar estruturas de constituintes. O modelo baseado em constituintes de Klein e Manning (2002) fornece bons resultados quando aplicado ao inglês. Iremos descrever como implementamos esse modelo e o aplicamos ao português histórico. Utilizando os escassos recursos disponíveis, foi possível conseguir resultados encorajadores com pouco mais de duas mil sentenças. Em termos absolutos o desempenho ainda é baixo, mas já está bastante acima da *linha base* dessa tarefa.

Depois de mostrar outros trabalhos em indução não-supervisionada e análise sintática do português na próxima seção, explicamos o Modelo de Constituintes e Contextos[5] (CCM) de Klein e Manning, depois os recursos usados e então os resultados atingidos. Finalmente, tecemos algumas conclusões e idéias de trabalhos futuros para análise sintática não-supervisionada.

4.2.2 Trabalhos Anteriores

Os principais trabalhos recentes em indução gramatical não-supervisionada usando modelos baseados em constituintes são os de Clark (2001a,b), Klein (2005) e Klein e

5 CCM, do inglês *Constituent-Context Model*.

Manning (2002, 2004). Clark usa *aglomeração*[6] distribucional para agrupar sentenças e então um critério de informação mútua (Manning, Raghavan e Schütze, 2008) para filtrar constituintes e destituintes. Klein e Manning usam uma idéia similar, e seu modelo CCM é explicado adiante. Smith e Eisner (2004) apresentam uma versão do CCM um pouco melhorada através de algumas adaptações, especialmente em relação ao algoritmo EM e seu problema de máximo local.

Todos esses trabalhos foram feitos e testados para o inglês. Para português, não há tais trabalhos similares que pudessem ser encontrados. O problema da análise sintática no português é abordado apenas por modelos supervisionados, tanto probabilísticos quanto baseados em regras (Bick, 2000; Bonfante, 2003; Carvalho e Sousa, 2003; Martins, Nunes e Hasegawa, 2003).

4.2.3 Modelo Baseado em Constituintes

Constituintes, como explicado na Seção 4.1.2, são palavras ou grupos de palavras que funcionam como uma única unidade na estrutura de uma sentença, podendo ser substituídos, expandidos ou movidos em várias posições dentro de uma cláusula.

Klein e Manning (2002) apresentam um *modelo gerativo* (veja Seção 2.1.2)[7] de análise sintática, chamado CCM, no qual a hipótese fundamental é que constituintes aparecem em contextos de constituintes. Constituintes longos geralmente possuem equivalentes mais curtos e mais frequentes que aparecem em contextos similares, e portanto são mais fáceis de terem sua constituência testada. Além de constituintes, o modelo também descreve destituintes e contextos, e com isso tenta transferir a constituência de uma sequência para o seu contexto, dessa forma fazendo com que novas sequências que ocorrem no mesmo contexto sejam classificadas com constituência semelhante mais tarde.

Um *intervalo* é uma subsequência contígua de uma sentença que envolve uma *produção* y – uma sequência de terminais tais como `D ADJ N` – e que ocorre em um *contexto* x – o par ordenado dos terminais à esquerda e à direita, como `VB`—`P`. Se o intervalo iniciar no i-ésimo terminal e for até o j-ésimo, a produção envolvendo esse intervalo é denotada por y_{ij}, e o contexto por x_{ij}.

Uma *parentetização* b de uma sentença é uma matriz booleana que indica quais intervalos são constituintes e quais não são. b é *árvore-equivalente* se não possuir dois intervalos constituintes se cruzando, se os intervalos terminais de tamanho um e o intervalo da sentença inteira forem constituintes, e se os intervalos de tamanho zero forem destituintes. Ela é binária se corresponder a uma árvore binária. A árvore sintática para uma sentença é obtida a partir da indução de uma parentetização árvore-equivalente da sentença.

Dada uma sentença s, o modelo escolhe uma parentetização b de acordo com uma distribuição $P(b)$, e então gera s dado b:

$$P(s, b) = P(b)P(s|b).$$

6 AGLOMERAÇÃO: A técnica de *aglomeração* (do inglês *clustering*) consiste em classificar as palavras de uma sentença em categorias de semelhança. Desse modo, palavras de algum tipo escolhido são referenciadas como um único grupo. Isso é usado principalmente por modelos não-lexicalizados de análise sintática, ou que utilizam aprendizado não-supervisionado. Diferentes modos de decidir quais palavras são de alguma forma semelhantes, e aglomerá-las, são melhor descritos por Schütze (1995).

7 Uma consequência aqui é que a classe do modelo indica a direção em que um analisador sintático gera a estrutura em árvore de uma sentença: no caso de modelos gerativos, de cima para baixo, i.e., da raiz até os itens léxicos; no caso de modelos discriminativos, de baixo para cima.

Cada intervalo é preenchido independentemente, e cada produção e contexto de cada intervalo são independentes um do outro e condicionados à constituência b_{ij} do intervalo:

$$\begin{aligned} P(s|b) &= \prod_{\langle i,j \rangle} P(y_{ij}, x_{ij}|b_{ij}) \\ &= \prod_{\langle i,j \rangle} P(y_{ij}|b_{ij}) P(x_{ij}|b_{ij}). \end{aligned}$$

A distribuição $P(y_{ij}|b_{ij})$ é um par de distribuições multinomiais sobre o conjunto de todas produções possíveis: uma para constituintes ($b_{ij} = c$) e outra para destituintes ($b_{ij} = d$). Igualmente para $P(x_{ij}|b_{ij})$ e contextos. A probabilidade marginal atribuída a uma sentença s é dada pela soma de todas as parentetizações possíveis de s: $P(s) = \sum_b P(b)P(s|b)$.

Para induzir estrutura o algoritmo EM é executado sobre o modelo, com as sentenças S sendo tratadas como observadas e as parentetizações B como não-observadas. Os parâmetros Θ do modelo são as distribuições multinomiais sobre produções e contextos, $P(y|b)$ e $P(x|b)$, $b \in B$.

Se $P(b)$ for uniforme sobre todas possíveis parentetizações, mesmo cruzadas, então esse procedimento equivale ao processo de aglomeração leve com duas classes[8]. Pode-se tornar esse processo em indução de árvores restringindo $P(b)$ a colocar massa somente em parentetizações árvore-equivalentes. Por exemplo, pode-se utilizar uma distribuição $P_{bin}(b)$, que é uniforme sobre parentetizações binárias e zero caso contrário, fazendo assim com que somente árvores binárias válidas sejam consideradas.

PASSO-E: De acordo com o Θ atual, encontre as expectativas das parentetizações B, $P(B|S, \Theta)$. A contagem esperada de existir parênteses ao redor do intervalo y_{ij} de s, i.e., a fração de árvores sobre s que contêm y_{ij} como um constituinte, é dada por

$$P_{\mathcal{B}}(y_{ij}|s) = \frac{\sum_{b \in B : b_{ij} = \mathcal{C}} P(s, b)}{\sum_{b \in B} P(s, b)}.$$

Esse valor é calculado usando um programa dinâmico cúbico bastante similar ao algoritmo *Interno-Externo* (Manning e Schütze, 1999). A fórmual vira

$$P_{\mathcal{B}}(y_{ij}|s) = \frac{\alpha(y_{ij}|s)\beta(y_{ij}|s)}{\beta(y_{0n}|s)}.$$

A equação de contagem esperada para destituintes é um pouco mais complicada, e não é especificada no texto de Klein. Mas intuitivamente ela corresponde à soma do número esperado de parênteses que cruzam o intervalo y_{ij}:

$$P_{\mathcal{C}}(y_{ij}|s) = \sum_{p<i, i<q<j} P_{\mathcal{B}}(y_{pq}|s) + \sum_{i<p<j, q>j} P_{\mathcal{B}}(y_{pq}|s),$$

normalizado pelo número de parênteses cruzados.

8 Aglomeração leve, do inglês *soft-clustering*, é o processo de aglomeração onde os elementos podem pertencer a mais de uma classe, diferentemente da aglomeração rígida, *hard-clustering*, onde cada elemento pertence exclusivamente a uma única classe.

PASSO-M: Compute o novo conjunto de parâmetros Θ', reestimando as distribuições de produções e contextos. As reestimativas são calculadas por ML. As distribuições de constituintes são calculados por

$$\bar{P}(y^a|b = \mathcal{C}) = \frac{\sum_{s\in S}\sum_{y_{ij}=y^a} P_{\mathcal{B}}(y_{ij}|s)}{\sum_{s\in S}\sum_{y} P_{\mathcal{B}}(y|s)},$$

$$\bar{P}(x^a|b = \mathcal{C}) = \frac{\sum_{s\in S}\sum_{x_{ij}=x^a} P_{\mathcal{B}}(x_{ij}|s)}{\sum_{s\in S}\sum_{x} P_{\mathcal{B}}(x|s)}.$$

As distribuições de destituintes são calculadas analogamente, usando as expectativas $P_{\mathcal{C}}$.

As parentetizações não podem ser eficientemente enumeradas, assim um programa dinâmico de complexidade cúbica similar ao *algoritmo Interno-Externo* (Baker, 1979) é usado para calcular as contagens esperadas de cada produção e cada contexto, tanto constituintes quanto destituintes.

Para suavização, duas contagens são adicionadas para cada produção e contexto como sendo constituintes, e oito contagens como sendo destituintes, já que intervalos aletórios têm maior probabilidade de serem destituintes.

O processo não começa no passo-E, e sim no passo-M usando uma distribuição inicial sobre as parentetizações. Mas a distribuição inicial usada não é a distribuição uniforme sobre árvores binárias $P_{\text{bin}}(B)$, porque combinatoriamente quase todas as árvores são relativamente balanceadas, enquanto que na linguagem quer-se permitir que estruturas desbalanceadas tenham uma chance razoável de serem descobertas. Klein descreve um processo de separação uniforme de geração de árvores para se obter uma distribuição $P_{\text{split}}(B)$: escolhe-se aleatoriamente um ponto de separação na sentença, então recursivamente constrói-se árvores em cada lado da separação. Essa distribuição coloca relativamente mais peso em árvores desbalanceadas do que a distribuição uniforme sobre árvores binárias.

Entretanto, Klein comenta que essa distribuição pareceu tender demais *contra* estruturas balanceadas, e por isso também não a usa. Acaba utilizando uma distribuição que obteve empiricamente, e de que infelizmente não dá detalhes e nem fala nada mais a respeito em nenhuma de suas publicações. De qualquer maneira, em nosso sistema implementamos a distribuição $P_{\text{split}}(B)$, e todos os resultados descritos adiante foram obtidos utilizando ela.

4.2.4 Recursos e Avaliação

Utilizamos os textos do Tycho Brahe que foram anotados manualmente com estrutura sintática. Para nos referirmos a esse conjunto de textos e o distinguirmos do córpus etiquetado, vamos chamá-lo de córpus sintático TB.

Para comparar os resultados com os para o inglês, e por questões de eficiência do modelo, removemos do córpus a pontuação e tomamos apenas as sentenças com até dez palavras. Esse é o processo adotado pelos autores do modelo. Chamaremos este novo conjunto de sentenças de córpus TB10, o qual contém 2414 sentenças e 16790 palavras. As 214 sentenças retiradas do final de um dos textos analisados sintaticamente[9] formam o córpus TB10-teste, o conjunto de teste.

A Tabela 10 mostra as etiquetas mais comuns no TB10. A Tabela 11 mostra as etiquetas de flexão de genêro e número. Elas são "(...) contempladas em nosso

9 Marquês da Fronteira e Alorna (1802): *Memórias do Marquês da Fronteira e d'Alorna*; http://www.tycho.iel.unicamp.br/~tycho/corpus/texts/xml/a_003 (é preciso realizar cadastro, gratuito).

ETIQUETA	SIGNIFICADO
D	Determinante
N	Nome
NPR	Nome próprio
VB	Verbo no infinitivo
P	Preposição
CONJ	Conjunção
ADV	Advérbio
ADJ	Adjetivo
PRO	Pronome

Tabela 10: Etiquetas frequentes do córpus Tycho Brahe.

TIPO	ETIQ. DE FLEXÃO	SIGNIFICADO
Gênero	*nenhuma*	Masculino
	-F	Feminino
	-G	Duplo
Número	*nenhuma*	Singular
	-P	Plural

Tabela 11: Etiquetas de flexão de gênero e número.

sistema de anotação para capturar a riqueza morfológica que o português exibe em determinantes, pronomes, nomes, adjetivos, quantificadores, passados particípio, e assim por diante."[10] Assim, por exemplo, D-F é o determinante feminino.

Para avaliar a eficiência de um analisador sintático utiliza-se as medidas de *precisão*, *cobertura* e *medida-f*. Considerando-se as árvores geradas por algum algoritmo de busca e as respectivas árvores originais obtidas do conjunto de teste, estas medidas são calculadas da seguinte maneira:

$$\text{precisão} = \frac{\text{número de nós corretos da árvore gerada}}{\text{número total de nós na árvore gerada}}$$

$$\text{cobertura} = \frac{\text{número de nós corretos da árvore gerada}}{\text{número total de nós na árvore original}}$$

$$\text{medida-f} = 2 \times \frac{\text{precisão} \times \text{cobertura}}{\text{precisão} + \text{cobertura}}$$

As medidas de precisão, cobertura e medida-f finais são calculadas através da média dessas medidas obtidas para cada sentença.

Para melhor avaliar o sucesso de um sistema, principalmente no caso de sistemas não-supervisionados, um método aleatório é utilizado para gerar resultados básicos, que são chamados de *linha base*[11]. Essa linha indica a forma mais trivial de se abordar um problema. Não se espera bons resultados disso, mas sua utilidade está exatamente em mostrar o mínimo que outros métodos que abordem o problema devem alcançar. No caso da análise sintática, um método para se obter a *linha base* é gerar aleatoriamente uma árvore para cada sentença. Um analisador sintático precisa

10 "(...) contemplated in our annotation system in order to capture the morphological richness Portuguese exhibits in determiners, pronouns, nouns, adjectives, quantifiers, passive participles, and so on", http://www.tycho.iel.unicamp.br/~tycho/corpus/manual/tags.html#item_agreement.

11 Em inglês, *baseline*.

ao menos ultrapassar essa linha para se dizer que possui um certo grau de sucesso. Além disso, uma solução pode estar bastante longe dos 100% e ainda assim estar bem acima da *linha base*, indicando um alto grau de sucesso do método.

Às vezes também é o caso de se calcular um *limitante superior* [12]. Isto acontece quando, por alguma limitação do modelo do analisador ou do próprio córpus sintático, as árvores geradas nunca irão corresponder exatamente às árvores do córpus. Um exemplo disto é quando o analisador gera apenas árvores binárias; em geral os córpus possuem árvores sem restrição de ordem, n-árias. Embora o cálculo de eficiência deva ser em relação ao córpus original, um *limitante superior* ajuda a ver quão perto do seu máximo um analisador alcança.

4.2.5 Experimentos

Para diferenciar do modelo e do sistema desenvolvido por Klein, chamaremos o analisador sintático que desenvolvemos para o português baseado no modelo CCM de CCM-P. Treinamos o CCM-P com o córpus TB10, e o testamos sobre o córpus de teste (TB10-teste).

Nossa *linha base* é dada pelo método ALEATÓRIO, que escolhe uniformemente uma árvore aleatória de um conjunto de árvores binárias. A medida-f obtida por esse processo foi de 39,74%.

Como CCM-P produz apenas árvores binárias, também estamos limitados quanto ao máximo que se pode alcançar. O *limitante superior*, LIMSUP, compara o córpus sintático de teste com uma cópia binarizada dele, e assim mostra a eficiência máxima que pode ser alcançada utilizando-se árvores binárias. No córpus TB10-teste esse limite foi de 77,86% de medida-f.

A eficiência é medida usando-se métricas PARSEVAL não-rotuladas calculadas pelo programa EVALB[13]. Uma métrica não-rotulada significa que os rótulos dos nós das árvores são ignorados, e o que é avaliado é a estrutura, i.e., a parentetização. Usa-se essas métricas aqui porque o objetivo não é inferir tais rótulos, embora isso possa ser feito com um certo grau de sucesso através de uma gramática sobre as etiquetas morfossintáticas das palavras. Mesmo assim, é preciso primeiro saber quais são os grupos corretos de palavras, para então poder atribuir rótulos aos grupos.

As métricas são PN, CN e FN, e se referem a Precisão, Cobertura e medida-F Não-rotuladas, respectivamente. A taxa de parênteses cruzados (PC) fornece o número médio de quantos parênteses adivinhados por sentença cruzam com um mais parênteses do córpus original de teste. Constituintes que não podem ser errados, como uma só palavra e sentenças inteiras, são descartados.

Utilizamos as etiquetas morfossintáticas como entrada para o modelo. O uso de etiquetas como entrada reduz a esparsidade dos dados e torna mais fácil enxergar um esboço da gramática. Entretanto, também limita o máximo que o modelo pode alcançar, caso a taxa de acerto da etiquetagem não seja 100%, e além disso algumas etiquetas dão dicas de conhecimento posterior da estrutura, isto é, carregam distinções que somente poderiam ser feitas sintaticamente. Como teste, Klein compara seus experimentos com experimentos feitos utilizando como entrada etiquetas induzidas distribucionalmente (Schütze, 1995). O resultado obtido foi um pouco pior, em torno de 12% menor, mas Klein continua utilizando as etiquetas do córpus para os principais experimentos, como também o fizemos.

A Tabela 12 mostra os resultados obtidos. Como estes são os primeiros resultados que conhecemos de análise sintática não-supervisionada para o português, não há

12 Do inglês *upper bound*.
13 Disponível livremente em http://nlp.cs.nyu.edu/evalb/.

outros resultados com os quais comparar diretamente. Mas o CCM-P obtém performance razoavelmente boa, com uma medida-f de 62,35%. Considerado a linha base e o limitante superior, o valor está mais perto do limitante superior do que da linha base.

MODELO	FN (%)	PN (%)	CN (%)	PC
ALEATÓRIO	39,74	32,68	50,66	2,52
CCM-P	62,35	51,28	79,49	0,88
LIMSUP	77,86	64,05	100,00	0,00

Tabela 12: Resultados da análise sintática não-supervisionada para o português.

A Tabela 13 mostra os dez constituintes com maior probabilidade. Com exceção dos números 7 e 10, todos os outros constituintes são Sintagmas Nominais (`NP`), consistindo de determinantes (`D`) e nomes (`N`), e diferindo apenas nos modificadores. Os números 5 e 9 possuem um pronome e um adjetivo, respectivamente. O constituinte 7 é uma cláusula pequena infinitiva, denotada por `IP-INF` no córpus TB, tendo uma conexão clítica com o verbo. O constituinte 10 também é `IP-INF`, com um verbo conectando a um sintagma nominal.

CONSTITUINTES	ORDEM	FREQUÊNCIA
`D-F N`	1	619
`D N`	2	451
`D-P N-P`	3	160
`D-F-P N-P`	4	157
`D-F PRO$-F N`	5	58
`D NPR`	6	73
`CL VB`	7	26
`D-F NPR`	8	37
`D-F ADJ-F N`	9	35
`VB D-F N`	10	17

Tabela 13: Constituintes com maior probabilidade.

A Tabela 14 mostra os dez destituintes com maior probabilidade. Os números 1 e 8 são quase sintagmas preposicionais (`PP`), mas como um nome está faltando do lado direito, eles são corretamente classificados como destituintes. Além disso, é interessante observar que o mesmo acontece com as outras oito subsequências: terminam com um determinante. Elas expandem apenas para a esquerda, e são contidas umas dentro das outras. Por exemplo, o número 1 expande para o número 2, que expande para o 5, depois para o 7, e então para o 10. Ou o número 3, que expande para o 9 e depois para o 6.

Nas Tabelas 15 e 16 temos os constituintes sobre- e sub-propostos. Constituintes sobre-propostos são os constituintes que aparecem mais vezes nas árvores geradas do que no córpus original. Semelhantemente, constituintes sub-propostos são mais frequentes no córpus original do que na estrutura gerada pelo analisador CCM-P. Como é possível notar, muitos `NP`s são sobre-propostos. Isso pode ser devido ao fato de o esquema de anotação do córpus não anexar primeiro determinante e nome a um `NP` e então a um `PP`, mas ao invés disso usar construções mais planas como `(NP (D )(N )(PP (...)))`, ou seja, apenas um `NP` dominando sobre `D N PP`, por exemplo. Isso também pode explicar por quê muitos `PP`s são sub-propostos.

DESTITUINTES	ORDEM
P D-F	1
N P D-F	2
VB-P P D-F	3
VB-D P D-F	4
D-F N P D-F	5
D-F N VB-P P D-F	6
VB-P D-F N P D-F	7
P D	8
N VB-P P D-F	9
ADV VB-P D-F N P D-F	10

Tabela 14: Destituintes mais frequentes.

ORDEM	CONSTITUINTES	SOBRE-PROPOSTAS
1	D-F N	13
2	D N	9
3	ADJ N	8
	D-F-P N-P	8
	PRO$-F N	8
6	D-P N-P	7
7	D-P PRO$-P	5
	D-UM-F N	5
9	ADJ-F N	4
	ADJ-G N	4
	CL VB-D	4
	D NPR	4
	PRO$ N	4

Tabela 15: Constituintes sobre-propostos mais frequentes.

4.2.6 Conclusões

Dado que ainda há demanda para a criação de córpus em português anotado manualmente com estrutura sintática, e os problemas de adaptação a domínio dos analisadores sintáticos supervisionados, o uso com sucesso de analisadores não-supervisionados é altamente desejado. Surpreendentemente, parece que não há muito trabalho sendo feito centrado no português, ao contrário do crescente interesse mostrado por trabalhos em outras línguas.

Nós reimplementamos um dos melhores modelos não-supervisionados de indução gramatical do inglês. Klein e Manning (2002) alcançam 71,9% de eficiência no inglês, no córpus sintático *Wall Street Journal* (WSJ10) (1994). Nossa implementação alcança 62,3% no português, usando o córpus anotado *Tycho Brahe* (TB10) (2010) que é três vezes menor. Este é o melhor resultado conhecido publicado para análise sintática não-supervisionada em qualquer conjunto de dados do português.

Mostramos que o sistema adquire uma quantidade razoavelmente boa de estrutura correta listando os constituintes e destituintes melhor aprendidos, e também os erros mais comuns feitos durante o teste. Os constituintes e destituintes aprendidos são linguisticamente significativos, enquanto os principais erros no teste são

ORDEM	CONSTITUINTES	SUB-PROPOSTAS
1	P NPR	9
2	P D N	5
	P DEM	5
	P PRO	5
5	P D NPR	4
	P D-F N	4
7	D-F N P D-P N-P	3
	P D-P N-P	3
	P N	3
	Q ADV	3

Tabela 16: Constituintes sub-propostos mais frequentes.

devidos principalmente a árvores achatadas no córpus original e a algumas análises alternativas.

Como desvantagens do modelo está sua alta dependência em relação à inicialização do modelo – que aliás não conseguimos duplicar, já que Klein e Manning usam uma distribuição empírica não descrita –, sua capacidade de lidar apenas com árvores binárias, e seu comprimento de sentença limitado a dez palavras. Essas deficiências podem ser foco de trabalho futuro, assim como experimentos com diferentes métodos de busca de parâmetros, como Estimativa Contrastiva (Smith e Eisner, 2005a) e técnicas de *annealing* para EM (Smith e Eisner, 2004, 2005b). Além disso, uma boa experiência seria construir um sistema que usa métodos semi-supervisionados (Pereira e Schabes, 1992), já que na verdade existe uma pequena quantidade de texto em português anotado à mão. A maior expectativa, entretanto, é que este trabalho desperte mais pesquisas em indução gramatical do português, e efetivamente ajude a anotação de novos recursos.

4.3 ETIQUETAGEM COM O AUXÍLIO DE ESTRUTURA SINTÁTICA

Utilizando o analisador sintático CCM-P conforme descrito na seção anterior, salvamos a lista de constituintes aprendidos, junto com sua probabilidade. No VLMM TAGGER, implementamos uma nova estrutura para armazenar esses constituintes.

Na fase de treinamento, um novo passo é executado: depois de adicionados à árvore de contexto esquerdo, os contextos são passados por um método de "compressão", e depois adicionados a uma árvore de contexto de constituintes. Esse método de compressão procura subsequências de etiquetas que formam constituintes. O método é guloso, tentando comprimir um contexto o máximo possível. No lugar das etiquetas que formam um constituinte, uma única etiqueta é colocada. Como os constituintes são não-rotulados, o rótulo que utilizamos foi '$*$'. Assim, por exemplo, se a sequência `D-F N` aparecer no contexto esquerdo, ela será substituída pela etiqueta $*$.

Na fase de etiquetagem/inferência, o mesmo método de compressão de contexto é usado, e então a probabilidade da etiqueta sendo predita é recuperada via árvore de contexto de constituintes. Seja c^* o método de compressão. Então $c^*(t_{i-k,i-1})$ retorna a versão comprimida de um contexto $t_{i-k,i-1}$, e $P(t_i|c^*(t_{i-k,i-1}))$ dá a proba-

bilidade de t_i dado o contexto esquerdo comprimido. Essa probabilidade é usada em conjunto com as outras probabilidades do modelo VLMM, mostradas na Equação 2.8. Chamaremos o etiquetador que implementa esse novo modelo de VLMM+SPANS.

A Tabela 17 mostra a diferença de desempenho entre esse novo modelo e o modelo básico. Houve uma grande piora no desempenho. A Tabela 18 mostra os resultados obtidos quando comprimimos apenas os contextos de *que*, e a Tabela 19 quando apenas o contexto comprimido de *a* é considerado. Em ambos os casos a taxa de acerto piorou. O problema com esse modelo parece ser a generalização excessiva dos contextos à esquerda, mesmo quando considerados apenas contextos de *que* ou de *a*. Esse excesso de generalização é agravado pelo fato de não distinguirmos entre diferentes constituintes, já que o analisador CCM-P não os rotula.

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	*84,5250*	*15,4750*	*1712*	*11063*
	80,0235	19,9765	2210	11063
a	*94,5576*	*5,4424*	*740*	*13597*
	91,4687	8,5313	1160	13597
Desconhecidas	*74,9214*	*25,0786*	*1995*	*7955*
	69,6543	30,3457	2414	7955
Conhecidas	*97,2642*	*2,7358*	*7348*	*268586*
	96,0076	3,9924	10723	268586
Total	*96,6215*	*3,3785*	*9343*	*276541*
	95,2495	4,7505	13137	276541

Tabela 17: Resultados do modelo VLMM+SPANS comparados aos do VLMM (em itálico) no córpus segmentado.

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	*84,5250*	*15,4750*	*1712*	*11063*
	83,6753	16,3247	1806	11063
a	*94,5576*	*5,4424*	*740*	*13597*
	94,5356	5,4644	743	13597
Desconhecidas	*74,9214*	*25,0786*	*1995*	*7955*
	74,9089	25,0911	1996	7955
Conhecidas	*97,2642*	*2,7358*	*7348*	*268586*
	97,2281	2,7719	7445	268586
Total	*96,6215*	*3,3785*	*9343*	*276541*
	96,5860	3,4140	9441	276541

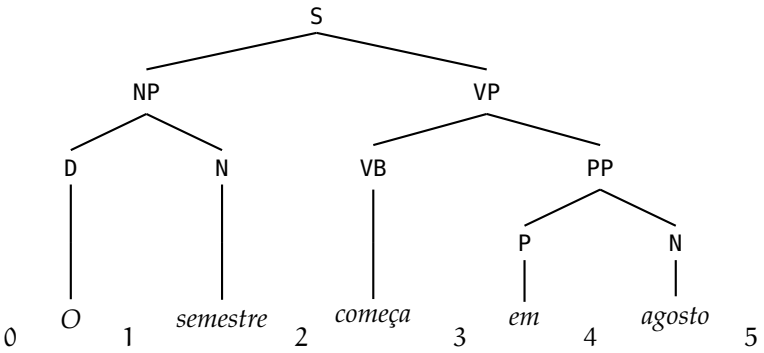
Tabela 18: Resultados do modelo VLMM+SPANS-QUE comparados aos do VLMM (em itálico) no córpus segmentado.

Para tentar resolver esse problema, e seguindo na mesma idéia de utilizar estrutura sintática para melhorar a taxa de acerto ao menos de *que*, experimentamos implementar um outro parâmetro para o modelo VLMM. Ao invés de utilizar os constituintes induzidos pelo analisador CCM-P, utilizamos diretamente as árvores sintáticas dis-

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	*84,5250* 84,5069	*15,4750* 15,4931	*1712* 1714	*11063* 11063
a	*94,5576* 93,0794	*5,4424* 6,9206	*740* 941	*13597* 13597
Desconhecidas	*74,9214* 74,9340	*25,0786* 25,0660	*1995* 1994	*7955* 7955
Conhecidas	*97,2642* 97,1812	*2,7358* 2,8188	*7348* 7571	*268586* 268586
Total	*96,6215* 96,5412	*3,3785* 3,4588	*9343* 9565	*276541* 276541

Tabela 19: Resultados do modelo VLMM+SPANS-A comparados aos do VLMM (em itálico) no córpus segmentado.

poníveis no Tycho Brahe. Para isso, na fase de treinamento, recuperamos todos não-terminais de penúltimo nível, ou seja, os não-terminais que dominam ao menos um pré-terminal, que são as etiquetas morfossintáticas. Para exemplificar, considere o exemplo de árvore sintática mostrado na Figura 7, repetida abaixo:



Nessa árvore, os não-terminais de penúltimo nível são `NP`, `VP` e `PP`, mas não `S`.

A partir daí o processo é semelhante ao do modelo com constituintes, só que desta vez temos um rótulo para atribuir a uma sequência de etiquetas que forma um constituinte. Vamos chamar esse modelo de VLMM+SYNTAX. Executando-o apenas sobre o córpus segmentado, já que as árvores sintáticas estão todas segmentadas, obtemos os resultados mostrados na Tabela 20. Novamente, o resultado piora em todos quesitos reportados. Mesmo depois de várias tentativas de ajuste dos argumentos da árvore de contexto de constituintes, este foi o melhor resultado alcançado.

Alterando o modelo para considerar apenas contextos de *que*, tanto no treinamento quanto na etiquetagem, obtemos os resultados da Tabela 21, na qual chamamos esse novo modelo de VLMM+SYNTAX-QUE. Desta vez, comparando com o modelo original, a taxa de acerto sobre *que* melhora. Há uma redução do erro de 2,39%. Comparando as matrizes de confusão das etiquetas de *que*, mostradas na Tabela 22, podemos ver que a melhora na taxa de acerto é devida a uma maior predisposição em atribuir a etiqueta `C` ao invés da etiqueta `WPRO`: os acertos de `WPRO` diminuem 0,52%, enquanto os de `C` aumentam 1,78%. O número de vezes em que `WPRO` é atribuída erroneamente no lugar de outras etiquetas também diminui, enquanto o número de atribuições

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	*84,5250* 75,1695	*15,4750* 24,8305	*1712* 2747	*11063* 11063
a	*94,5576* 91,5496	*5,4424* 8,4504	*740* 1149	*13597* 13597
Desconhecidas	*74,9214* 71,0999	*25,0786* 28,9001	*1995* 2299	*7955* 7955
Conhecidas	*97,2642* 95,3728	*2,7358* 4,6272	*7348* 12428	*268586* 268586
Total	*96,6215* 94,6746	*3,3785* 5,3254	*9343* 14727	*276541* 276541

Tabela 20: Resultados do modelo VLMM+SYNTAX comparados aos do VLMM (em itálico) no córpus segmentado.

erradas de `C` a outras etiquetas aumenta. A taxa de acerto de `WD` também aumenta, e a das etiquetas minoritárias não se altera.

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	*84,5250* 84,8956	*15,4750* 15,1044	*1712* 1671	*11063* 11063
a	*94,5576* 94,5429	*5,4424* 5,4571	*740* 742	*13597* 13597
Desconhecidas	*74,9214* 74,8712	*25,0786* 25,1288	*1995* 1999	*7955* 7955
Conhecidas	*97,2642* 97,2798	*2,7358* 2,7202	*7348* 7306	*268586* 268586
Total	*96,6215* 96,6352	*3,3785* 3,3648	*9343* 9305	*276541* 276541

Tabela 21: Resultados para o modelo VLMM+SYNTAX-QUE comparados aos do VLMM (em itálico) no córpus segmentado.

Também testamos o uso do modelo VLMM+SYNTAX apenas para a palavra *a*, gerando o modelo VLMM+SYNTAX-A. Os resultados estão na Tabela 23. Ao contrário do que acontece com *que*, o uso de constituintes rotulados esquerdos piora a taxa de acerto de *a*.

4.4 PROBLEMAS E CONCLUSÕES

A primeira idéia para tratar o problema da ambiguidade de *que* foi utilizar o etiquetador VLMM em conjunto com um analisador sintático que inferisse estrutura, ou seja, que usasse métodos não-supervisionados de aprendizado, devido à escassez de recursos anotados para o português. Implementamos para isso o analisador CCM-P, que utiliza o modelo CCM para inferir constituintes. Utilizando esses constituintes

MODELO		WPRO	C	WD	CONJ	FW	D	WD-P	% PREC.
VLMM	WPRO	<5562>	575	14	8	.	.	.	90,3069
	C	850	<3712>	6	26	.	.	.	80,8010
	WD	13	20	<68>	4	.	.	.	64,7619
	CONJ	101	92	.	<5>	.	.	.	2,5253
	FW	.	.	.	.	<4>	.	.	100,0000
	D	.	2	.	.	.	<.>	.	0,0000
	WD-P	.	.	1	.	.	.	<.>	0,0000
VLMM+SYNTAX-QUE	WPRO	<5533>	604	14	8	.	.	.	89,8360
	C	783	<3778>	6	27	.	.	.	82,2377
	WD	7	22	<72>	4	.	.	.	68,5714
	CONJ	93	100	.	<5>	.	.	.	2,5253
	FW	.	.	.	.	<4>	.	.	100,0000
	D	.	2	.	.	.	<.>	.	0,0000
	WD-P	.	.	1	.	.	.	<.>	0,0000

Tabela 22: Matrizes de confusão de *que* no córpus segmentado com os modelos VLMM e VLMM+SYNTAX-QUE (linha: referência; coluna: teste).

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	*84,5250*	*15,4750*	*1712*	*11063*
	84,5069	15,4931	1714	11063
a	*94,5576*	*5,4424*	*740*	*13597*
	94,2708	5,7292	779	13597
Desconhecidas	*74,9214*	*25,0786*	*1995*	*7955*
	74,9466	25,0534	1993	7955
Conhecidas	*97,2642*	*2,7358*	*7348*	*268586*
	97,2489	2,7511	7389	268586
Total	*96,6215*	*3,3785*	*9343*	*276541*
	96,6074	3,3926	9382	276541

Tabela 23: Resultados para o modelo VLMM+SYNTAX-A comparados aos do VLMM (em itálico) no córpus segmentado.

como parâmetros no modelo VLMM para tentar generalizar contextos esquerdos, os resultados obtidos não melhoraram a taxa de acerto geral, nem de *que* e nem de *a*. Diferentes constituintes são representados da mesma maneira porque não são rotulados, devido à natureza não-supervisionada do modelo CCM. O problema, portanto, parece ser uma generalização excessiva dos contextos no modelo VLMM. Ao mesmo tempo em que acrescenta massa às probabilidades da árvore de constituintes, gerando uma árvore baixa e populada, a generalização do contexto também acaba caindo em nova ambiguidade, na qual diferentes etiquetas para uma palavra terminam por ter probabilidades semelhantes dado um contexto esquerdo comprimido.

A idéia seguinte, tentando contornar o problema da generalização, foi utilizar árvores sintáticas diretamente disponíveis, mesmo não sendo em grande número. Esta abordagem se mostrou mais eficaz que a anterior quando aplicada sobre a palavra *que*, o que reforça a idéia de que ela depende de relações não-locais e que ocorrem

em contextos variados. O mesmo não se aplica a *a*, que teve a taxa de acerto piorada quando utilizados contextos generalizados. Por outro lado, isso reforça a idéia de que o tipo de ambiguidade de *a* é diferente da de *que*, e incentiva a investigação de outros contextos para auxiliar sua etiquetagem.

5 ANÁLISE SINTÁTICA SUPERFICIAL

Uma técnica largamente usada em PLN, principalmente para recuperação e extração de informação, é a *Análise Sintática Superficial*[1], ou *Rasa* (Manning e Schütze, 1999). Como o próprio nome sugere, ela é uma análise sintática, mas que não busca produzir uma árvore sintática completa. Ao invés disso, tenta identificar os elementos constituintes de uma frase, como sintagmas nominais e grupos de verbos, sem especificar suas estruturas internas e nem suas funções na sentença. Assim, em aplicações que não precisam de uma análise completa, a análise superficial oferece a vantagem de ser mais simples e rápida.

Como o uso de constituintes não rotulados não ajudou na etiquetagem, e também não houve muito progresso no uso de sintagmas de árvores sintáticas, a idéia de uma nova abordagem é utilizar novamente o córpus sintático do Tycho Brahe e convertê-lo em um córpus sintático raso. A partir desse córpus, podemos realizar a análise sintática superficial sobre os córpus de treinamento e de teste, e então utilizar a marcação extra em conjunto com as etiquetas morfossintáticas. Isto poderá levar a uma melhor distinção do contexto esquerdo das palavras, diferenciando etiquetas morfossintáticas de acordo com sua posição na estrutura da sentença.

A seguir, primeiro explicamos como criamos o córpus sintático raso a partir do córpus sintático, e o córpus morfossintático expandido a partir do sintático raso. Então mostramos como os utilizamos para tentar melhorar a análise morfossintática. Terminamos o capítulo discutindo os resultados, problemas encontrados e conclusões.

5.1 RECURSOS

Usamos os textos disponíveis no córpus Tycho Brahe que estavam anotados manualmente com estrutura sintática. O conjunto desses textos é bastante menor do que os textos anotados com etiquetas morfossintáticas, somando 307.087 palavras e 13.246 sentenças. A análise sintática superficial divide uma sentença em sintagmas não sobrepostos. Como as árvores sintáticas do Tycho Brahe são completas, primeiro as achatamos e então removemos os nós que não são sintagmas relevantes ou que cruzam outros constituintes. Os sintagmas considerados relevantes serão mostramos adiante. Convertemos a notação parentizada, explicada na Seção 4.1.1, para a notação IOB, descrita a seguir. Esse processo dá origem a um *córpus IOB*, que é um córpus sintático superficial.

A notação IOB usa as etiquetas `B`, `I` e `O` para indicar se uma palavra ou etiqueta morfossintática começa (*Begin*), está dentro (*Inside*) ou está fora (*Outside*) de um constituinte, respectivamente. Por exemplo, para uma sentença com a parentetização superficial a seguir

`(NP (D` *O*`)(D` *semestre*`))(VB` *começa*`)(PP (P` *em*`)(N` *agosto*`))`

a notação IOB em palavras fica

O`/B` *semestre*`/I` *começa*`/O` *em*`/B` *agosto*`/I .`

1 Em inglês, *Shallow Parsing* ou *Chunking*.

Opcionalmente, as etiquetas `B` e `I` podem especificar o tipo de sintagma – ainda que `I` possa ser induzida a partir da primeira etiqueta `B` anterior. O exemplo acima, com etiquetas de tipo, fica:

O/`B-NP` *semestre*/`I-NP` *começa*/`O` *em*/`B-PP` *agosto*/`I-PP` `.`

O processo para etiquetagem usando análise sintática superficial consiste em:

A. Gerar um córpus IOB selecionando os sintagmas de interesse a partir do córpus sintático;

B. Treinar o etiquetador VLMM com o córpus IOB e etiquetar os córpus segmentados de treinamento e de teste, gerando novos córpus com etiquetas IOB;

C. Juntar as etiquetas dos córpus resultantes acima com as dos córpus originais, gerando mais um conjunto de córpus compostos de palavras e etiquetas expandidas, onde essas etiquetas são compostas por classes morfossintáticas (`POS`) e de constituintes (`IOB`), no formato *palavra*/`POS^IOB` ;

D. Treinar o etiquetador com o córpus de treinamento `POS^IOB` e etiquetar o córpus de teste;

E. Medir a eficiência tomando o córpus de teste etiquetado acima e retirando a parte IOB das etiquetas; em seguida, comparando com o córpus de teste original, tem-se a taxa de acerto da etiquetagem.

A junção e separação das etiquetas POS e IOB é feita por pequenos programas em Ruby e em Python, que desenvolvemos. A avaliação da eficiência é feita por outro programa nosso que utiliza o NLTK, *Natural Language Tool Kit*[2], e mais um feito em Ruby que calcula a eficiência de diversas palavras.

5.2 ETIQUETAGEM COM SINTAXE SUPERFICIAL

Para iniciar o processo descrito acima, geramos, no passo A, um córpus IOB contendo apenas sintagmas nominais. Fizemos um pequeno experimento com esse córpus IOB: dividimos aleatoriamente as sentenças em conjuntos de 90% e 10%, o primeiro para treinamento e o segundo para teste. Executamos então o etiquetador VLMM sobre esses conjuntos. A taxa de acerto atingida foi 81,16%. Isso significa a taxa de acerto de um modelo VLMM sintonizado para análise morfossintática na detecção de sintagmas nominais, usando pouco treinamento. As palavras com maior quantidade de erros nesse pequeno teste foram:

PALAVRAS	ERROS	OCORRÊNCIAS	% ERRO
,	723	2532	28,55
que	495	1016	48,72
a	329	1652	19,92
de	286	1969	14,53
e	181	877	20,64

2 http://www.nltk.org.

A maioria dos erros se deve, exatamente, à pouca quantidade de dados de treinamento. Com isso o contexto adquire peso menor, e a probabilidade final é definida com base na probabilidade entre palavra e etiquetas.

Considerando o córpus IOB inteiro – e não mais os conjuntos IOB de treinamento e teste –, verificamos que a vírgula, ‘, ’, mais de 77% das vezes está fora de sintagmas (`O`), mas em alguns casos, 20%, está dentro (`I`). A palavra *de*, 70% das vezes, possui a etiqueta `I`, e outros 29% a etiqueta `O`. Já *e*, 75% das vezes está fora de qualquer constituinte, e 23% aparece internamente em um sintagma. Para *que* e *a* tabulamos os resultados. A Tabela 24 mostra as frequências de etiquetas para *que* e *a* no córpus IOB. É interessante notar que *a* praticamente sempre faz parte de um sintagma (73%), geralmente sendo o começo dele (63%). Já *que* poucas vezes é começo de um sintagma, sendo quase metade das vezes (47%) externo a sintagmas e outras quase tantas vezes (44%) interno a um sintagma.

ETIQUETA	OCORRÊNCIAS
`O`	5325
`I-NP`	4936
`B-NP`	980

(a) *que*

ETIQUETA	OCORRÊNCIAS
`B-NP`	11344
`O`	4801
`I-NP`	1815

(b) *a*

Tabela 24: Frequência de etiquetas de *que* e *a* no córpus IOB com sintagmas `NP`.

Seguindo para os passos B e C, obtemos dois córpus que possuem etiquetas morfossintáticas expandidas com etiquetas IOB: o de treino e o de teste. A Tabela 25 mostra a frequência dessas etiquetas expandidas para *que* e *a* no córpus de teste. Podemos observar que *que* e *a* novamente apresentam comportamentos distintos. As duas principais etiquetas de *que*, `C` e `WPRO`, continuam bastante ambíguas, pois foram em sua maioria expandidas com as mesmas etiquetas IOB: nesse caso, `O`. Já *a* apresenta uma diferenciação maior entre suas duas principais etiquetas, `D-F` e `P`: a grande maioria das ocorrências de `D-F` foi expandida com `B-NP`, enquanto `P` foi mais expandida com `O`. A esta altura nos parece que *a* vai ser melhor beneficiada por este método do que *que*.

Continuando com os passos D e E do processo de etiquetagem, usamos os córpus `POS^IOB` para treinar o etiquetador e etiquetar o córpus de teste. A taxa de acerto, com as etiquetas expandidas, é de 89,44%. Mas retirando a parte IOB do córpus etiquetado e comparando com o córpus de teste original, alcançamos 96,57% de taxa de acerto. Isso é ligeiramente pior que o modelo original, como mostrado na Tabela 26. Mas a tabela também mostra resultados interessantes: enquanto a taxa de acerto sobre *que* piora bastante, a taxa de acerto de *a* é bastante melhorada. Há uma diminuição no erro sobre *a* de mais de 20%. A comparação dos erros cometidos em *a* no modelo original e no modelo usando os dados dos sintagmas, que chamamos de VLMM+IOB, é mostrada na Tabela 27. Podemos ver que o uso de sintagmas `NP` aumenta a tendência do etiquetador de classificar *a* como preposição ao invés de determinante. Enquanto a taxa de acerto sobre `D-F` cai um pouco, a de `P` aumenta bastante.

Para fazer mais experimentos, geramos outro córpus IOB, dessa vez selecionando sintagmas nominais e também preposicionais – `NP` e `PP`. A frequência das etiquetas IOB de *que* e *a* é mostrada na Tabela 28. A palavra *que* continua sendo, na maioria dos casos, externo a qualquer sintagma. Já *a*, apesar de continuar sendo primariamente começo de um sintagma, com o uso de sintagmas preposicionais deixou quase de ser

ETIQUETA	OCORRÊNCIAS
C^O	3762
WPRO^O	3723
WPRO^I-NP	2406
C^I-NP	776
CONJ^O	184
C^B-NP	56
WD^O	54
WPRO^B-NP	30
WD^B-NP	26
WD^I-NP	25
CONJ^I-NP	10
CONJ^B-NP	4
FW^I-NP	3
D^O	2
WD-P^B-NP	1
FW^O	1

(a) *que*

ETIQUETA	OCORRÊNCIAS
D-F^B-NP	7861
P^O	3513
P^B-NP	1051
P^I-NP	297
CL^B-NP	256
D-F^I-NP	215
D-F^O	194
CL^O	191
CL^I-NP	7
FW^B-NP	4
D^B-NP	3
FW^O	3
N^O	1
D-P^B-NP	1

(b) *a*

Tabela 25: Frequência de etiquetas de *que* e *a* no córpus POS^IOB com sintagmas NP.

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	*84,5250* 83,0607	*15,4750* 16,9393	*1712* 1874	*11063* 11063
a	*94,5576* 95,7417	*5,4424* 4,2583	*740* 579	*13597* 13597
Desconhecidas	*74,9214* 74,5569	*25,0786* 25,4431	*1995* 2024	*7955* 7955
Conhecidas	*97,2642* 97,2214	*2,7358* 2,7786	*7348* 7463	*268586* 268586
Total	*96,6215* 96,5694	*3,3785* 3,4306	*9343* 9487	*276541* 276541

Tabela 26: Resultados do modelo VLMM usando sintagmas superficiais (VLMM+IOB) comparados aos do VLMM original (em itálico) no córpus segmentado.

externo. Com o uso apenas de sintagmas nominais, como visto acima *a* muitas vezes possuía a etiqueta O. Com o uso também de sintagmas preposicionais, o número de ocorrências com a etiqueta O caiu mais de 30 vezes.

Para ver o efeito de PP sobre a etiquetagem, principalmente sobre *a*, executamos todos os passos do processo da etiquetagem com sintaxe superficial usando o novo córpus IOB. Os resultados são mostramos na Tabela 29. O resultado geral piorou com a adição de sintagmas preposicionais. Mesmo que o erro em *a* seja menor do que no modelo original, ele é maior do que quando usamos apenas sintagmas nominais. Também *que* tem a taxa de acerto diminuída em relação aos dois outros modelos.

MODELO		D-F	P	CL	FW	D	D-P	N	% PREC.
VLMM	D-F	<8056>	206	8	.	.	.	.	97,4123
	P	433	<4411>	17	.	.	.	.	90,7426
	CL	29	39	<386>	.	.	.	.	85,0220
	FW	2	1	.	<4>	.	.	.	57,1428
	D	3	.	.	.	<.>	.	.	0,0000
	D-P	1	.	.	.	.	<.>	.	0,0000
	N	1	.	.	.	.	.	<.>	0,0000
VLMM+IOB	D-F	<8040>	218	12	.	.	.	.	97,2189
	P	270	<4582>	9	.	.	.	.	94,2604
	CL	26	37	<391>	.	.	.	.	86,1233
	FW	2	.	.	<5>	.	.	.	71,4286
	D	3	.	.	.	<.>	.	.	0,0000
	D-P	1	.	.	.	.	<.>	.	0,0000
	N	1	.	.	.	.	.	<.>	0,0000

Tabela 27: Matrizes de confusão de *a* no córpus segmentado com os modelos VLMM original e VLMM usando estrutura sintática superficial (VLMM+IOB) (linha: referência; coluna: teste).

ETIQUETA	OCORRÊNCIAS
O	4504
I-NP	4273
I-PP	1567
B-NP	894
B-PP	3

(a) *que*

ETIQUETA	OCORRÊNCIAS
B-NP	9643
B-PP	4966
I-PP	1993
I-NP	1209
O	149

(b) *a*

Tabela 28: Frequência de etiquetas de *que* e *a* no córpus IOB com sintagmas NP e PP.

5.3 PROBLEMAS E CONCLUSÕES

O uso de informação sintática superficial favoreceu mais a taxa de acerto de *a* do que de *que*. Dado que a palavra *que* praticamente leva apenas a etiqueta O, a sua ambiguidade não foi diminuída. Isso também se deve provavelmente por causa da grande quantidade de ocorrências da etiqueta O, já que os constituintes das árvores anotadas são muito amplos e encadeados, o que causa a utilização apenas de poucos constituintes pequenos.

Um fator interessante é que *a*, quando um determinante, na maioria das vezes recebeu corretamente a etiqueta de começo de sintagma nominal, e quando preposição, a etiqueta de não pertencente a um sintagma. Isso colaborou com uma maior taxa de acerto sobre as ocorrências em que *a* deveria ser uma preposição ao invés de um determinante.

Apesar da grande redução de erro em *a*, o resultado geral é pior do que o modelo básico, principalmente para *que*, que parece se tornar mais ambiguo com uso de marcação de sintagmas. Uma idéia de desenvolvimento futuro é experimentar considerar sintagmas apenas para *a*, marcando-a apenas com começo de sintagma ou fora de sintagma, e verificar o que acontece.

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	*84,5250*	*15,4750*	*1712*	*11063*
	83,0607	**16,9393**	**1874**	**11063**
	82,5093	17,4907	1935	11063
a	*94,5576*	*5,4424*	*740*	*13597*
	95,7417	**4,2583**	**579**	**13597**
	95,5505	4,4495	605	13597
Desconhecidas	*74,9214*	*25,0786*	*1995*	*7955*
	74,5569	**25,4431**	**2024**	**7955**
	73,9661	26,0339	2071	7955
Conhecidas	*97,2642*	*2,7358*	*7348*	*268586*
	97,2214	**2,7786**	**7463**	**268586**
	97,1484	2,8516	7659	268586
Total	*96,6215*	*3,3785*	*9343*	*276541*
	96,5694	**3,4306**	**9487**	**276541**
	96,4815	3,5185	9730	276541

Tabela 29: Resultados do modelo VLMM usando sintagmas nominais e preposicionais comparados aos do VLMM original (em itálico) e aos do VLMM usando apenas sintagmas nominais (em negrito) no córpus segmentado.

6 TREINAMENTO COM PERCEPTRON

Collins (2002) descreve algoritmos para estimação de parâmetros que até então não haviam sido avaliados para tarefas de PLN. A idéia central dos algoritmos tem por base o algoritmo *Perceptron*, descrito inicialmente em Rosenblatt (1958). Duas novas versões do algoritmo Perceptron, o Perceptron médio e o votado, são mais tarde descritas por Freund e Schapire (1999) como fortes alternativas a algoritmos de aprendizado, como *Máquinas de Suporte Vetorial* (SVM[1]) (Vapnik e Kotz, 2006). Collins coloca o algoritmo Perceptron como uma alternativa aos CRFs[2] para estimação de parâmetros. Os CRFs, por sua vez, foram propostos por Lafferty, Mccallum e Pereira (2001) como uma alternativa aos modelos de Máxima-Entropia, que sofrem do problema de "viés de etiqueta",

Nas próximas seções descrevemos a variante do algoritmo Perceptron dada por Collins para o problema da etiquetagem morfossintática, e a estendemos para usar VLMMs. Na Seção 6.3 descrevemos a forma como implementamos o algoritmo Perceptron dentro do etiquetador VLMM e os resultados obtidos com ele. Por fim, na Seção 6.4 apontamos os principais problemas com essa abordagem e tecemos algumas conclusões.

6.1 ESTIMAÇÃO DE PARÂMETROS

Embora o foco do algoritmo Perceptron seja modelos de Máxima-Entropia, Collins (2002) descreve o caso especial do algoritmo no qual ele é aplicado a um etiquetador trigrama baseado em HMMs. Vamos primeiro descrever esse caso e então estendê-lo para uso com VLMMs.

Denotamos as palavras de uma sentença de comprimento n como $w_1, w_2, \ldots, w_n$, ou, para simplificar a notação, como $w_{1,n}$. A etiqueta associada à palavra w_i, $1 \leqslant i \leqslant n$, é denotada por t_i. Um trigrama de etiquetas numa posição i em uma sentença é o trio de etiquetas t_{i-2}, t_{i-1}, t_i, ou simplesmente $t_{i-2,i}$.

Cada trigrama de etiquetas e cada par etiqueta/palavra possui parâmetros associados. Denotamos os parâmetros associados a um trigrama de etiquetas $\langle x, y, z\rangle$ como $\alpha_{x,y,z}$, e os parâmetros associados a um par etiqueta/palavra $\langle t, w\rangle$ como $\alpha_{t,w}$. Geralmente os parâmetros em um modelo baseado em HMM são estimados via probabilidade condicional. Assim, $\alpha_{x,y,z} = P(z|x,y)$ e $\alpha_{t,w} = P(w|t)$.

A probabilidade de uma sequência de etiquetas $t_{1,n}$ ser atribuída a uma sequência de palavras $w_{1,n}$ é dada por

$$P(w_{1,n}, t_{1,n}) = \prod_{i=1}^{n} P(t_i|t_{i-1}, t_{i-2})P(w_i|t_i), \tag{6.1}$$

que substituindo pelos termos dos parâmetros fica

$$P(w_{1,n}, t_{1,n}) = \prod_{i=1}^{n} \alpha_{t_{i-2},t_{i-1},t_i}\alpha_{t_i,w_i}. \tag{6.2}$$

1 *Support Vector Machines*, em inglês.
2 *Conditional Markov Random Fields*, em inglês.

Essa probabilidade conjunta é usada pelo algoritmo de Viterbi (1967) para encontrar a sequência de etiquetas mais provável.

Uma variação bastante usada das equações acima é a aplicação de logaritmo, transformando o produto de probabilidades em soma, o que é mais fácil de calcular e ajuda a evitar *underflow* de ponto flutuante na implementação do etiquetador. A equação então fica

$$\log P(w_{1,n}, t_{1,n}) = \sum_{i=1}^{n} \log P(t_i|t_{i-1}, t_{i-2}) + \sum_{i=1}^{n} \log P(w_i|t_i).$$

Para simplificar a notação, podemos redefinir as equações dos parâmetros α para incluirem o logaritmo. Assim, $\alpha_{x,y,z} = \log P(z|x,y)$ e $\alpha_{t,w} = \log P(w|t)$, e a Equação 6.2 se torna

$$P(w_{1,n}, t_{1,n}) = \sum_{i=1}^{n} \alpha_{t_{i-2},t_{i-1},t_i} + \sum_{i=1}^{n} \alpha_{t_i,w_i}.$$

Ao invés de usar Máxima-Verossimilhança para estimar os parâmetros, o algoritmo Perceptron utiliza o procedimento descrito a seguir. Seja S o número de sentenças etiquetadas no córpus de treinamento e n_s o comprimento da s-ésima sentença. O par de sequências de palavras e etiquetas de uma sentença s é então denotado por $(w^s_{1,n_s}, t^s_{1,n_s})$, $s = 1 \ldots S$. O Algoritmo 2 é o algoritmo de treinamento descrito em Collins (2002).

Entrada: R: número de iterações sobre o córpus de treinamento.

1: inicialize todos parâmetros $\alpha_{x,y,z}$ e $\alpha_{t,w}$ como zero
2: **para** $r = 1 \ldots R$, $s = 1 \ldots S$ **faça**
3: usando os parâmetros correntes, use o algoritmo de Viterbi para encontrar a melhor sequência de etiquetas para w^s_{1,n_s}
4: chamamos essa sequência de z_{1,n_s}
5: **para cada** *trigrama* $\langle x, y, z\rangle$ *visto* c_1 *vezes em* t^s_{1,n_s} *e* c_2 *vezes em* z_{1,n_s} *onde* $c1 \neq c2$ **faça**
6: $\alpha_{x,y,z} = \alpha_{x,y,z} + c_1 - c_2$
7: **para cada** *par* $\langle t, w\rangle$ *visto* c_1 *vezes em* $(w^s_{1,n_s}, t^s_{1,n_s})$ *e* c_2 *vezes em* (w^s_{1,n_s}, z_{1,n_s}) *onde* $c1 \neq c2$ **faça**
8: $\alpha_{t,w} = \alpha_{t,w} + c_1 - c_2$

Algoritmo 2: Algoritmo Perceptron para treinamento de um HMM de ordem 3.

O treinamento penaliza os parâmetros que contribuíram para erros na sequência z_{1,n_s} e aumenta os valores dos parâmetros que não foram utilizados na predição de z_{1,n_s}. Se $z_{1,n_s} = t^s_{1,n_s}$, nenhum valor de parâmetro é alterado.

Para exemplificar o funcionamento do algoritmo de treinamento, considere a s-ésima sentença etiquetada do córpus de treinamento, $(w^s_{1,n_s}, t^s_{1,n_s})$:

```
tornaram/VB-D a/P bradar/VB as/D-F-P sentinelas/N-P ./.
```

E de acordo com os atuais valores dos parâmetros, digamos que a sequência de etiquetas com maior probabilidade, (w^s_{1,n_s}, z_{1,n_s}), seja:

```
tornaram/VB-D a/P bradar/N as/D-F-P sentinelas/N-P ./.
```

Então os parâmetros que serão subtraídos serão

$$\alpha_{\texttt{N,bradar}},\ \alpha_{\texttt{VB-D,P,N}},\ \alpha_{\texttt{P,N,D-F-P}} \text{ e } \alpha_{\texttt{N,D-F-P,N-P}},$$

e os que serão incrementados em 1 serão

$$\alpha_{\texttt{VB,bradar}},\ \alpha_{\texttt{VB-D,P,VB}},\ \alpha_{\texttt{P,VB,D-F-P}} \text{ e } \alpha_{\texttt{VB,D-F-P,N-P}}.$$

Depois de executadas as R iterações de treinamento, o algoritmo de Viterbi pode ser usado como no caso dos HMMs para predição das melhores etiquetas para um conjunto de sentenças.

Para mais detalhes, veja (Collins, 2002, seção 3), que apresenta teoremas (e provas) que descrevem sob quais condições o algoritmo Perceptron converge. As versões Perceptron médio e Perceptron votado também são brevemente abordadas por Collins.

6.2 PERCEPTRON PARA VLMMS

Podemos expandir o Algoritmo 2 de treinamento de um HMM para uso no treinamento de um VLMM. Os parâmetros associados a um par etiqueta/palavra $\langle t, w\rangle$ continuam denotados da mesma maneira, por $\alpha_{t,w}$, que representa $\log P(w|t)$. Seja k a ordem da VLMC utilizada. As sequências de etiquetas não possuem sempre comprimento k. Em vez disso, como visto na Seção 2.3, uma função contexto $c(\cdot)$ é definida e retorna uma subsequência de uma sequência de k etiquetas, que representa o contexto relevante dessa sequência. Assim, dada uma sentença etiquetada $(w^s_{1,n_s}, t^s_{1,n_s})$, a função contexto, aplicada na posição i da sequência de etiquetas, $c(t_{i-k,i-1})$, retorna a sequência de etiquetas relevantes $t_{i-h,i-1}$, $h \leqslant k$, dado o modelo VLMM atual. Denotamos o logaritmo da probabilidade de uma etiqueta t dado seu contexto relevante, $\log P(t|c(\cdot))$, como $\alpha_{c(\cdot),t}$.

A probabilidade conjunta definida na Equação 6.1 acima passa então a ser definida, com base na Equação 2.9, como:

$$P(w_{1,n}, t_{1,n}) = \sum_{i=1}^{n} \alpha_{c(t_{i-k,i-1}),t_i} + \sum_{i=1}^{n} \alpha_{t_i,w_i}.$$

O Algoritmo 3 mostra o treinamento para um VLMM via Perceptron.

Usando o exemplo da seção anterior, temos a s-ésima sentença etiquetada do córpus de treinamento, $(w^s_{1,n_s}, t^s_{1,n_s})$:

```
tornaram/VB-D a/P bradar/VB as/D-F-P sentinelas/N-P
```

E a sequência de etiquetas mais provável, (w^s_{1,n_s}, z_{1,n_s}), predita via Viterbi de acordo com os atuais valores dos parâmetros:

```
tornaram/VB-D a/P bradar/N as/D-F-P sentinelas/N-P
```

No caso de um VLMM, os parâmetros que serão subtraídos serão

$$\alpha_{\texttt{N,bradar}},\ \alpha_{\texttt{P,N}},\ \alpha_{\texttt{N,D-F-P}},$$
$$\alpha_{\texttt{VB-D,P,N}},\ \alpha_{\texttt{P,N,D-F-P}},\ \alpha_{\texttt{N,D-F-P,N-P}},$$
$$\alpha_{\texttt{VB-D,P,N,D-F-P}},\ \alpha_{\texttt{P,N,D-F-P,N-P}},$$
$$\text{e } \alpha_{\texttt{VB-D,P,N,D-F-P,N-P}},$$

Entrada: R: número de iterações sobre o córpus de treinamento.

1: inicialize todos parâmetros $\alpha_{c(\cdot)}$ e $\alpha_{t,w}$ como zero
2: **para** $r = 1 \ldots R$, $s = 1 \ldots S$ **faça**
3: usando os parâmetros correntes, use o algoritmo de Viterbi para encontrar a melhor sequência de etiquetas para w^s_{1,n_s}
4: chamamos essa sequência de z_{1,n_s}
5: **para cada** *sequência* o_k *de até* k *etiquetas vista* c_1 *vezes em* t^s_{1,n_s} *e* c_2 *vezes em* z_{1,n_s} *onde* $c1 \neq c2$ **faça**
6: $\alpha_{c(o_k)} = \alpha_{c(o_k)} + c_1 - c_2$
7: **para cada** *par* $\langle t, w \rangle$ *visto* c_1 *vezes em* $(w^s_{1,n_s}, t^s_{1,n_s})$ *e* c_2 *vezes em* (w^s_{1,n_s}, z_{1,n_s}) *onde* $c1 \neq c2$ **faça**
8: $\alpha_{t,w} = \alpha_{t,w} + c_1 - c_2$

Algoritmo 3: Algoritmo Perceptron para treinamento de um VLMM de ordem k.

e os que serão incrementados em 1 serão

$\alpha_{\texttt{VB,bradar}}$, $\alpha_{\texttt{P,VB}}$, $\alpha_{\texttt{VB,D-F-P}}$, ▷ `bigramas`
$\alpha_{\texttt{VB-D,P,VB}}$, $\alpha_{\texttt{P,VB,D-F-P}}$, $\alpha_{\texttt{VB,D-F-P,N-P}}$, ▷ `trigramas`
$\alpha_{\texttt{VB-D,P,VB,D-F-P}}$, $\alpha_{\texttt{P,VB,D-F-P,N-P}}$, ▷ `quadrigramas`
e $\alpha_{\texttt{VB-D,P,VB,D-F-P,N-P}}$, ▷ `quinquegrama`

6.3 RESULTADOS

Aplicamos o Algoritmo 3 ao etiquetador VLMM, utilizando a árvore de contexto como os parâmetros $\alpha_{c(\cdot)}$. Iremos nos referir a essa versão do etiquetador baseado em VLMM e usando Perceptron para treinamento como VLMM+PERCEPTRON.

Executamos o etiquetador VLMM+PERCEPTRON sobre o córpus Tycho Brahe, utilizando uma VLMC de ordem 10, treinando via Perceptron por 30 iterações. Nenhuma poda é feita na árvore de contexto durante ou após o treinamento, já que o próprio algoritmo Perceptron penaliza galhos (contextos) ruins e promove os que melhoram a predição. Assim, a poda descartaria o efeito do treinamento.

Os resultados obtidos são mostrados na Tabela 30, em comparação com os resultados do modelo VLMM original. O erro sobre *que* diminui quase 3%, e sobre *a* diminiu quase 16%. Mesmo assim, o resultado final é pior do que o do modelo original. A razão para isso parece ser devido em boa parte a um pior desempenho sobre palavras desconhecidas, cuja taxa de erro sobe quase 30%.

A Tabela 31 mostra a matriz de confusão das etiquetas de *que* quando usados os modelos original e com Perceptron sobre o córpus normal. Podemos ver um aumento na tendência do modelo com Perceptron de atribuir menos vezes a etiqueta `WPRO` a *que*, e mais vezes as etiquetas `C`, `WD` e `CONJ`.

As matrizes de confusão das etiquetas de *a* usando os dois modelos sobre o córpus normal são mostradas na Tabela 32. O número de predições de `D-F` e `P` continua semelhante, mas mais vezes atribuído corretamente. Já a taxa de acerto sobre os clíticos (`CL`) diminui um pouco.

Rodando o modelo com Perceptron sobre o córpus segmentado obtemos os resultados da Tabela 33. Novamente, o treinamento com Perceptron melhora a taxa de acerto tanto sobre *que* quanto sobre *a*. O erro sobre *que* é reduzido em 3,45%, e sobre

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	*84,7775* 85,2207	*15,2225* 14,7793	*1683* 1634	*11056* 11056
a	*90,9726* 92,4109	*9,0274* 7,5890	*659* 554	*7300* 7300
Desconhecidas	*74,6631* 67,2619	*25,3369* 32,7381	*2256* 2915	*8904* 8904
Conhecidas	*97,1787* 97,1229	*2,8213* 2,8771	*7084* 7224	*251087* 251087
Total	*96,4076* 96,1002	*3,5924* 3,8998	*9340* 10139	*259991* 259991

Tabela 30: Resultados para o modelo VLMM+PERCEPTRON comparados aos do VLMM original (em itálico) no córpus normal.

MODELO		WPRO	C	WD	CONJ	FW	D	WD-P	% PREC.
VLMM	WPRO	<5569>	561	13	9	.	.	.	90,5234
	C	833	<3724>	6	31	.	.	.	81,0623
	WD	14	19	<68>	4	.	.	.	64,7619
	CONJ	99	91	.	<8>	.	.	.	4,0404
	FW	.	.	.	.	<4>	.	.	100,0000
	D	.	2	.	.	.	<.>	.	0,0000
	WD-P	.	.	1	.	.	.	<.>	0,0000
VLMM+PERCEPTRON	WPRO	<5496>	623	17	16	.	.	.	89,3368
	C	724	<3831>	18	21	.	.	.	83,3914
	WD	12	20	<70>	3	.	.	.	66,6667
	CONJ	81	92	.	<25>	.	.	.	12,6263
	FW	1	3	.	.	<.>	.	.	0,0000
	D	.	2	.	.	.	<.>	.	0,0000
	WD-P	.	.	1	.	.	.	<.>	0,0000

Tabela 31: Matrizes de confusão de *que* no córpus normal com os modelos VLMM e VLMM+PERCEPTRON (linha: referência; coluna: teste).

a 14,59%. E outra vez a taxa de acerto sobre palavras desconhecidas cai consideravelmente em relação ao modelo original.

Comparando as matrizes de confusão de *que*, mostradas na Tabela 34, novamente observamos que o treinamento com Perceptron causa uma predisposição maior em atribuir a etiqueta `C` ao invés da `WPRO`.

Como podemos ver nas matrizes de confusão de *a* da Tabela 35, a taxa de acerto sobre `P` aumenta consideravelmente em relação ao modelo original. Embora a taxa de acerto sobre `D-F` e `CL` caia um pouco, a taxa de acerto geral aumenta razoavelmente, conforme notado acima.

Testando o modelo VLMM+PERCEPTRON com uma VLMC de ordem menor, 5, obtemos os resultados da Tabela 36. O resultado é pior do que usando Perceptron com ordem 10, o que indica que, com essa forma de treinamento, ordens maiores capturam contextos mais relevantes.

MODELO		D-F	P	CL	FW	D	N	% PREC.
VLMM	D-F	<4023>	214	3	.	.	1	94,8597
	P	395	<2317>	7	.	.	.	85,2152
	CL	19	14	<296>	.	.	.	89,9696
	FW	2	.	.	<5>	.	.	71,4285
	D	3	.	.	.	<.>	.	0,0000
	N	1	.	.	.	.	<.>	0,0000
VLMM+PERCEPTRON	D-F	<4054>	174	11	.	2	.	95,5907
	P	315	<2401>	3	.	.	.	88,3045
	CL	25	15	<288>	.	.	1	87,5380
	FW	3	.	1	<3>	.	.	42,8571
	D	3	.	.	.	<.>	.	0,0000
	N	1	.	.	.	.	<.>	0,0000

Tabela 32: Matrizes de confusão de *a* no córpus normal com os modelos VLMM original e VLMM usando Perceptron (VLMM+PERCEPTRON) (linha: referência; coluna: teste).

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	*84,5250*	*15,4750*	*1712*	*11063*
	85,0583	14,9417	1653	11063
a	*94,5576*	*5,4424*	*740*	*13597*
	95,3519	4,6481	632	13597
Desconhecidas	*74,9214*	*25,0786*	*1995*	*7955*
	66,5996	33,4004	2657	7955
Conhecidas	*97,2642*	*2,7358*	*7348*	*268586*
	97,2039	2,7961	7510	268586
Total	*96,6215*	*3,3785*	*9343*	*276541*
	96,3235	3,6765	10167	276541

Tabela 33: Resultados do modelo VLMM+PERCEPTRON comparados aos do VLMM (em itálico) no córpus segmentado.

6.4 PROBLEMAS E CONCLUSÕES

O treinamento do modelo VLMM usando o algoritmo Perceptron aumentou a taxa de acerto de *que* e de *a*, mas não a taxa de acerto total. Esta, aliás, piorou consideravelmente, com a taxa de erro aumentando 30% em relação ao modelo VLMM inicial.

O uso da árvore de contexto permitiu a modelagem de cadeias longas, mesmo que não tenha sido podada como descrito na Seção 2.3.2. Uma cadeia mais longa obteve taxa de acerto melhor do que uma mais curta.

A questão que fica aberta e como trabalho futuro é um melhor treinamento para palavras desconhecidas, talvez alterando a forma como elas são tratadas ou melhorando a parametrização da árvore de sufixos. Uma melhora na eficiência sobre palavras desconhecidas talvez faça o modelo com Perceptron obter uma eficiência geral melhor que a do modelo original.

MODELO		WPRO	C	WD	CONJ	FW	D	WD-P	% PREC.
VLMM	WPRO	<5562>	575	14	8	.	.	.	90,3069
	C	850	<3712>	6	26	.	.	.	80,8010
	WD	13	20	<68>	4	.	.	.	64,7619
	CONJ	101	92	.	<5>	.	.	.	2,5253
	FW	.	.	.	.	<4>	.	.	100,0000
	D	.	2	.	.	.	<.>	.	0,0000
	WD-P	.	.	1	.	.	.	<.>	0,0000
VLMM+PERCEPTRON	WPRO	<5541>	584	22	12	.	.	.	89,9659
	C	794	<3771>	9	20	.	.	.	82,0853
	WD	14	15	<75>	1	.	.	.	71,4286
	CONJ	82	93	.	<23>	.	.	.	11,6162
	FW	1	3	.	.	<.>	.	.	0,0000
	D	.	2	.	.	.	<.>	.	0,0000
	WD-P	.	.	1	.	.	.	<.>	0,0000

Tabela 34: Matrizes de confusão de *que* no córpus segmentado com os modelos VLMM e VLMM+PERCEPTRON (linha: referência; coluna: teste).

MODELO		D-F	P	CL	FW	D	D-P	N	% PREC.
VLMM	D-F	<8056>	206	8	.	.	.	.	97,4123
	P	433	<4411>	17	.	.	.	.	90,7426
	CL	29	39	<386>	.	.	.	.	85,0220
	FW	2	1	.	<4>	.	.	.	57,1428
	D	3	.	.	.	<.>	.	.	0,0000
	D-P	1	.	.	.	.	<.>	.	0,0000
	N	1	.	.	.	.	.	<.>	0,0000
VLMM+PERCEPTRON	D-F	<8009>	237	15	.	2	2	4	96,8440
	P	285	<4569>	7	.	.	.	.	93,9930
	CL	25	46	<383>	.	.	.	.	84,3612
	FW	3	.	.	<4>	.	.	.	57,1429
	D	3	.	.	.	<.>	.	.	0,0000
	D-P	1	.	.	.	.	<.>	.	0,0000
	N	1	.	.	.	.	.	<.>	0,0000

Tabela 35: Matrizes de confusão de *a* no córpus segmentado com os modelos VLMM original e VLMM usando Perceptron (VLMM+PERCEPTRON) (linha: referência; coluna: teste).

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	84,2348	15,7652	1743	11056
a	91,6027	8,3973	613	7300
Desconhecidas	64,0162	35,9838	3204	8904
Conhecidas	96,9736	3,0264	7599	251087
Ambíguas	95,0555	4,9445	6723	135969
Total	95,8449	4,1551	10803	259991

Tabela 36: Resultados do modelo VLMM+PERCEPTRON de ordem 5 sobre o córpus normal.

7 MODELAGEM BI-DIRECIONAL SIMPLES

Gostaríamos de verificar se a palavra *a* realmente perde um pouco da ambiguidade se considerarmos o contexto da direita. O problema que torna essa abordagem não trivial é que o contexto de etiquetas da direita não está ainda disponível quando uma palavra está sendo etiquetada. Isso por causa da ordem natural da esquerda-para-direita com que os etiquetadores comumente analisam uma sentença. O contexto direito que está disponível é o de palavras, mas ele apresenta o problema da esparsidade se for tratado de maneira geral.

Uma abordagem é então modelar o contexto de etiquetas da direita quando as palavras à direita não forem ambíguas, isso é, quando elas puderem ter apenas uma etiqueta atribuída, de acordo com o léxico do treinamento. Essa abordagem é implementada construindo-se, durante o treinamento, uma nova árvore de contexto para o contexto direito. Para cada palavra em uma sentença, uma sequência contínua mas de tamanho variável de etiquetas de palavras não ambíguas é adicionada como galho à árvore de contexto direito. Ou seja, se k palavras à direita de uma determinada palavra forem não ambíguas, então a sequência de k etiquetas que estas palavras terão é adicionada à árvore direita. Essa nova árvore direita também é podada como a árvore de contexto esquerdo, e o algoritmo de Viterbi é adaptado para considerar esses novos parâmetros. A equação para fornecer a probabilidade de uma sequência de etiquetas $t_{1,n}$ associada a uma sequência de palavras $w_{1,n}$, baseada na Equação 2.9, é agora definida como:

$$P(w_{1,n}, t_{1,n}) = \prod_{i=1}^{n} P(t_i|c(t_{i-k,i-1}))P(w_i|t_i)P(t_i|c_r(t^*_{i+1,i+k})),$$

onde a função $c_r(\cdot)$ é dada pela árvore de contexto direito, e $t^*_{i+1,i+k}$ denota a sequência de até k etiquetas se a sequência de palavras $w_{i+1,i+k}$ não for ambígua.

7.1 ETIQUETAGEM COM CONTEXTO DIREITO

Utilizando a abordagem descrita acima, implementamos tal modelo, que chamaremos de VLMM+RIGHT, e o testamos sobre os córpus normal e segmentado. A Tabela 37 mostra os resultados sobre o córpus normal, e a Tabela 38 sobre o córpus segmentado. Em ambos casos, houve piora no desempenho, em todos quesitos considerados. Tanto *que* quanto *a* têm a taxa de erro aumentada.

Para testar a influência do contexto direito diretamente sobre *a*, implementamos um modelo VLMM+RIGHT-A, que usa parâmetros para o contexto direito apenas para *a*. O resultado sobre o córpus normal é mostrado na Tabela 39. Dessa vez a taxa de acerto sobre *a* aumenta, com uma redução do erro de 14,26%.

A Tabela 40 mostra as matrizes de confusão de *a* usando o modelo VLMM original e o modelo VLMM+RIGHT-A sobre o córpus normal. Analisando-as, vemos que a taxa de acerto melhora tanto sobre `D-F` quanto sobre `P`, diferentemente do que ocorre com a abordagem da análise sintática superficial, que melhora a taxa de acerto apenas para uma das etiquetas.

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	*84,7775* 79,7847	*15,2225* 20,2153	*1683* 2235	*11056* 11056
a	*90,9726* 88,8904	*9,0274* 11,1096	*659* 811	*7300* 7300
Desconhecidas	*74,6631* 70,1146	*25,3369* 29,8854	*2256* 2661	*8904* 8904
Conhecidas	*97,1787* 95,7529	*2,8213* 4,2471	*7084* 10664	*251087* 251087
Total	*96,4076* 94,8748	*3,5924* 5,1252	*9340* 13325	*259991* 259991

Tabela 37: Resultados para o modelo VLMM+RIGHT comparados aos do VLMM original (em itálico) no córpus normal.

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	*84,5250* 79,1738	*15,4750* 20,8262	*1712* 2304	*11063* 11063
a	*94,5576* 94,2267	*5,4424* 5,7733	*740* 785	*13597* 13597
Desconhecidas	*74,9214* 69,3652	*25,0786* 30,6348	*1995* 2437	*7955* 7955
Conhecidas	*97,2642* 96,0005	*2,7358* 3,9995	*7348* 10742	*268586* 268586
Total	*96,6215* 95,2343	*3,3785* 4,7657	*9343* 13179	*276541* 276541

Tabela 38: Resultados para o modelo VLMM+RIGHT comparados aos do VLMM original (em itálico) no córpus segmentado.

Testando o modelo VLMM+RIGHT-A sobre o córpus segmentado obtemos os resultados mostrados na Tabela 41. Novamente, a taxa de acerto de *a* melhora em relação ao modelo original, mesmo com o número de ocorrências de *a* quase duplicado em relação ao córpus normal.

A Tabela 42 mostra as matrizes de confusão das etiquetas de *a* quando executados o modelo original e o de contexto direito sobre o córpus segmentado. A taxa de acerto sobre a etiqueta `D-F` aumenta, mas ao contrário de quando usamos o córpus normal, a taxa de acerto sobre `P` diminui. O número de etiquetas `P` atribuídas pelo etiquetador VLMM+RIGHT-A, aliás, diminui em relação ao etiquetador original, ao passo que o de `D-F` aumenta. A causa parece ser o número de palavras *a* como determinantes que aumentou bastante em relação ao córpus normal, devido a segmentações do tipo `P+D-F` com *a*. Esse aumento de determinantes femininos faz o etiquetador tender a classificar *a* como tal.

Para verificar se *que* de alguma forma apresenta dependência de contexto direito, executamos um modelo do etiquetador da mesma forma que acima, mas apenas para a palavra *que*. Os resultados obtidos por esse modelo, que chamaremos de

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	*84,7775* 84,8408	*15,2225* 15,1592	*1683* 1676	*11056* 11056
a	*90,9726* 92,2603	*9,0274* 7,7397	*659* 565	*7300* 7300
Desconhecidas	*74,6631* 74,6743	*25,3369* 25,3257	*2256* 2255	*8904* 8904
Conhecidas	*97,1787* 97,2145	*2,8213* 2,7855	*7084* 6994	*251087* 251087
Total	*96,4076* 96,4426	*3,5924* 3,5574	*9340* 9249	*259991* 259991

Tabela 39: Resultados do modelo VLMM+RIGHT-A comparados aos do VLMM original (em itálico) no córpus normal.

MODELO		D-F	P	CL	FW	D	N	% PREC.
VLMM	D-F	<4023>	214	3	.	.	1	94,8597
	P	395	<2317>	7	.	.	.	85,2152
	CL	19	14	<296>	.	.	.	89,9696
	FW	2	.	.	<5>	.	.	71,4285
	D	3	.	.	.	<.>	.	0,0000
	N	1	.	.	.	.	<.>	0,0000
VLMM+RIGHT-A	D-F	<4080>	151	8	1	.	1	96,2037
	P	357	<2354>	8	.	.	.	86,5759
	CL	20	13	<296>	.	.	.	89,9696
	FW	2	.	.	<5>	.	.	71,4285
	D	3	.	.	.	<.>	.	0,0000
	N	1	.	.	.	.	<.>	0,0000

Tabela 40: Matrizes de confusão de *a* no córpus normal com os modelos VLMM original e VLMM usando contexto direto para *a* (VLMM+RIGHT-A) (linha: referência; coluna: teste).

VLMM+RIGHT-QUE, são mostrados na Tabela 43. A taxa de acerto geral cai por volta de 0,8%, o que significa um aumento nos erros de 2,20%. Além disso, há um aumento no erro sobre *que* de quase 12%. Isso significa que, ao contrário do que ocorre com *a*, *que* não possui dependência local direita, ao menos não quando o contexto direito não é ambíguo. As matrizes de confusão de *que* obtidas com esse modelo são mostradas na Tabela 44, e mostram que o modelo VLMM+RIGHT-QUE apresenta maior confusão entre as etiquetas WPRO e C.

7.2 PROBLEMAS E CONCLUSÕES

Conforme intuímos, o uso do contexto direito para *a* reduz a sua ambiguidade. Entretanto, o contexto direito não é um parâmetro que funciona para todas palavras. Quando testamos utilizá-lo para qualquer palavra, a taxa de acerto alcançada foi

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	*84,5250* 84,5792	*15,4750* 15,4208	*1712* 1706	*11063* 11063
a	*94,5576* 94,9327	*5,4424* 5,0673	*740* 689	*13597* 13597
Desconhecidas	*74,9214* 74,9466	*25,0786* 25,0534	*1995* 1993	*7955* 7955
Conhecidas	*97,2642* 97,2832	*2,7358* 2,7168	*7348* 7297	*268586* 268586
Total	*96,6215* 96,6406	*3,3785* 3,3594	*9343* 9290	*276541* 276541

Tabela 41: Resultados para o modelo VLMM+RIGHT-A comparados aos do VLMM original (em itálico) no córpus segmentado.

MODELO		D-F	P	CL	FW	D	D-P	N	P (%)
VLMM	D-F	<8056>	206	8	.	.	.	.	97,4123
	P	433	<4411>	17	.	.	.	.	90,7426
	CL	29	39	<386>	.	.	.	.	85,0220
	FW	2	1	.	<4>	.	.	.	57,1428
	D	3	.	.	.	<.>	.	.	0,0000
	D-P	1	.	.	.	.	<.>	.	0,0000
	N	1	.	.	.	.	.	<.>	0,0000
VLMM+RIGHT-A	D-F	<8149>	102	18	.	.	.	.	98,5488
	P	475	<4375>	11	.	.	.	.	90,0021
	CL	35	40	<379>	.	.	.	.	83,4802
	FW	2	.	.	<5>	.	.	.	71,4295
	D	3	.	.	.	<.>	.	.	0,0000
	D-P	1	.	.	.	.	<.>	.	0,0000
	N	.	.	1	.	.	.	<.>	0,0000

Tabela 42: Matrizes de confusão de *a* no córpus segmentado com os modelos VLMM original e VLMM usando contexto direito para *a* (VLMM+RIGHT-A) (linha: referência; coluna: teste).

bem pior que a que o modelo original alcança, tanto no córpus normal quanto no segmentado. Quando testamos apenas sobre *que*, a taxa de acerto também piorou, indicando que *que* não depende de contexto direito local obtido desta maneira simples.

Sobre o córpus normal o modelo com contexto direito para *a* aprendeu melhor a classificar determinantes femininos e preposições. Sobre o córpus segmentado, o modelo apresentou maior tendência em classificar *a* como determinante, o que aumentou a taxa de acerto nisso mas diminui um pouco a taxa de acerto sobre *a* como preposição.

Embora seja possível, utilizando VLMMs, modelar uma janela longa para o contexto direito, o problema é que a inferência é feita da esquerda para a direita. Isso limita o uso que podemos fazer do contexto direito, embora, como mostramos, um uso

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	*84,7775* 82,9595	*15,2225* 17,0405	*1683* 1884	*11056* 11056
a	*90,9726* 90,9589	*9,0274* 9,0411	*659* 660	*7300* 7300
Desconhecidas	*74,6631* 74,7192	*25,3369* 25,2808	*2256* 2251	*8904* 8904
Conhecidas	*97,1787* 97,0946	*2,8213* 2,9054	*7084* 7295	*251087* 251087
Total	*96,4076* 96,3283	*3,5924* 3,6717	*9340* 9546	*259991* 259991

Tabela 43: Resultados do modelo VLMM+RIGHT-QUE comparados aos do VLMM original (em itálico) no córpus normal.

MODELO		WPRO	C	WD	CONJ	FW	D	WD-P	% PREC.
VLMM	WPRO	<5569>	561	13	9	.	.	.	90,5234
	C	833	<3724>	6	31	.	.	.	81,0623
	WD	14	19	<68>	4	.	.	.	64,7619
	CONJ	99	91	.	<8>	.	.	.	4,0404
	FW	.	.	.	.	<4>	.	.	100,0000
	D	.	2	.	.	.	<.>	.	0,0000
	WD-P	.	.	1	.	.	.	<.>	0,0000
VLMM+RIGHT-QUE	WPRO	<5419>	713	13	7	.	.	.	88,0852
	C	897	<3668>	12	17	.	.	.	79,8433
	WD	11	18	<74>	2	.	.	.	70,4762
	CONJ	97	94	.	<7>	.	.	.	3,5354
	FW	.	.	.	.	<4>	.	.	100,0000
	WD-P	.	.	1	.	.	<.>	.	0,0000
	D	.	2	.	.	.	.	<.>	0,0000

Tabela 44: Matrizes de confusão de *que* no córpus normal com os modelos VLMM e VLMM+RIGHT-QUE (linha: referência; coluna: teste).

simples é vantajoso sobre *a*. Novas abordagens são necessárias para investigar a total importância desse contexto sobre todas palavras na etiquetagem morfossintática.

8 APRENDIZADO GUIADO PARA MARKOV DE ORDEM 3

Conforme notado no Capítulo 7, a etiquetagem tradicional ocorre da esquerda para a direita. Como consequência o contexto direito completo de etiquetas não está disponível. Mesmo assim, mostramos que uma abordagem simples já auxilia a classificação de *a*.

Uma das abordagens que propõe formas de se utilizar mais contextos é a proposta por Tsuruoka e Tsujii (2005), chamada "*Mais fácil primeiro*"[1], e que alcança 97,10% de acerto no inglês. Nela, a ordem de inferência não necessariamente ocorre da esquerda para a direita. A estratégia é primeiro etiquetar palavras que apresentam menos ambiguidade, e então usar as etiquetas já disponíveis como contexto para etiquetar as palavras mais difíceis. Entretanto, essa abordagem serve apenas como uma regra heurística. A ordem de inferência não é incorporada no treinamento.

Shen, Satta e Joshi (2007) propõem algoritmos baseados na estratégia *mais fácil primeiro* e no algoritmo perceptron (Collins, 2002), que incluem o treinamento da ordem em que a inferência é feita. A taxa de acerto alcançada é de 97,33%, que é o estado da arte para o inglês. Apresentamos esses algoritmos na próxima seção.

8.1 CLASSIFICAÇÃO BIDIRECIONAL DE SEQUÊNCIAS

A idéia é modelar hipóteses de etiquetas para intervalos de palavras, e utilizar ações para atribuir etiquetas a palavras ainda não etiquetadas, de acordo com as hipóteses disponíveis nos contextos esquerdo e direito. O algoritmo de inferência é apresentado na próxima seção, e em seguida mostramos o algoritmo de treinamento, que é baseado no algoritmo perceptron.

8.1.1 Algoritmo de Inferência

Relembrando a notação, $w_1, w_2, \ldots, w_n$ é uma sequência de palavras. A cada palavra w_i é associada uma etiqueta $t_i \in T$, T sendo o conjunto de etiquetas. Chamamos uma subsequência $w_i, \ldots, w_j$ de intervalo, denotado por $[i, j]$. A cada intervalo v associamos uma ou mais sequências sobre T que possuem comprimento $|v|$. Chamamos essas sequências de hipóteses.

As etiquetas nas fronteiras de uma hipótese são usadas para etiquetar as palavras fora do intervalo v. No caso de Shen, Satta e Joshi (2007), um modelo trigrama é usado. Assim, para etiquetar a palavra w_i, as duas etiquetas (t_{i-2}, t_{i-1}) da fronteira do intervalo à esquerda $[k, i-1]$, $0 \leqslant k \leqslant i-1$, podem ser usadas se esse intervalo já tiver sido etiquetado. Da mesma forma, podemos usar as duas etiquetas (t_{i+1}, t_{i+2}) à direita se o intervalo $[i+1, j]$ já tiver sido etiquetado. Nos referiremos às duas etiquetas da esquerda como interface esquerda, I_{esq}, e às duas etiquetas da direita como interface direita, I_{dir}.

1 Do inglês, *Easiest-first*.

Denotamos as fronteiras de um intervalo v por $b = (I_{esq}, I_{dir})$, também chamado estado. Particionamos as hipóteses associadas a um intervalo v em conjuntos compatíveis com a mesma fronteira, isto é, em hipóteses que possuem o mesmo b. Para cada intervalo v usa-se uma matriz M_v indexada por b, tal que $M_v(b)$ é o conjunto de todas hipóteses associadas a v que possuem I_{esq} e I_{dir} compatíveis.

Para um intervalo v, a melhor hipótese é denotada por

$$h_v^* = \underset{h \in M_v(b), \forall b : M_v(b) \neq \emptyset}{\arg\max} X(h),$$

onde $X(h)$ é a função de pontuação de uma hipótese (definida na Equação 8.1 abaixo). Em outras palavras, para cada intervalo v existe uma melhor hipótese h_v^* que possui a pontuação mais alta para o intervalo v.

Hipóteses são iniciadas e aumentadas através de ações de etiquetagem. Há três tipos de ações:

1. Iniciar um novo intervalo etiquetando uma palavra sem contexto;
2. Expandir um intervalo existente etiquetando uma palavra adjacente;
3. Mesclar dois intervalos etiquetando a palavra entre eles.

Nesse último caso, os dois intervalos originais seriam subsequências do intervalo resultante, e a ação de etiquetagem da palavra entre os intervalos usaria informação de ambos contextos, da direita e da esquerda.

Para cada hipótese h de um intervalo v, mantemos a ação de etiquetagem mais recente $a(h)$ envolvendo uma palavra dentro de v, assim como as hipóteses, se houver, que foram usadas por tal ação como contexto esquerdo, $h_L^*(h)$, e contexto direito, $h_R^*(h)$. Note que $h_L^*(h)$ e $h_R^*(h)$ se referem a intervalos que são subsequências de v.

A pontuação de uma hipótese h é a soma da pontuação da ação mais recente $a(h)$ e as pontuações das melhores hipóteses das fronteiras de contexto. Ou seja, a pontuação de h é computada recursivamente por

$$X(h) = X(h_L^*(h)) + X(h_R^*(h)) + U(a(h)), \tag{8.1}$$

onde U é a função pontuação de uma ação. U é computada através de uma combinação linear do vetor de pesos $\mathbf{w}$ e o vetor de *traços*[2] da ação, $\mathbf{f}(a(h))$:

$$U(a(h)) = \mathbf{w}.\mathbf{f}(a(h)). \tag{8.2}$$

Para reduzir o espaço de busca explorado durante o algoritmo de inferência, o algoritmo de *Busca por Feixe*[3] é usado. Busca por feixe é um algoritmo de busca heurístico que otimiza a busca em largura através da redução do espaço de estados considerados, assim diminuindo a quantidade de memória necessária. Isso significa, por outro lado, que completude[4] e otimalidade[5] são sacrificados. O parâmetro usado pela busca por feixe é a largura do feixe, ou seja, o número de melhores estados candidatos mantidos a cada passo de execução. No algoritmo de inferência do Aprendizado Guiado, a largura B do feixe determina o número máximo de fronteiras b mantidas para cada intervalo v.

O Algoritmo 4 mostra o algoritmo de inferência. O valor de B é dado como entrada do algoritmo, assim como o vetor de pesos $\mathbf{w}$ e a sequência de palavras $w_{1,n}$ a ser

2 *Features*, em inglês.
3 Tradução livre do termo em inglês *Beam Search*.
4 Completude: garantia de término.
5 Otimalidade: garantia de encontrar a solução ótima.

etiquetada. O conjunto P contém a lista de intervalos aceitos, e Q a fila de intervalos candidatos. Primeiro P é inicializado com o conjunto vazio. Depois, Q é inicializada com os intervalos candidatos $[i, i]$ para cada w_i em $w_{1,n}$, e, para cada etiqueta $t \in T$ possível para w_i, fazemos

$$M_{[i,i]}((t, t)) = i \to t,$$

onde $i \to t$ representa a hipótese que consiste de uma única ação que não utiliza contexto e atribui a etiqueta t a w_i. Esta matriz M fornece o conjunto de hipóteses iniciais.

Entrada: $w_{1,n}$: sequência de palavras.
Entrada: B: largura do feixe de busca.
Entrada: **w**: vetor de pesos.

1: inicialize $P \leftarrow \emptyset$, o conjunto de intervalos aceitos
2: inicialize Q, a fila de intervalos candidatos
3: inicialize M_v para cada $v \in Q$
4: **repita**
5: intervalo $v' \leftarrow \arg\max_{v \in Q} U(a(h_v^*))$
6: atualize P com v'
7: atualize Q com v' e P
8: **até** $Q = \emptyset$

Algoritmo 4: Algoritmo de inferência para aprendizado guiado.

O laço de repetição das linhas 4-8 seleciona de Q um intervalo v' cuja melhor hipótese possui a ação com maior pontuação. Isso representa a ação de etiquetagem com maior grau de confiabilidade. P é então atualizado com a inserção de v' e a remoção dos intervalos incluídos em v', se houver. Esses intervalos removidos de P são usados para atualizar Q: cada intervalo de Q que usa como contexto um dos intervalos removidos é substituído por um novo intervalo candidato que usa v' como contexto, sempre mantendo-se B diferentes fronteiras para cada intervalo. O algoritmo termina quando Q estiver vazia e P contiver um intervalo que cobre a sentença inteira.

A cada passo do algoritmo um novo intervalo é inserido em P ou intervalos já presentes em P são estendidos. P nunca possui dois intervalos que se sobrepõem, e o número de pares de intervalos em Q que se sobrepõem são delimitados por uma constante. Assim, o algoritmo executa no máximo n iterações, e seu tempo de execução é $O(B^2 n)$, linear no comprimento da sequência de entrada.

Adaptando o exemplo de Shen, Satta e Joshi (2007), considere a sentença

tornaram	*a*	*bradar*	*as*	*sentinelas*
w_1	w_2	w_3	w_4	w_5

Inicialmente,

$$P = \emptyset$$
$$Q = \{[1,1], [2,2], [3,3], [4,4], [5,5]\}$$

Suponha que `VB` e `N` sejam as duas etiquetas possíveis para w_3, `bradar`. Temos então

$$M_{[3,3]}(\texttt{N}, \texttt{N}) = \{h^1_{[3,3]} = 3 \to \texttt{N}\}$$
$$M_{[3,3]}(\texttt{VB}, \texttt{VB}) = \{h^2_{[3,3]} = 3 \to \texttt{VB}\}$$

A ação mais recente da hipótese $h^2_{[3,3]}$ é atribuir `VB` a w_3. De acordo com a equação 8.2, a pontuação dessa ação, $U(a(h^2_{[3,3]}))$, depende dos *traços* definidos no contexto local da ação, por exemplo:

$$f_{1001}(a(h^2_{[3,3]})) = \begin{cases} 1 & \text{se } t = \texttt{VB} \wedge w^{-1} = a \\ 0 & \text{caso contrário,} \end{cases}$$

onde w^{-1} representa a palavra à esquerda. Para todos os *traços* que dependem de etiquetas no contexto o valor será 0 por enquanto, porque ainda não há palavras etiquetadas. Como o *traço* acima não depende de outras etiquetas, o valor da hipótese $h^2_{[3,3]}$ será, de acordo com a equação 8.1, $X(h^2_{[3,3]}) = U(a(h^2_{[3,3]}))$.

Digamos que, das duas hipóteses para w_3, a segunda seja mais provável dados os *traços* baseadas em palavras. Como as duas melhores hipóteses são mantidas ($B = 2$), teremos

		`VB`		
		`N`		
tornaram	*a*	*bradar*	*as*	*sentinelas*
w_1	w_2	w_3	w_4	w_5

Suponha agora que a ação mais favorável seja atribuir `N-P` a `sentinelas`.

		`VB`		`N-P`
		`N`		`N-P`
tornaram	*a*	*bradar*	*as*	*sentinelas*
w_1	w_2	w_3	w_4	w_5

Temos dois intervalos separados já etiquetados. Nessa configuração, teríamos P e Q assim:

$$P = \{[3,3],[5,5]\}$$
$$Q = \{[1,1],[2,3],[3,5]\}$$

Há três intervalos candidatos em Q, cada um com suas hipóteses associadas e ações mais recentes. Podemos resolver:

- w_1, que não possui hipóteses de contexto, e resultaria no intervalo $[1,1]$;
- w_2, baseados nas hipóteses de $[3,3]$, que expandiria o intervalo $[3,3]$ para $[2,3]$;
- w_4, baseados nas hipóteses de contexto em $[3,3]$ e $[5,5]$, que resultaria no intervalo mesclado $[3,5]$.

No caso de w_4, primeiro são computadas as hipóteses resultantes de todas possíveis atribuições de etiquetas a w_4 sob todas possíveis combinações de fronteiras dos intervalos de contexto, $[3,3]$ e $[5,5]$. Digamos que w_4 possa ter as etiquetas `D-F-P` e `CL`. Dados os contextos `VB` à esquerda e `N-P` à direita, e `N` à esquerda e `N-P` à direita, a pontuação das novas hipóteses, de acordo com a equação 8.1, é dada por:

$$X(h^1_{[3,5]}) = X(h^2_{[3,3]}) + X(h^1_{[5,5]}) + \mathbf{w}.\mathbf{f}(a(h^1_{[3,5]}))$$
$$X(h^2_{[3,5]}) = X(h^1_{[3,3]}) + X(h^1_{[5,5]}) + \mathbf{w}.\mathbf{f}(a(h^2_{[3,5]}))$$

Desta vez, os *traços* para as ações das hipóteses podem depender das etiquetas dos contextos esquerdo e direito, já que elas já foram resolvidas. Por exemplo, podemos ter *traços* como

$$f_{2002}(a(h^1_{[3,5]})) = \begin{cases} 1 & \text{se } t = \texttt{D-F-P} \wedge t^{+1} = \texttt{N-P} \\ 0 & \text{caso contrário.} \end{cases}$$

Com B = 2, mantemos as duas melhores fronteiras que possuem as hipóteses com pontuação mais alta. Vamos supor que a etiqueta mais provável seja D-F-P quando houver VB à esquerda e N-P à direita, e CL quando houver N à esquerda. As duas melhores fronteiras para o intervalo [3, 5] serão então

$$M_{[3,5]}(\texttt{VB} - \texttt{D-F-P}, \texttt{D-F-P} - \texttt{N-P}) = \{h^1_{[3,5]} = (\texttt{VB}, \texttt{VB})4 \rightarrow \texttt{D-F-P}(\texttt{N-P}, \texttt{N-P})\}$$

e

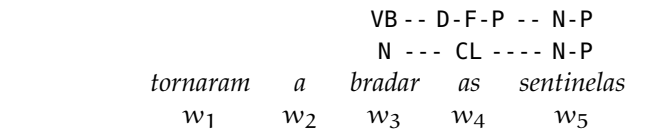
$$M_{[3,5]}(\texttt{N} - \texttt{CL}, \texttt{CL} - \texttt{N-P}) = \{h^2_{[3,5]} = (\texttt{N}, \texttt{N})4 \rightarrow \texttt{CL}(\texttt{N-P}, \texttt{N-P})\}.$$

Aqui, $(\texttt{VB}, \texttt{VB})4 \rightarrow \texttt{D-F-P}(\texttt{N-P}, \texttt{N-P})$ representa a hipótese resultante da ação de atribuir D-F-P a w_4 dada a fronteira de contexto esquerdo (VB, VB) e a fronteira de contexto direito (N-P, N-P). O mesmo sentido tem $(\texttt{N}, \texttt{N})4 \rightarrow \texttt{CL}(\texttt{N-P}, \texttt{N-P})$.

De maneira similar, calculamos as melhores hipóteses e fronteiras para os intervalos [1, 1] e [2, 3]. Vamos supor que a hipótese com maior pontuação de ação seja $h^1_{[3,5]}$. Atualizamos então P inserindo [3, 5] e removendo [3, 3] e [5, 5], que foram mesclados com 4 e agora são cobertos por [3, 5]. Atualizamos também Q, removendo [3, 5] e [2, 3] (que dependia de [3, 3] que estava e P), e inserindo o novo intervalo candidato [2,5]. A configuração resultante é

$$P = \{[3, 5]\}$$
$$Q = \{[1, 1], [2, 5]\},$$

e o exemplo fica



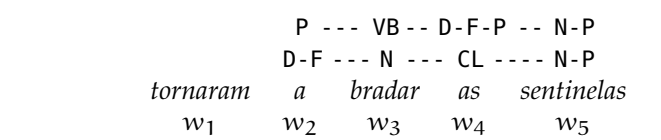
Podemos agora resolver w_1, ainda sem contexto de etiquetas, ou w_2, baseado nas hipóteses em [3, 5]. As hipóteses possíveis para [2, 5], supondo que w_2 possa ter as etiquetas P ou D-F, são:

$$h^1_{[2,5]} = 2 \rightarrow \texttt{P}(\texttt{VB} - \texttt{D-F-P})$$
$$h^2_{[2,5]} = 2 \rightarrow \texttt{P}(\texttt{N} - \texttt{CL})$$
$$h^3_{[2,5]} = 2 \rightarrow \texttt{D-F}(\texttt{VB} - \texttt{D-F-P})$$
$$h^4_{[2,5]} = 2 \rightarrow \texttt{D-F}(\texttt{N} - \texttt{CL})$$

Vamos supor que a hipótese com maior pontuação de ação seja a 1, seguida pela 4. Atualizamos P e Q

$$P = \{[2, 5]\}$$
$$Q = \{[1, 5]\},$$

e o exemplo fica



No último passo a etiqueta de w_1 é resolvida com base nas hipóteses de contexto do intervalo [2, 5], e o intervalo [1, 5] é inserido em P e $Q = \emptyset$.

8.1.2 Algoritmo de Treinamento

Shen, Satta e Joshi (2007) propõem o algoritmo *Aprendizado Guiado*, um algoritmo baseado em Perceptron usado para aprender os valores do vetor de pesos $\mathbf{w}$. Ele é mostrado no Algoritmo 5 . Um conjunto X de S sequências de palavras é dado como entrada, junto com um conjunto Y de etiquetas alinhado com X. Também como entrada são dados R, o número de iterações de treinamento, e B, a largura do feixe de busca.

Entrada: $\{(X_s, Y_s)\}_{1 \leqslant s \leqslant S}$: pares de sequências de treinamento.
Entrada: R: número de iterações sobre o conjunto de treinamento.
Entrada: B: largura do feixe de busca.

1: $\mathbf{w} \leftarrow 0$
2: **para todo** $r, 1 \leqslant r \leqslant R$, **faça**
3: **para todo** $s, 1 \leqslant s \leqslant S$, **faça**
4: carregue as sequências de palavras X_s e etiquetas Y_s
5: inicialize $P \leftarrow \emptyset$, o conjunto de intervalos aceitos
6: inicialize Q, a fila de intervalos candidatos
7: **repita**
8: intervalo $v' \leftarrow \arg\max_{v \in Q} U(a(h^*_v))$
9: **se** $h^*_{v'} = h^Y_{v'}$ **então**
10: atualize P com v'
11: atualize Q com v' e P
12: **senão**
13: *promova*$(\mathbf{w}, \mathbf{f}(a(h^Y_{v'})))$
14: *rebaixe*$(\mathbf{w}, \mathbf{f}(a(h^*_{v'})))$
15: regenere Q com $\mathbf{w}$ e P
16: **até** $Q = \emptyset$

Algoritmo 5: Algoritmo *aprendizado guiado*.

Primeiro P e Q são inicializados da mesma forma que no algoritmo de inferência: $P = \emptyset$ e $Q = \{[1,1],[2,2],[3,3],[4,4],[5,5]\}$. Então o laço de repetição começa selecionando de Q o intervalo candidato v' para a próxima ação de etiquetagem. Se a melhor hipótese para v' é igual à etiqueta do córpus de treinamento, então P e Q são atualizados como no algoritmo de inferência. Caso contrário, o vetor de pesos $\mathbf{w}$ é atualizado através da promoção dos *traços* do córpus de treinamento e o rebaixamento dos *traços* da ação da hipótese candidata, como no algoritmo perceptron. Compare com a linha 6 do Algoritmo 2. Depois disso a última ação de etiquetagem é desfeita – os elementos em Q são substituídos por novos intervalos candidatos baseados nos intervalos em P – e novas pontuações são calculadas com o vetor de pesos atualizado.

Shen, Satta e Joshi (2007) e Gesmundo (2009) usam Perceptron Médio (Collins, 2002) e Perceptron com Margem (Krauth e Mézard, 1987).

8.2 EXPERIMENTOS E RESULTADOS

Os algoritmos de inferência e aprendizado guiado foram implementados por Shen, Satta e Joshi (2007) em Java, como parte de seu etiquetador bidirecional. Vamos nos referir ao modelo que usa aprendizado guiado como GL (*Guided Learning*). Ele

usa trigramas para os contextos esquerdo e direito, e portanto poderia ser potencialmente estendido pelo uso de VLMCs. É nosso objetivo desenvolver um etiquetador combinando os modelos VLMM e GL. Entretanto, atualmente, ainda não terminamos com sucesso uma implementação desse modelo em C++, a fim de combiná-lo com o código do etiquetador VLMM. Mesmo assim, para testar os possíveis benefícios do aprendizado guiado, executamos o etiquetador bidirecional original[6] sobre nossos córpus de treinamento e de teste.

A Tabela 45 mostra os resultados obtidos sobre o córpus normal. A primeira coisa a notar é que o modelo GL faz um bom trabalho de etiquetagem. A taxa de acerto geral corresponde a uma redução de 10% no erro em relação ao VLMM original. Mas o detalhe mais importante é em relação às duas palavras que estamos focando. A taxa de acerto de *a* aumenta consideravelmente, com uma redução no erro de praticamente 50%. Já a taxa de acerto de *que* praticamente não se altera. Além disso, é interessante notar que no modelo GL a taxa de acerto sobre palavras desconhecidas é bastante maior do que a do modelo VLMM, enquanto a taxa de acerto sobre palavras conhecidas acaba sendo menor. A principal diferença no desempenho do aprendizado guiado, além da taxa de acerto sobre *a*, está justamente na taxa de acerto sobre palavras desconhecidas. A diferença no número de erros total entre o modelo GL e o VLMM é de 690 palavras, enquanto nas palavras desconhecidas a diferença é de 878 palavras acertadas a mais no etiquetador GL. Isso significa que há palavras que o modelo VLMM classifica melhor do que o modelo GL.

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	*84,7775* 84,8951	*15,2225* 15,1049	*1683* 1670	*11056* 11056
a	*90,9726* 95,4932	*9,0274* 4,5068	*659* 329	*7300* 7300
Desconhecidas	*74,6631* 84,5238	*25,3369* 15,4762	*2256* 1378	*8904* 8904
Conhecidas	*97,1787* 97,1038	*2,8213* 2,8962	*7084* 7272	*251087* 251087
Total	*96,4076* 96,6730	*3,5924* 3,3270	*9340* 8650	*259991* 259991

Tabela 45: Resultados da classificação bidirecional usando Aprendizado Guiado comparados aos do VLMM original (em itálico) no córpus normal.

A Tabela 46 mostra as matrizes de confusão de *que* sobre o córpus normal obtidas pelos modelos VLMM e GL. As diferenças são poucas, com o modelo VLMM acertando mais a etiqueta WD e o GL a etiqueta CONJ.

Em relação a *a*, a Tabela 47 mostra as matrizes de confusão no córpus normal. A taxa de acerto sobre as etiquetas D-F e P aumenta, principalmente em P. A das outras etiquetas permanece praticamente a mesma entre os dois modelos. Portanto, a melhor taxa de acerto do modelo GL sobre D-F e P vem de uma menor confusão entre elas, com o número de atribuições de uma quando o correto era a outra diminuindo bastante.

6 Agradecemos ao Mark Dredze (Dredze e Wallenberg, 2008) por ter-nos fornecido o código de Shen, Satta e Joshi (2007).

MODELO		WPRO	C	WD	CONJ	FW	D	WD-P	% PREC.
VLMM	WPRO	<5569>	561	13	9	.	.	.	90,5234
	C	833	<3724>	6	31	.	.	.	81,0623
	WD	14	19	<68>	4	.	.	.	64,7619
	CONJ	99	91	.	<8>	.	.	.	4,0404
	FW	.	.	.	.	<4>	.	.	100,0000
	D	.	2	.	.	.	<.>	.	0,0000
	WD-P	.	.	1	.	.	.	<.>	0,0000
GL	WPRO	<5596>	546	7	3	.	.	.	90,9622
	C	861	<3726>	3	4	.	.	.	81,1057
	WD	31	30	<44>	.	.	.	.	41,9047
	CONJ	86	91	1	<20>	.	.	.	10,1010
	FW	3	1	.	.	<.>	.	.	0,0000
	D	.	2	.	.	.	<.>	.	0,0000
	WD-P	.	.	1	.	.	.	<.>	0,0000

Tabela 46: Matrizes de confusão de *que* no córpus normal com os modelos VLMM e GL (linha: referência; coluna: teste).

MODELO		D-F	P	CL	FW	D	N	% PREC.
VLMM	D-F	<4023>	214	3	.	.	1	94,8597
	P	395	<2317>	7	.	.	.	85,2152
	CL	19	14	<296>	.	.	.	89,9696
	FW	2	.	.	<5>	.	.	71,4285
	D	3	.	.	.	<.>	.	0,0000
	N	1	.	.	.	.	<.>	0,0000
GL	D-F	<4144>	92	5	.	.	.	97,7128
	P	189	<2528>	2	.	.	.	92,9753
	CL	26	9	<294>	.	.	.	89,3617
	FW	2	.	.	<5>	.	.	71,4285
	D	3	.	.	.	<.>	.	0,0000
	N	1	.	.	.	.	<.>	0,0000

Tabela 47: Matrizes de confusão de *a* no córpus normal com os modelos VLMM original e GL (linha: referência; coluna: teste).

A Tabela 48 mostra os resultados obtidos sobre o córpus segmentado. Novamente, a taxa de acerto do modelo GL é maior que a do modelo VLMM, mas com uma redução do erro um pouco menor do que a obtida no córpus normal: 6,8715%. Mesmo assim a redução do erro sobre *a* continua ao redor de 50%. A taxa de acerto de *que* se altera pouco, e a de palavras desconhecidas é novamente bem maior usando GL.

A Tabela 49 mostra as matrizes de confusão de *que*. Novamente, pouca coisa é diferente do que com o córpus normal. A taxa de acerto do modelo GL é um pouco mais alta em todas etiquetas, exceto WD e as que são erros de anotação.

As matrizes de confusão das etiquetas de *a* são mostradas na Tabela 50. Em relação aos resultados sobre o córpus normal, as observações são semelhantes. A melhora na taxa de acerto obtida pelo modelo GL ocorre devido a uma melhor classificação entre D-F e P, havendo uma pequena diminuição na taxa de acerto de CL em relação ao modelo VLMM.

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	*84,5250*	*15,4750*	*1712*	*11063*
	85,1125	14,8875	1647	11063
a	*94,5576*	*5,4424*	*740*	*13597*
	97,1979	2,8021	381	13597
Desconhecidas	*74,9214*	*25,0786*	*1995*	*7955*
	84,6637	15,3363	1220	7955
Conhecidas	*97,2642*	*2,7358*	*7348*	*268586*
	97,2147	2,7853	7481	268586
Total	*96,6215*	*3,3785*	*9343*	*276541*
	96,8536	3,1464	8701	276541

Tabela 48: Resultados do modelo GL comparados aos do VLMM (em itálico) no córpus segmentado.

MODELO		WPRO	C	WD	CONJ	FW	D	WD-P	% PREC.
VLMM	WPRO	<5562>	575	14	8	.	.	.	90,3069
	C	850	<3712>	6	26	.	.	.	80,8010
	WD	13	20	<68>	4	.	.	.	64,7619
	CONJ	101	92	.	<5>	.	.	.	2,5253
	FW	.	.	.	.	<4>	.	.	100,0000
	D	.	2	.	.	.	<.>	.	0,0000
	WD-P	.	.	1	.	.	.	<.>	0,0000
GL	WPRO	<5607>	541	8	3	.	.	.	91,0375
	C	842	<3745>	4	3	.	.	.	81,5193
	WD	32	30	<43>	.	.	.	.	40,9523
	CONJ	84	92	1	<21>	.	.	.	10,6060
	FW	1	3	.	.	<.>	.	.	0,0000
	D	.	2	.	.	.	<.>	.	0,0000
	WD-P	.	.	1	.	.	.	<.>	0,0000

Tabela 49: Matrizes de confusão de *que* no córpus segmentado com os modelos VLMM e GL (linha: referência; coluna: teste).

8.3 PROBLEMAS E CONCLUSÕES

O resultado do modelo GL confirma a necessidade de usar o contexto direito para classificar *a* corretamente. A ambiguidade de *a* entre determinante e preposição é grandemente reduzida. Ao mesmo tempo, o uso de contexto direito exerce pequena influência na classificação de *que*, o que continua confirmando que o tipo de informação necessária é diferente de contextos locais de qualquer lado.

O modelo GL apresenta alta taxa de acerto em palavras desconhecidas e em *a*, enquanto o modelo VLMM possui melhor taxa de acerto nas palavras conhecidas. Isso parece indicar que os modelos são altamente complementares. O trabalho futuro é combinar os dois e verificar se contextos de tamanho variável para ambos os lados, com uma ordem de inferência também variável, incrementam a taxa de acerto tanto do modelo VLMM quanto do modelo GL.

MODELO		D-F	P	CL	FW	D	D-P	N	% PREC.
VLMM	D-F	<8056>	206	8	.	.	.	.	97,4123
	P	433	<4411>	17	.	.	.	.	90,7426
	CL	29	39	<386>	.	.	.	.	85,0220
	FW	2	1	.	<4>	.	.	.	57,1428
	D	3	.	.	.	<.>	.	.	0,0000
	D-P	1	.	.	.	.	<.>	.	0,0000
	N	1	.	.	.	.	.	<.>	0,0000
GL	D-F	<8170>	94	6	.	.	.	.	98,7908
	P	191	<4665>	5	.	.	.	.	95,9679
	CL	38	42	<374>	.	.	.	.	82,3788
	FW	1	.	.	<6>	.	.	.	85,7142
	D	3	.	.	.	<.>	.	.	0,0000
	D-P	1	.	.	.	.	<.>	.	0,0000
	N	.	.	.	.	.	.	<1>	100,0000

Tabela 50: Matrizes de confusão de *a* no córpus segmentado com os modelos VLMM original e GL (linha: referência; coluna: teste).

9 OUTRAS IMPLEMENTAÇÕES

Além das abordagens desenvolvidas acima, realizamos diversos outros testes de implementação, configuração e execução no etiquetador VLMM. Alguns apenas de caráter informativo. Nas seções seguintes apresentamos estes testes.

9.1 SEM AMBIGUIDADE EM QUE E A

Para testarmos os efeitos dos erros de *que* e *a*, criamos dois novos conjuntos de treinamento e teste baseados no córpus normal: um em que *que* não é ambíguo e outro em que *a* não é ambíguo. Para isso, no córpus sem *que* ambíguo substituímos todas ocorrências de etiquetas de *que* pela etiqueta `QUE`, e no córpus sem *a* ambíguo substituímos todas suas etiquetas pela etiqueta `A`. Depois executamos o etiquetador original sobre esses córpus, e comparamos os resultados com os obtidos sobre o córpus normal.

A Tabela 51 mostra os resultados sobre o córpus sem *que* ambíguo. A taxa de acerto aumenta consideravelmente. Há 1701 erros a menos no córpus sem *que* ambíguo. Entretanto, o número de vezes em que *que* é mal classificado no córpus normal é 1683. Ou seja, há 18 palavras que são etiquetadas corretamente no córpus sem *que* ambíguo. Mas o número de erros sobre *a* aumenta em 31 na etiquetagem, o que significa que a não diferenciação de *que* prejudica uma pequena parcela (4,5%) da etiquetagem de *a*. Analisando os erros cometidos no córpus sem *que* ambíguo contra os erros no córpus normal, constatamos que, sem contar as ocorrências de *que*, há 384 palavras etiquetadas de forma diferente. Portanto, a não especialização da classe morfossintática de *que* prejudica a etiquetagem de algumas palavras e melhora a de outras.

A Tabela 52 mostra as matrizes de confusão das etiquetas de *a* no córpus normal e no sem *que* ambíguo, executando-se o etiquetador VLMM. Há um aumento no acerto de *a* como determinante, mas uma diminuição no acerto de *a* como preposição. De modo geral, o que parece acontecer é um maior número de atribuições a *a* como determinante, em detrimento de atribuições como preposição. Assim, a distinção da classe de *que* influencia a classificação de *a* de alguma forma.

Executando o etiquetador VLMM sobre o córpus sem *a* ambíguo, obtemos os resultados da Tabela 53, comparados com os resultados sobre o córpus normal. Há 673 erros de etiquetagem a menos sobre o córpus sem *a* ambíguo. Sendo o número de ocorrências de *a* que não são mais errados igual a 659, há 14 classificações corretas a mais usando-se o córpus sem *a* ambíguo. Entretanto, há 8 erros a mais sobre *que*. Há também um pequeno aumento na taxa de acerto de palavras desconhecidas.

Os erros sobre *que* apresentam uma tendência semelhante à que acontece sobre *a* usando o córpus sem *que* ambíguo, conforme pode ser visto nas matrizes de confusão da Tabela 54. Desta vez, o etiquetador tende a atribuir menor número de etiquetas `WPRO` e mais vezes a etiqueta `C`.

Finalmente, criamos um córpus de treinamento e de teste nos quais nem *que* e nem *a* são ambíguos. O primeiro recebe sempre a etiqueta `QUE`, e o segundo a etiqueta `A`. A Tabela 55 mostra os resultados obtidos pelo modelo VLMM em comparação com

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	*84,7775* 100,0000	*15,2225* 0,0000	*1683* 0	*11056* 11056
a	*90,9726* 90,5479	*9,0274* 9,4521	*659* 690	*7300* 7300
Desconhecidas	*74,6631* 74,5845	*25,3369* 25,4155	*2256* 2263	*8904* 8904
Conhecidas	*97,1787* 97,8589	*2,8213* 2,1411	*7084* 5376	*251087* 251087
Ambíguas	*95,4343* 96,3936	*4,5657* 3,6064	*6208* 4500	*135969* 124778
Total	*96,4076* 97,0618	*3,5924* 2,9382	*9340* 7639	*259991* 259991

Tabela 51: Resultados do modelo VLMM original no córpus normal (em itálico) e no córpus sem *que* ambíguo.

CÓRPUS		D-F	P	CL	FW	D	N	% PREC.
Normal	D-F	<4023>	214	3	.	.	1	94,8597
	P	395	<2317>	7	.	.	.	85,2152
	CL	19	14	<296>	.	.	.	89,9696
	FW	2	.	.	<5>	.	.	71,4285
	D	3	.	.	.	<.>	.	0,0000
	N	1	.	.	.	.	<.>	0,0000
Sem *que*	D-F	<4037>	200	3	.	.	1	95,1898
	P	440	<2272>	7	.	.	.	83,5601
	CL	19	14	<296>	.	.	.	89,9696
	FW	2	.	.	<5>	.	.	71,4286
	D	3	.	.	.	<.>	.	0,0000
	N	1	.	.	.	.	<.>	0,0000

Tabela 52: Matrizes de confusão de *a* com o modelo VLMM original no córpus normal e no córpus sem *que* ambíguo (linha: referência; coluna: teste).

os obtidos sobre o córpus normal. A diferença na taxa de acerto de palavras desconhecidas é menor aqui do que usando os outros dois córpus mostrados acima. Mas o número de erros é menor do que a soma dos erros de *que* e de *a* no córpus normal, indicando que 81 mais palavras são etiquetadas corretamente quando não há ambiguidade nem com *que* e nem com *a*.

9.2 CONTEXTO ESQUERDO ESPECÍFICO PARA *que*

Experimentamos gerar um novo modelo criando como novo parâmetro uma árvore de contexto esquerdo apenas para *que*. Ou seja, sempre que *que* for visto, seu contexto esquerdo de etiquetas é adicionado a essa nova árvore, além da árvore de contexto normal. Na inferência, quando uma palavra *que* estiver sendo etiquetada,

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	*84,7775* 84,7051	*15,2225* 15,2949	*1683* 1691	*11056* 11056
a	*90,9726* 100,0000	*9,0274* 0,0000	*659* 0	*7300* 7300
Desconhecidas	*74,6631* 74,8428	*25,3369* 25,1572	*2256* 2240	*8904* 8904
Conhecidas	*97,1787* 97,4403	*2,8213* 2,5597	*7084* 6427	*251087* 251087
Ambíguas	*95,4343* 95,6734	*4,5657* 4,3266	*6208* 5551	*135969* 128299
Total	*96,4076* 96,6664	*3,5924* 3,3336	*9340* 8667	*259991* 259991

Tabela 53: Resultados do modelo VLMM original no córpus normal (em itálico) e no córpus sem *a* ambíguo.

CÓRPUS		WPRO	C	WD	CONJ	FW	D	WD-P	% PREC.
Normal	WPRO	<5569>	561	13	9	.	.	.	90,5234
	C	833	<3724>	6	31	.	.	.	81,0623
	WD	14	19	<68>	4	.	.	.	64,7619
	CONJ	99	91	.	<8>	.	.	.	4,0404
	FW	.	.	.	.	<4>	.	.	100,0000
	D	.	2	.	.	.	<.>	.	0,0000
	WD-P	.	.	1	.	.	.	<.>	0,0000
Sem *a*	WPRO	<5532>	598	13	9	.	.	.	89,9220
	C	805	<3752>	5	32	.	.	.	81,6717
	WD	13	19	<69>	4	.	.	.	65,7143
	CONJ	98	92	.	<8>	.	.	.	4,0404
	FW	.	.	.	.	<4>	.	.	100,0000
	D	.	2	.	.	.	<.>	.	0,0000
	WD-P	.	.	1	.	.	.	<.>	0,0000

Tabela 54: Matrizes de confusão de *que* com o modelo VLMM original no córpus normal e no córpus sem *a* ambíguo (linha: referência; coluna: teste).

a probabilidade da etiqueta sendo analisada é obtida dessa árvore, dado o contexto esquerdo. Essa probabilidade é então interpolada com as outras probabilidades relevantes. Chamaremos o modelo resultante de VLMM+LC-QUE (LC de *Left Context*, contexto esquerdo).

A Tabela 56 mostra os resultados obtidos utilizando o modelo VLMM+LC-QUE sobre o córpus normal. Como pode ser visto, a taxa de acerto sobre *que* aumenta, havendo uma redução no erro de 3,15%. Indiretamente, a taxa de acerto sobre *a* piora levemente. O erro total é diminuído em apenas 0,40%.

A Tabela 57 mostra as matrizes de confusão de *que* obtidas pelos modelos original e com contexto esquerdo extra para *que*. O efeito desse contexto extra parece se

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	*84,7775* 100,0000	*15,2225* 0,0000	*1683* 0	*11056* 11056
a	*90,9726* 100,0000	*9,0274* 0,0000	*659* 0	*7300* 7300
Desconhecidas	*74,6631* 74,6855	*25,3369* 25,3145	*2256* 2254	*8904* 8904
Conhecidas	*97,1787* 98,1429	*2,8213* 1,8571	*7084* 4663	*251087* 251087
Total	*96,4076* 97,3395	*3,5924* 2,6605	*9340* 6917	*259991* 259991

Tabela 55: Resultados do modelo VLMM original no córpus normal (em itálico) e no córpus sem *que* nem *a* ambíguos.

PALAVRAS	% ACERTO	% ERRO	ERROS	OCORRÊNCIAS
que	*84,7775* 85,2569	*15,2225* 14,7431	*1683* 1630	*11056* 11056
a	*90,9726* 90,8630	*9,0274* 9,1370	*659* 667	*7300* 7300
Desconhecidas	*74,6631* 74,6294	*25,3369* 25,3706	*2256* 2259	*8904* 8904
Conhecidas	*97,1787* 97,1946	*2,8213* 2,8054	*7084* 7044	*251087* 251087
Total	*96,4076* 96,4218	*3,5924* 3,5782	*9340* 9303	*259991* 259991

Tabela 56: Resultados do modelo VLMM original (em itálico) e do modelo VLMM+LC-QUE no córpus normal.

resumir a uma maior tendência do modelo em classificar *que* como `WPRO`. A taxa de acerto sobre `WPRO` sobe, enquanto desce para todas as outras etiquetas.

9.3 MODELOS HÍBRIDOS

Experimentamos juntar modelos com efeitos positivos que parecem se complementar. Testamos diferentes combinações de modelos, algumas no córpus normal e outras no segmentado:

VLMM+RIGHT-A+LC-QUE Combinação do modelo com contexto esquerdo extra para *que* com o modelo que usa contexto direito simples para *a*;

VLMM+SYNTAX-QUE+RIGHT-A Combinação do modelo usando estrutura sintática para *que* com o modelo de contexto direito simples para *a*;

MODELO		WPRO	C	WD	CONJ	FW	D	WD-P	% PREC.
VLMM	WPRO	<5569>	561	13	9	.	.	.	90,5234
	C	833	<3724>	6	31	.	.	.	81,0623
	WD	14	19	<68>	4	.	.	.	64,7619
	CONJ	99	91	.	<8>	.	.	.	4,0404
	FW	.	.	.	.	<4>	.	.	100,0000
	D	.	2	.	.	.	<.>	.	0,0000
	WD-P	.	.	1	.	.	.	<.>	0,0000
VLMM+LC-QUE	WPRO	<5682>	466	2	2	.	.	.	92,3602
	C	892	<3694>	.	8	.	.	.	80,4092
	WD	25	38	<41>	1	.	.	.	39,0476
	CONJ	93	98	1	<6>	.	.	.	3,0303
	FW	1	.	.	.	<3>	.	.	75,0000
	D	.	2	.	.	.	<.>	.	0,0000
	WD-P	.	.	1	.	.	.	<.>	0,0000

Tabela 57: Matrizes de confusão de *que* com o modelo VLMM original e o modelo VLMM+LC-QUE no córpus normal (linha: referência; coluna: teste).

VLMM+SYNTAX-QUE+LC-QUE Combinação do modelo que usa estrutura sintática com o modelo com contexto extra, ambos para *que*;

VLMM+SYNTAX-QUE+RIGHT-A+LC-QUE Combinação dos três modelos: dois específicos para *que* e um para *a*.

A Tabela 58 mostra os resultados obtidos sobre o córpus normal. Como não utilizamos o modelo com estrutura sintática sobre o córpus normal, o único modelo híbrido testado é o VLMM+RIGHT-A+LC-QUE, que junta um modelo para *que* com um para *a*. Os modelos realmente se complementam, com o modelo combinado obtendo um resultado geral melhor que os dois modelos isoladamente, e melhor que o modelo original. O modelo combinado praticamente mantém a taxa de acerto do modelo VLMM+LC-QUE sobre *que* e a do modelo VLMM+RIGHT-A sobre *a*.

A Tabela 59 mostra os resultados obtidos sobre o córpus segmentado pelos modelos listados acima. Dessa vez o modelo usando estrutura sintática pôde ser combinado com os de contexto extra e contexto direito simples. Os dois modelos com contextos específicos para *que*, VLMM+SYNTAX-QUE e VLMM+LC-QUE, quando combinados, obtém taxa de acerto sobre *que* melhor do que quando isolados. A melhor taxa de acerto sobre *a* é obtida usando apenas o modelo com contexto direito simples não combinado com outros modelos. Entretanto, a melhor taxa de acerto geral é obtida quando um dos modelos para *que* é combinado com o modelo específico para *a*, com uma diferença pequena entre o uso de contexto extra, de contexto com estrutura sintática e de ambos.

PALAVRAS	MODELOS	% ACERTO	ERROS
que	VLMM	84,7775	1683
	VLMM+RIGHT-A	84,8408	1676
	VLMM+LC-QUE	85,2569	1630
	VLMM+RIGHT-A+LC-QUE	85,2931	1626
a	VLMM	90,9726	659
	VLMM+RIGHT-A	92,2603	565
	VLMM+LC-QUE	90,8630	667
	VLMM+RIGHT-A+LC-QUE	92,1370	574
Desconhecidas	VLMM	74,6631	2256
	VLMM+RIGHT-A	74,6743	2255
	VLMM+LC-QUE	74,6294	2259
	VLMM+RIGHT-A+LC-QUE	74,6294	2259
Conhecidas	VLMM	97,1787	7084
	VLMM+RIGHT-A	97,2145	6994
	VLMM+LC-QUE	97,1946	7044
	VLMM+RIGHT-A+LC-QUE	97,2288	6958
Total	VLMM	96,4076	9340
	VLMM+RIGHT-A	96,4426	9249
	VLMM+LC-QUE	96,4218	9303
	VLMM+RIGHT-A+LC-QUE	96,4549	9217

Tabela 58: Comparação de modelos híbridos sobre o córpus normal.

PALAVRAS	MODELOS	% ACERTO	ERROS
que	VLMM	84,5250	1712
	VLMM+SYNTAX-QUE	84,8956	1671
	VLMM+RIGHT-A	84,5792	1706
	VLMM+LC-QUE	85,0312	1656
	VLMM+RIGHT-A+LC-QUE	85,0493	1654
	VLMM+SYNTAX-QUE+RIGHT-A	84,9498	1665
	VLMM+SYNTAX-QUE+LC-QUE	85,1306	1645
	VLMM+SYNTAX-QUE+RIGHT-A+LC-QUE	85,1577	1642
a	VLMM	94,5576	740
	VLMM+SYNTAX-QUE	94,5429	742
	VLMM+RIGHT-A	94,9327	689
	VLMM+LC-QUE	94,4841	750
	VLMM+RIGHT-A+LC-QUE	94,8592	699
	VLMM+SYNTAX-QUE+RIGHT-A	94,9106	692
	VLMM+SYNTAX-QUE+LC-QUE	94,4841	750
	VLMM+SYNTAX-QUE+RIGHT-A+LC-QUE	94,8592	699
Desconhecidas	VLMM	74,9214	1995
	VLMM+SYNTAX-QUE	74,8712	1999
	VLMM+RIGHT-A	74,9466	1993
	VLMM+LC-QUE	74,8586	2000
	VLMM+RIGHT-A+LC-QUE	74,8837	1998
	VLMM+SYNTAX-QUE+RIGHT-A	74,8963	1997
	VLMM+SYNTAX-QUE+LC-QUE	74,8334	2002
	VLMM+SYNTAX-QUE+RIGHT-A+LC-QUE	74,8586	2000
Conhecidas	VLMM	97,2642	7348
	VLMM+SYNTAX-QUE	97,2798	7306
	VLMM+RIGHT-A	97,2832	7297
	VLMM+LC-QUE	97,2776	7312
	VLMM+RIGHT-A+LC-QUE	97,2951	7265
	VLMM+SYNTAX-QUE+RIGHT-A	97,2984	7256
	VLMM+SYNTAX-QUE+LC-QUE	97,2813	7302
	VLMM+SYNTAX-QUE+RIGHT-A+LC-QUE	97,2992	7254
Total	VLMM	96,6215	9343
	VLMM+SYNTAX-QUE	96,6352	9305
	VLMM+RIGHT-A	96,6406	9290
	VLMM+LC-QUE	96,6327	9312
	VLMM+RIGHT-A+LC-QUE	96,6504	9263
	VLMM+SYNTAX-QUE+RIGHT-A	96,6540	9253
	VLMM+SYNTAX-QUE+LC-QUE	96,6356	9304
	VLMM+SYNTAX-QUE+RIGHT-A+LC-QUE	96,6537	9254

Tabela 59: Comparação dos modelos híbridos sobre o córpus segmentado.

Parte III

COMPARAÇÕES E CONCLUSÕES

10 COMPARAÇÕES

Agrupamos os resultados dos modelos que mostraram melhora em algum aspecto e os mostramos na Tabela 60. Estes resultados são os obtidos sobre o córpus normal, portanto os modelos VLMM+SYNTAX-QUE e VLMM+IOB não aparecem. Utilizamos a coluna da taxa de acerto e geramos um gráfico de barras para auxiliar a visualização, conforme pode ser visto na Figura 10. Fica claro que os aspectos com maior variação entre modelos são a etiquetagem de *a* e a de palavras desconhecidas. O modelo GL atinge a maior taxa de acerto em ambos, e o modelo VLMM+PERCEPTRON é pior em relação a palavras desconhecidas. Os demais aspectos diferem pouco entre modelos, e é importante notar que isso ocorre com *que*. Mesmo o modelo GL obtendo boa diferença na taxa de acerto de *a*, o modelo que melhor classifica *que* é o VLMM+PERCEPTRON, mesmo que com pequena margem em relação aos demais modelos.

PALAVRAS	MODELOS	% ACERTO	% ERRO	ERROS	OCORR.
que	VLMM	*84,7775*	*15,2225*	*1683*	*11056*
	VLMM+PERCEPTRON	85,2207	14,7793	1634	11056
	VLMM+RIGHT-A	84,8408	15,1592	1676	11056
	GL	84,8951	15,1049	1670	11056
a	VLMM	*90,9726*	*9,0274*	*659*	*7300*
	VLMM+PERCEPTRON	92,4109	7,5890	554	7300
	VLMM+RIGHT-A	92,2603	7,7397	565	7300
	GL	95,4932	4,5068	329	7300
Desconhecidas	VLMM	*74,6631*	*25,3369*	*2256*	*8904*
	VLMM+PERCEPTRON	67,2619	32,7381	2915	8904
	VLMM+RIGHT-A	74,6743	25,3257	2255	8904
	GL	84,5238	15,4762	1378	8904
Conhecidas	VLMM	*97,1787*	*2,8213*	*7084*	*251087*
	VLMM+PERCEPTRON	97,1229	2,8771	7224	251087
	VLMM+RIGHT-A	97,2145	2,7855	6994	251087
	GL	97,1038	2,8962	7272	251087
Total	VLMM	*96,4076*	*3,5924*	*9340*	*259991*
	VLMM+PERCEPTRON	96,1002	3,8998	10139	259991
	VLMM+RIGHT-A	96,4426	3,5574	9249	259991
	GL	96,6730	3,3270	8650	259991

Tabela 60: Comparação dos melhores modelos sobre o córpus normal.

Os resultados dos modelos sobre o córpus segmentado são mostrados na Tabela 61, desta vez incluindo VLMM+SYNTAX-QUE em VLMM+IOB. Também geramos um gráfico de barras da taxa de acerto dos modelos, o qual mostramos na Figura 11. Novamente, o modelo GL obtém taxa de acerto significativamente maior do que os outros modelos na etiquetagem de *a* e de palavras desconhecidas. O modelo VLMM+PERCEPTRON

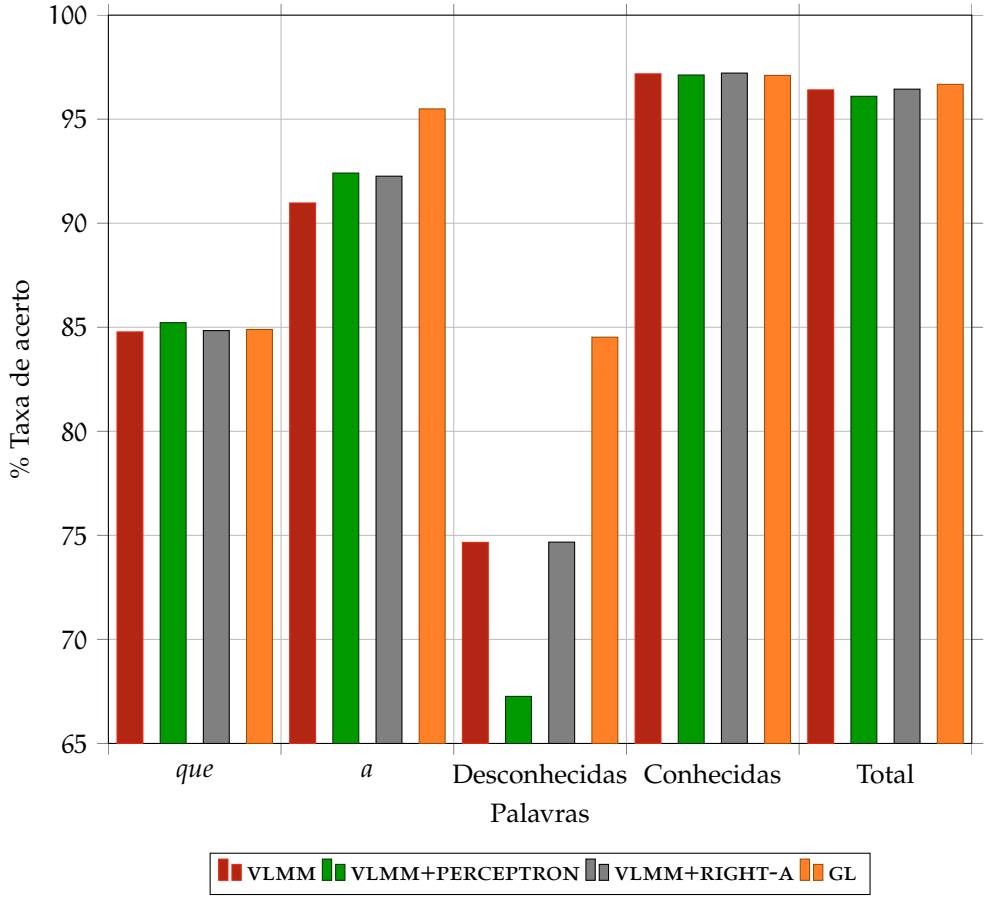


Figura 10: Taxa de acerto obtida pelos melhores modelos sobre o córpus normal.

apresenta uma grande dificuldade na classificação de palavras desconhecidas, mas por outro lado obtém a melhor taxa de acerto, mesmo que pequena relativamente, sobre palavras conhecidas. Além disso obtém a segunda maior taxa de acerto sobre *que*, o qual desta vez o modelo GL acerta mais vezes. Isso pode significar que um conjunto de etiquetas menor auxilia o modelo GL, ao diminuir a esparsidade de contextos possíveis.

A dificuldade em se classificar a palavra *que* corretamente surge já na fase de anotação manual, constituindo em uma tarefa complexa mesmo para especialistas em linguística, segundo o depoimento pessoal de linguistas encarregados dessa tarefa. Isso implica em dificuldades de se manter uniformidade ao longo de diversas pessoas trabalhando. Além disso, há ocorrências de *que* que provavelmente estão mal classificadas no córpus anotado, como indicam as atribuições errôneas de `D` e `WD-P` a *que*, vistas anteriormente nas matrizes de confusão de *que*. Seria interessante obter esse limitante superior. Uma idéia para trabalho futuro é listar os erros de *que* e recorrer a um linguista para analisá-los individualmente, verificando se são erros do etiquetador ou se são realmente erros de anotação. E além disso, verificar "a consistência e relevância da distinção entre *que*/`C` e *que*/`WPRO` do ponto de visto morfológico, isto é, enquanto classe de palavra"[1]. O mesmo poderia ser feito para *a*.

1 Agradeço à professora Maria Clara Paixão de Sousa pela discussão levantada quando da defesa desta tese.

PALAVRAS	MODELOS	% ACERTO	% ERRO	NÚM. ERROS	OCORR.
que	VLMM	*84,5250*	*15,4750*	*1712*	*11063*
	VLMM+SYNTAX-QUE	84,8956	15,1044	1671	11063
	VLMM+IOB	83,0607	16,9393	1874	11063
	VLMM+PERCEPTRON	85,0583	14,9417	1653	11063
	VLMM+RIGHT-A	84,5792	15,4208	1706	11063
	GL	85,1125	14,8875	1647	11063
a	VLMM	*94,5576*	*5,4424*	*740*	*13597*
	VLMM+SYNTAX-QUE	94,5429	5,4571	742	13597
	VLMM+IOB	95,7417	4,2583	579	13597
	VLMM+PERCEPTRON	95,3519	4,6481	632	13597
	VLMM+RIGHT-A	94,9327	5,0673	689	13597
	GL	97,1979	2,8021	381	13597
Desconhecidas	VLMM	*74,9214*	*25,0786*	*1995*	*7955*
	VLMM+SYNTAX-QUE	74,8712	25,1288	1999	7955
	VLMM+IOB	74,5569	25,4431	2024	7955
	VLMM+PERCEPTRON	66,5996	33,4004	2657	7955
	VLMM+RIGHT-A	74,9466	25,0534	1993	7955
	GL	84,6637	15,3363	1220	7955
Conhecidas	VLMM	*97,2642*	*2,7358*	*7348*	*268586*
	VLMM+SYNTAX-QUE	97,2798	2,7202	7306	268586
	VLMM+IOB	97,2214	2,7786	7463	268586
	VLMM+PERCEPTRON	97,2039	2,7961	7510	268586
	VLMM+RIGHT-A	97,2832	2,7168	7297	268586
	GL	97,2147	2,7853	7481	268586
Total	VLMM	*96,6215*	*3,3785*	*9343*	*276541*
	VLMM+SYNTAX-QUE	96,6352	3,3648	9305	276541
	VLMM+IOB	96,5694	3,4306	9487	276541
	VLMM+PERCEPTRON	96,3235	3,6765	10167	276541
	VLMM+RIGHT-A	96,6406	3,3594	9290	276541
	GL	96,8536	3,1464	8701	276541

Tabela 61: Comparação dos melhores modelos sobre o córpus segmentado.

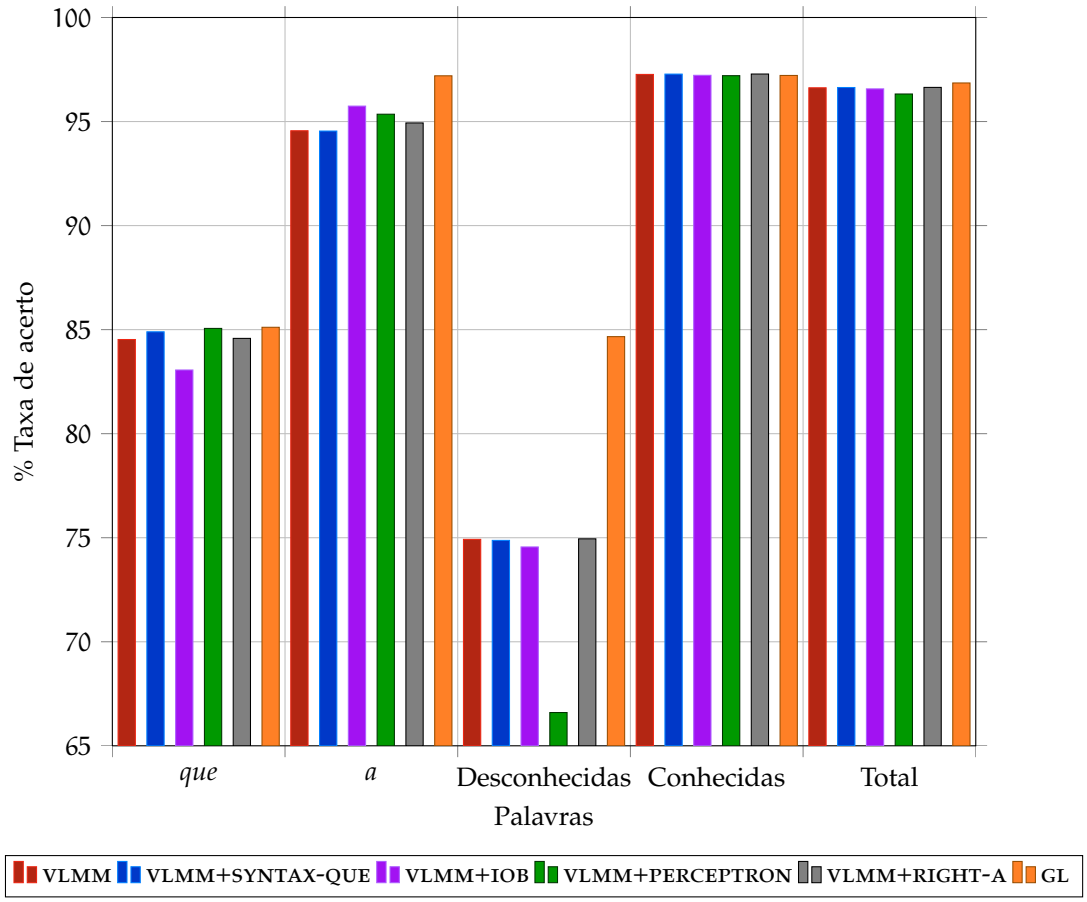


Figura 11: Taxa de acerto obtida pelos melhores modelos sobre o córpus segmentado.

11 CONCLUSÕES

Duas palavras muito comuns no português, e entre as cinco mais frequentes no córpus Tycho Brahe, *que* e *a* impõem grande dificuldade na sua correta classificação morfossintática. Um quarto dos erros cometidos pelo etiquetador VLMM são sobre elas.

Experimentamos formas de diminuir a ambiguidade imposta por elas, através de modelos que consideram mais contexto ou usam mais informação além de apenas o contexto local esquerdo. Os resultados foram variados, mas permitiram que chegássemos a conclusões sobre as duas idéias postas como objetivo: as de que as palavras *que* e *a* possuem ambiguidades diferentes e portanto dependem de informações contextuais diferentes.

Primeiro, a compactação do contexto esquerdo acabou generalizando-o demais, e não se mostrou vantajoso na classificação de *que*. A utilização direta de informação sintática obteve um resultado melhor sobre *que*, mas não significativamente. O treinamento com perceptron foi a abordagem que gerou a melhor taxa de acerto sobre *que*, mas com uma redução no erro de apenas 2,91%. Até mesmo o etiquetador usando aprendizado guiado, que obteve a melhor taxa de acerto geral, não foi capaz de melhorar a taxa de acerto sobre *que* mais do que o modelo com perceptron. Além disso, nos experimentos usando contexto direito para etiquetar *que*, o desempenho piorou. Portanto, podemos constatar que:

- *que* não depende de contexto direito; e
- não sabemos se *que* depende de contexto não local ou não, ao menos não de forma generalista demais.

Mesmo que a abordagem em que tentamos generalizar contextos de *que* não tenha funcionado, isto não exclui a possibilidade de que, realmente, *que* dependa de contexto não local esquerdo. Trabalhos futuros devem investigar outras abordagens possíveis. Além disso, é preciso uma investigação sobre erros de anotação no córpus Tycho Brahe, e, mais importante ainda, uma verificação sobre a conceituação das etiquetas de *que*, bem como o comportamento e influência delas nas tarefas seguintes de PLN.

A segunda idéia descrita no começo do trabalho é a de que a palavra *a* dependeria do contexto direito. Mesmo usando uma abordagem simples, mostramos que, realmente, o contexto direito tira ambiguidade de *a*. Uma abordagem mais elaborada, implementada no etiquetador usando aprendizado guiado, mostrou-se melhor ainda na classificação de *a*, diminuindo a taxa de erro em aproximadamente 50%. No futuro resta investigar se a maior taxa de acerto sobre palavras desconhecidas alcançada pelo modelo GL é a principal responsável pela melhor classificação de *a*, se a melhora na taxa de acerto de *a* é devida tão somente ao uso elaborado do contexto direito, ou o quanto as duas coisas colaboram para isso.

As principais contribuições deste trabalho são, portanto, as verificações empíricas de que *que* e *a* possuem efetivamente ambiguidades diferentes, e que a distinção entre as etiquetas de *que* – considerando que quase todas as abordagens testadas falharam – talvez seja apenas sintática, e não morfossintática. Dada essa última observação, e o nosso teste de que a utilização de apenas uma etiqueta para *que* praticamente não

prejudica a análise morfossintática das demais palavras, tem-se com esta tese uma base empírica para um novo exame da consistência e relevância do inventário de etiquetas definido para o córpus Tycho Brahe. Uma discussão do que pode ser feito para isso é apresentada mais abaixo. Por último, uma contribuição adicional deste trabalho é a implementação de um analisador sintático não-supervisionado para o português, que até aqui é o único de que tivemos conhecimento.

O etiquetador VLMM pode ser encontrado em http://www.ime.usp.br/~kepler. O código fonte foi escrito em C++, e inclui a implementação de todos os modelos citados nesse trabalho, com exceção do etiquetador GL. O analisador sintático CCM-P também será disponibilizado no endereço acima. Quanto aos recursos, o projeto do córpus Tycho Brahe exige que o usuário se registre para poder obter acesso aos textos anotados. Devido a isso, não disponibilizamos os córpus de treinamento e teste diretamente na internet. Entretanto, eles podem ser requisitados por email[1].

11.1 TRABALHOS FUTUROS

Uma implementação do etiquetador com aprendizado guiado permitirá misturá-lo com o modelo VLMM. A idéia parece promissora, já que a implementação atual do modelo GL usa apenas trigramas para os contextos esquerdo e direito. Implementar o modelo GL em C++, ao invés de implementar o modelo VLMM em Java, permitirá continuarmos obtendo a execução rápida da implementação em C++ do modelo VLMM, e também permitirá compararmos a diferença de desempenho entre esse novo modelo e os vários modelos VLMM.

Para tentar diminuir a dificuldade em etiquetar *que*, uma idéia usando árvores para contextos não locais é, após termos a junção dos modelos VLMM e GL, fazer uma hipótese do Aprendizado Guiado considerar também intervalos aceitos não adjacentes. Isso envolveria alterar a primeira ação de etiquetagem do algoritmo, que inicia um novo intervalo quando a palavra não possui contextos, para que considere contextos não locais, ou seja, use a informação de intervalos já aceitos mas não adjacentes. Essa ainda é uma idéia abstrata, portanto precisa ser melhor investigada quando for possível ter um modelo VLMM+GL.

Outra investigação a ser feita é a análise de erros cometidos por etiquetadores no português moderno e em outros idiomas românticos como o espanhol, para verificar se *que* e *a* continuam apresentando o mesmo grau de ambiguidade ou, no caso do espanhol, se há palavras similares que mostram dificuldades semelhantes. Pode-se, além disso, verificar se há palavras com ambiguidade equivalente no inglês. Essas investigações também envolvem testar outros etiquetadores disponíveis, em especial baseados em regras ou em máxima entropia, usando os conjuntos de treinamento e teste que utilizamos, para verificar se eles cometem os mesmos erros sobre *que* e *a*.

Há, ainda, três importantes questões a serem testadas sobre *que*. A primeira é comparar a diferença de erros entre a etiqueta C e as demais etiquetas de *que*, para verificar se a distinção entre apenas duas classes para *que* influencia outras palavras e acarreta uma melhora ou piora no resultado geral da etiquetagem. O segundo teste é verificar se os erros sobre as outras palavras quando *que* não é ambíguo são os mesmos que os erros usando o conjunto todo de etiquetas de *que*, dado que numericamente já mostramos que não há diferença significativa. A terceira questão é verificar se essas ou alguma outra modificação no inventário de etiquetas de *que* influencia de alguma forma a análise sintática subsequente. Caso não influencie

1 Veja em http://www.ime.usp.br/~kepler ou http://www.ime.usp.br/~mfinger.

negativamente, haverá mais um indicativo de que o conjunto de etiquetas de *que* poderia ser simplificado, com ganhos na análise morfossintática.

Uma possível combinação dos vários modelos apresentados é formar um comitê de votação simples. O resultado mostrará quão complementares os modelos são, indicando se os erros sobre *que* e *a* são distintos para cada modelo individual ou se há erros persistentes. Outros métodos de combinação poderão ser testados, inclusive com a inserção de outros etiquetadores disponíveis.

Uma idéia de uma ferramenta a ser desenvolvida seria criar uma interface para análise de erros, que gravasse as correções feitas. Isso poderia realçar diferenças de contexto de *que* com cada etiqueta possível – também para *a* e suas etiquetas. Ou então, como parte da correção de uma etiqueta, o linguista apontaria quais palavras ou etiquetas na sentença o mostraram qual era a etiqueta correta. Isso poderia reunir pistas quanto a alguma forma de generalizar ou pular contextos de *que* – e confirmar ou não a dependência de contexto direito de *a*. Um possível começo é buscar implementar isto junto à ferramenta E-Dictor (Paixão de Sousa, Kepler e Faria, 2009, 2010).

Um passo necessário é utilizar os novos textos disponíveis no córpus Tycho Brahe e, junto com isso, desenvolver uma forma reprodutível de se criar os córpus de treinamento, de teste e de desenvolvimento.

Parte IV

APÊNDICES

A TEXTOS SELECIONADOS

Lista em ordem cronológica dos textos anotados morfossintaticamente selecionados do córpus Tycho Brahe (IEL-UNICAMP e IME-USP, 2010) que foram utilizados para compor o córpus de treinamento e teste usados na tese.

(1517-1584) FRANCISCO DE HOLANDA – *Da Pintura Antiga* (Número de palavras: 55691)

(1542-1606) DIOGO DO COUTO – *Décadas* (Número de palavras: 53916)

(1556-1632) LUIS DE SOUSA – *A vida de Frei Bertolameu dos Mártires* (Número de palavras: 59158)

(1579-1621) FRANCISCO RODRIGUES LOBO – *Côrte na Aldeia e Noites de Inverno* (Número de palavras: 59686)

(1601-1667) MANUEL DA COSTA – *A Arte de Furtar* (Número de palavras: 58400)

(1608-1697) ANTONIO VIEIRA – *Cartas* (Número de palavras: 57823)
– *Sermões* (Número de palavras: 62387)

(1608-1666) FRANCISCO MANUEL DE MELO – *Cartas Familiares* (Número de palavras: 58118)

(1631-1682) ANTONIO DAS CHAGAS – *Cartas Espirituais* (Número de palavras: 57429)

(1644-1710) MANUEL BERNARDES – *Nova Floresta* (Número de palavras: 59116)

(1651-1735) JOSÉ DA CUNHA BROCHADO – *Cartas* (Número de palavras: 34743)

(1675-1754) ANDRÉ DE BARROS – *A Vida do Padre António Vieira* (Número de palavras: 50207)

(1695-?) ALEXANDRE DE GUSMÃO – *Cartas* (Número de palavras: 31235)

(1702-1783) CAVALEIRO DE OLIVEIRA – *Cartas* (Número de palavras: 53672)

(1705-1763) MATIAS AIRES – *Reflexão sobre a Vaidade dos Homens e Carta sobre a Fortuna* (Número de palavras: 66722)

(1713-1792) LUIZ ANTONIO VERNEY – *Verdadeiro Método de Estudar* (Número de palavras: 54952)

(1724-1772) CORREIA GARCAO – *Obras Completas* (Número de palavras: 26845)

(1750-1839) MARQUESA D'ALORNA – *Cartas e outros Escritos* (Número de palavras: 49881)

(1799-1854) ALMEIDA GARRETT – *Viagens na minha terra* (Número de palavras: 51474)

(1836-1915) RAMALHO ORTIGAO – *Cartas a Emília* (Número de palavras: 34138)

Total de palavras: 1.035.593.

B SISTEMA DE ANOTAÇÃO

A Tabela 62 lista todas as etiquetas que ocorrem no nosso córpus de treinamento tirado do Tycho Brahe. Para uma descrição sobre o uso de cada uma delas, inclusive das não listadas aqui, consulte http://www.tycho.iel.unicamp.br/~tycho/corpus/manual/tags.html.

ETIQUETAS				
(	,	.	ADJ	ADJ-F
ADJ-F-P	ADJ-G	ADJ-G-P	ADJ-P	ADJ-R
ADJ-R-F	ADJ-R-F-P	ADJ-R-G	ADJ-R-G-P	ADJ-R-P
ADJ-S	ADJ-S-F	ADJ-S-F-P	ADJ-S-P	ADV
ADV+CL	ADV-NEG	ADV-R	ADV-S	C
CL	CL+CL	CONJ	CONJ-NEG	CONJS
D	D+OUTRO	D+OUTRO-F	D+OUTRO-F-P	D+OUTRO-P
D-F	D-F-P	D-G	D-G-P	D-P
D-UM	D-UM-F	D-UM-F-P	D-UM-P	DEM
ET	ET+CL	ET-D	ET-D+CL	ET-D+SE
ET-F	ET-G	ET-I	ET-P	ET-P+CL
ET-P+SE	ET-PP	ET-R	ET-RA	ET-SD
ET-SP	ET-SR	FP	FW	HV
HV+CL	HV+SE	HV-AN-F	HV-AN-F-P	HV-AN-P
HV-D	HV-F	HV-F+CL	HV-F+SE	HV-G
HV-G+CL	HV-G+SE	HV-I	HV-P	HV-P+CL
HV-P+P	HV-P+P+CL	HV-P+P+SE	HV-P+SE	HV-P+SE+CL
HV-PP	HV-R	HV-RA	HV-RA+SE+CL	HV-SD
HV-SP	HV-SP+SE	HV-SR	INTJ	N
N-P	NEG	NPR	NPR-P	NUM
NUM-F	OUTRO	OUTRO-F	OUTRO-F-P	OUTRO-P
P	P+ADV	P+CL	P+D	P+D-F
P+D-F-P	P+D-P	P+D-UM	P+D-UM-F	P+D-UM-F-P
P+D-UM-P	P+DEM	P+NPR	P+OUTRO	P+OUTRO-F
P+OUTRO-F-P	P+OUTRO-P	P+PRO	P+Q	P+Q-F
P+Q-F-P	P+Q-P	P+WADV	P+WD	P+WPRO
PRO	PRO$	PRO$-F	PRO$-F-P	PRO$-P
Q	Q-F	Q-F-P	Q-G	Q-G-P
Q-NEG	Q-NEG-F	Q-NEG-F-P	Q-NEG-P	Q-P
QT	SE	SENAO	SR	SR+CL
SR-D	SR-D+CL	SR-F	SR-F+CL	SR-G
SR-G+CL	SR-I+CL	SR-P	SR-P+CL	SR-PP
SR-R	SR-R!CL	SR-RA	SR-RA+CL	SR-SD
SR-SP	SR-SP+CL	SR-SR	TR	TR+CL
TR+SE	TR+SE+CL	TR-AN	TR-AN-P	TR-D
TR-D+CL	TR-D+SE	TR-F	TR-F+CL	TR-G
TR-G+CL	TR-G+SE	TR-I	TR-I+CL	TR-P
TR-P+CL	TR-P+SE	TR-PP	TR-R	TR-R!CL
TR-RA	TR-SD	TR-SP	TR-SP+CL	TR-SR
VB	VB+CL	VB+CL+CL	VB+P	VB+SE
VB+SE+CL	VB-AN	VB-AN-F	VB-AN-F-P	VB-AN-P
VB-D	VB-D+CL	VB-D+CL+CL	VB-D+SE	VB-D+SE+CL
VB-F	VB-F+CL	VB-F+CL+CL	VB-F+SE	VB-G
VB-G+CL	VB-G+CL+CL	VB-G+SE	VB-G+SE+CL	VB-I
VB-I+CL	VB-I+CL+CL	VB-P	VB-P+CL	VB-P+CL+CL
VB-P+P	VB-P+SE	VB-P+SE+CL	VB-PP	VB-R
VB-R!CL	VB-R!SE	VB-R+CL	VB-RA	VB-RA+CL
VB-RA+SE	VB-SD	VB-SD+CL	VB-SP	VB-SP+CL
VB-SP+CL+CL	VB-SP+SE	VB-SP+SE+CL	VB-SR	VB-SR+CL
VB-SR+SE	W	WADV	WD	WD-F
WD-F-P	WD-P	WPRO	WPRO$	WPRO$-F
WPRO$-F-P	WPRO$-P	WPRO-F	WPRO-F-P	WPRO-P
WQ	XX			

Tabela 62: Lista de etiquetas que ocorrem no córpus de treinamento e no de teste.

BIBLIOGRAFIA

Aires, Rachel Virgínia Xavier (out. de 2000). "Implementação, Adaptação, Combinação e Avaliação de Etiquetadores para o Português do Brasil". Diss. de mestrado. Universidade de São Paulo - Campus São Carlos: Instituto de Ciências Matemáticas e Computação. URL: http://www.nilc.icmc.usp.br/nilc/download/dissertacao2000rachel.zip. (Ver pp. 5, 20).

Allen, James (1995). *Natural Language Understanding*. Benjamin/Cummings Publishing. (Ver p. 33).

Alves, Carlos Daniel Chacur e Marcelo Finger (set. de 1999). "Etiquetagem do Português Clássico Baseada em Córpora". In: *Proceedings of IV Encontro para o Processamento Computacional da Língua Portuguesa Escrita e Falada (PROPOR99)*. Évora, Portugal, pp. 21–22. (Ver p. 5).

Baker, James K. (1979). "Trainable grammars for speech recognition". In: *The Journal of the Acoustical Society of America* 65.S1, S132–S132. DOI: 10.1121/1.2017061. URL: http://link.aip.org/link/?JAS/65/S132/1. (Ver p. 38).

Bick, Eckhard (2000). "The Parsing System "Palavras": Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework". PhD. Denmark: Aarhus University. (Ver p. 36).

Bick, Eckhard, Diana Santos, Susana Afonso e Rachel Marchi (2007). "Floresta Sintá(c)tica: Ficção ou realidade?" In: *Diana Santos (ed.), Avaliação conjunta: um novo paradigma no processamento computacional da língua portuguesa*, pp. 291–300. URL: http://www.linguateca.pt/Floresta. (Ver p. 34).

Bonfante, Andréia Gentil (jun. de 2003). "Parsing Probabilístico para o Português do Brasil". Tese de doutorado. Universidade de São Paulo: Programa de Pós-Graduação em Ciência da Computação, Instituto de Ciências Matemáticas e Computação, p. 153. URL: http://www.nilc.icmc.usp.br/nilc/download/TeseFinalBonfante.zip. (Ver p. 36).

Brants, Thorsten (2000). "TnT – A Statistical Part-of-Speech Tagger". In: *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP-2000)*. Seattle, WA. URL: http://citeseer.ist.psu.edu/brants00tnt.html. (Ver p. 6).

Brill, Eric (1995). "Unsupervised Learning of Disambiguation Rules for Part of Speech Tagging". In: *Proceedings of the Third Workshop on Very Large Corpora*. Ed. por David Yarovsky e Kenneth Church. Somerset, New Jersey: Association for Computational Linguistics, pp. 1–13. URL: http://citeseer.ist.psu.edu/brill95unsupervised.html. (Ver p. 6).

Brito, Rogério Theodoro (set. de 2003). "Alinhamento de Sequências Biológicas". Diss. de mestrado. Universidade de São Paulo: Programa de Pós-Graduação em Ciência da Computação, Instituto de Matemática e Estatística. URL: http://www.ime.usp.br/dcc/posgrad/teses/rogerio.pdf. (Ver p. 11).

Britto, Helena e Marcelo Finger (jun. de 1999). "Constructing a Parsed Corpus of Historical Portuguese". In: *ACH/ALLC-99 International Humanities Computing Conference*. University of Virginia. Charlottesville, Virginia. URL: http://www2.iath.virginia.edu/ach-allc.99/proceedings/britto.html. (Ver p. 10).

Britto, Helena, Charlotte Galves, Ilza Ribeiro, Marina Augusto e Ana Paula Scher (1999). "Morphological Annotation System for Automatic Tagging of Eletronic Textual Corpora: From English to Romance Languages". In: *Anais do VI Simposio*

Internacional de Comunicación Social. Santiago de Cuba: Editora Oriente, pp. 582–589. (Ver p. 10).

Bühlmann, Peter e Abraham J. Wyner (1999). "Variable Length Markov Chains". In: *Annals of Statistics* 27.2, pp. 480–513. (Ver p. 14).

Carvalho e Sousa, Fabiano (out. de 2003). "Analisador Sintático Estatístico Orientado ao Núcleo-Léxico para a Língua Portuguesa". Diss. de mestrado. Universidade de São Paulo: Programa de Pós-Graduação em Ciência da Computação, Instituto de Matemática e Estatística. (Ver p. 36).

Charniak, Eugene (1993). *Statistical Language Learning*. The MIT Press. (Ver p. 6).

Charniak, Eugene e Mark Johnson (jun. de 2005). "Coarse-to-fine n-best parsing and MaxEnt discriminative reranking". In: *Proceedings of the 43rd Annual Meeting of the ACL*. Association for Computational Linguistics, pp. 173–180. URL: http://acl.ldc.upenn.edu/P/P05/P05-1022.pdf. (Ver p. 35).

Cinlar, Erhan (1975). *Introduction to Stochastic Processes*. New Jersey, US: Prentice-Hall. (Ver p. 11).

Clark, Alexander (2001a). "Unsupervised induction of stochastic context-free grammars using distributional clustering". In: *ConLL '01: Proceedings of the 2001 workshop on Computational Natural Language Learning*. Toulouse, France: Association for Computational Linguistics, pp. 1–8. (Ver p. 35).

– (2001b). "Unsupervised Language Acquisition: Theory and Practice". Tese de doutorado. University of Sussex. URL: http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0212024. (Ver pp. 35, 36).

Collins, Michael (1999). "Head-Driven Statistical Models for Natural Language Parsing". Tese de doutorado. University of Pennsylvania: Department of Computer e Information Science. URL: http://people.csail.mit.edu/u/m/mcollins/public_html/. (Ver p. 7).

– (2002). "Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms". In: *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*. Morristown, NJ, USA: Association for Computational Linguistics, pp. 1–8. DOI: http://dx.doi.org/10.3115/1118693.1118694. URL: http://portal.acm.org/ft_gateway.cfm?id=1118694&type=external&coll=GUIDE&dl=GUIDE&CFID=75108837&CFTOKEN=22963155. (Ver pp. 55–57, 69, 74).

IEL-UNICAMP e IME-USP (2010). *Córpus Histórico do Português Anotado Tycho Brahe*. IEL-UNICAMP e IME-USP. URL: http://www.tycho.iel.unicamp.br/. (Ver pp. 10, 31, 42, 99).

Cutting, Doug, Julian Kupiec, Jan Pedersen e Penelope Sibun (1991). *A Practical Part-of-Speech Tagger*. Rel. téc. Xerox Palo Alto Research Center. (Ver p. 6).

Dempster, Arthur P., Nan M. Laird e Donald B. Rubin (1977). "Maximum Likelihood from Incomplete Data Via the EM Algorithm". In: *Journal of the Royal Statistical Society*. B 39, pp. 1–38. (Ver p. 34).

Dredze, Mark e Joel Wallenberg (2008). "Icelandic data driven part of speech tagging". In: *HLT '08: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies*. Columbus, Ohio: Association for Computational Linguistics, pp. 33–36. URL: http://portal.acm.org/ft_gateway.cfm?id=1557700&type=pdf&coll=Portal&dl=GUIDE&CFID=82067137&CFTOKEN=58144922. (Ver p. 75).

Earley, Jay (1970). "An efficient context-free parsing algorithm". In: *Communications of the ACM* 13.2, pp. 94–102. ISSN: 0001-0782. DOI: http://doi.acm.org/10.1145/362007.362035. URL: http://portal.acm.org/ft_gateway.cfm?id=362035&type=pdf&coll=GUIDE&dl=GUIDE&CFID=85021399&CFTOKEN=17532965. (Ver p. 34).

Finger, Marcelo (nov. de 2000). "Técnicas de Otimização da Precisão Empregadas no Etiquetador Tycho Brahe". In: *PROPOR - 5th International Workshop on Computational Processing of Written and Spoken Portuguese*. Atibaia, Brazil: Springer-Verlag Berlin Heidelberg, pp. 19–22. URL: http://www.ime.usp.br/~tycho/participants/finger/propor2000.pdf. (Ver p. 20).

Freund, Yoav e Robert E. Schapire (1999). "Large margin classification using the perceptron algorithm". In: *Machine learning* 37.3, pp. 277–296. URL: http://www.cs.ucsd.edu/~yfreund/papers/LargeMarginsUsingPerceptron.pdf. (Ver p. 55).

Galves, Charlotte e Helena Britto (1999). "A Construção do Corpus Anotado do Português Europeu: o sistema de anotação". In: *Anais do IV Encontro para o Processamento Computacional da Língua Portuguesa Escrita e Falada (PROPOR 99)*. Évora, pp. 81–92. (Ver p. 10).

Gesmundo, Andrea (2009). "Bidirectional Sequence Classification for Part of speech Tagging". In: *Proceedings of EVALITA*. (Ver p. 74).

Jurafsky, Daniel S. e James H. Martin (2000). *Speech and Language Processing*. Prentice Hall. (Ver p. 11).

Kepler, Fábio Natanael (abr. de 2005). "Um Etiquetador Morfo-Sintático Baseado em Cadeias de Markov de Tamanho Variável". Diss. de mestrado. Universidade de São Paulo: Programa de Pós-Graduação em Ciência da Computação, Instituto de Matemática e Estatística. URL: http://www.ime.usp.br/~kepler/. (Ver pp. 14, 16, 18, 23).

Klein, Dan (2005). "The Unsupervised Learning of Natural Language Structure". Tese de doutorado. Stanford University. URL: http://www.cs.berkeley.edu/~klein/. (Ver pp. 34, 35, 38, 40).

Klein, Dan e Christopher D. Manning (2001). "Distributional phrase structure induction". In: *ConLL '01: Proceedings of the 2001 workshop on Computational Natural Language Learning*. Toulouse, France: Association for Computational Linguistics, pp. 1–8. (Ver p. 34).

– (jul. de 2002). "A Generative Constituent-Context Model for Improved Grammar Induction". In: *ACL '02: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania: Association for Computational Linguistics, pp. 128–135. (Ver pp. 34–36, 42).

– (2004). "Corpus-Based Induction of Syntactic Structure: Models of Dependency and Constituency". In: *Proceedings of the 42nd Annual Meeting of the ACL*. (Ver pp. 34, 36, 43).

Krauth, Werner e Marc Mézard (1987). "Learning algorithms with optimal stability in neural networks". In: *Journal of Physics A: Mathematical and General* 20, pp. 745–752. (Ver p. 74).

Lafferty, John, Andrew Mccallum e Fernando C. N. Pereira (2001). "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: *Proc. 18th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, pp. 282–289. URL: http://citeseer.ist.psu.edu/lafferty01conditional.html. (Ver p. 55).

Linguateca (2010). *Linguateca, centro de recursos distribuído para o processamento computacional da língua portuguesa*. URL: http://www.linguateca.pt (acesso em 13 de jul. de 2010). (Ver p. 34).

Mächler, Martin e Peter Bühlmann (mar. de 2002). *Variable Length Markov Chains: Methodology, Computing and Software*. Rel. de pesquisa 104. Seminar fur Statistik. CH-8091 Zürich, Switzerland: Eidgenossische Technische Hochschule (ETH). (Ver p. 14).

Manning, Christopher D., Prabhakar Raghavan e Hinrich Schütze (2008). *Introduction to information retrieval*. 1ª ed. New York: Cambridge University Press. ISBN:

0521865719. URL: http://nlp.stanford.edu/IR-book/information-retrieval-book.html. (Ver p. 36).

Manning, Christopher D. e Hinrich Schütze (1999). *Foundations Of Statistical Natural Language Processing*. Cambridge, MA, USA: MIT Press. ISBN: 0-262-13360-1. (Ver pp. 5, 6, 9, 31, 33, 37, 49).

Marcus, Mitchell P., Beatrice Santorini e Mary Ann Marcinkiewicz (1994). "Building a Large Annotated Corpus of English: the Penn Treebank". In: *Computational Linguistics* 19.2, pp. 313–330. URL: http://citeseer.ist.psu.edu/marcus04building.html. (Ver pp. 9, 31, 35, 42).

Martins, Ronaldo, Graça Nunes e Ricardo Hasegawa (jun. de 2003). "Curupira: A Functional Parser for Brazilian Portuguese". In: *PROPOR - 6th International Workshop on Computational Processing of Written and Spoken Portuguese*. Ed. por Nuno J. Mamede, Jorge Baptista, Isabel Trancoso e Maria Graças Volpe Nunes. Vol. 2721. Lecture Notes in Computer Science. Faro, Portugal: Springer-Verlag Berlin Heidelberg, pp. 179–183. ISBN: 3-540-40436-8. URL: http://springerlink.metapress.com/openurl.asp?genre=article{\&}issn=0302-9743{\&}volume=2721{\&}spage=179. (Ver p. 36).

McClosky, David, Eugene Charniak e Mark Johnson (jun. de 2006). "Effective Self-Training for Parsing". In: *Proceedings of HLT/NAACL*. Association for Computational Linguistics, pp. 152–159. URL: http://www.aclweb.org/anthology/N/N06/N06-1020.pdf. (Ver p. 35).

NILC (2010). *NILC – Núcleo Interinstitucional de Linguística Computacional*. ICMC-USP. URL: http://www.nilc.icmc.usp.br. (Ver p. 34).

Paixão de Sousa, Maria Clara, Fábio Natanael Kepler e Pablo Picasso Feliciano Faria (nov. de 2009). "E-Dictor: Novas perspectivas na codificação e edição de corpora de textos históricos". In: *Linguística de Corpus: Sínteses e Avanços. Anais do VIII Encontro de Linguística de Corpus*. To be published. Shepherd, T., Berber Sardinha, T. e Veirano Pinto, M. UERJ, Rio de Janeiro, RJ, Brasil. (Ver p. 95).

– (2010). *E-Dictor*. Software. URL: http://purl.org/edictor (acesso em 2010). (Ver p. 95).

Pereira, Fernando C. N. e Yves Schabes (1992). "Inside-Outside Reestimation from Partially Bracketed Corpora". In: *Meeting of the Association for Computational Linguistics*, pp. 128–135. URL: citeseer.ist.psu.edu/pereira92insideoutside.html. (Ver p. 43).

Petrov, Slav e Dan Klein (abr. de 2007). "Improved Inference for Unlexicalized Parsing". In: *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. Rochester, New York: Association for Computational Linguistics, pp. 404–411. URL: http://www.aclweb.org/anthology/N/N07/N07-1051. (Ver p. 35).

Ratnaparkhi, Adwait (mai. de 1996). "A Maximum Entropy Model for Part-Of-Speech Tagging". In: *Conference on Empirical Methods in Natural Language Processing*. University of Pennsylvania. URL: http://acl.ldc.upenn.edu/W/W96/W96-0213.pdf. (Ver p. 6).

Rissanen, Jorma (1983). "A Universal Data Compression System". In: *IEEE Trans. Inform. Theory* IT-29, pp. 656 –664. (Ver p. 16).

Rosenblatt, Frank (nov. de 1958). "The perceptron: A probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6, pp. 386–408. ISSN: 1939-1471. URL: http://psycnet.apa.org/doi/10.1037/h0042519. (Ver p. 55).

Ross, Sheldon M. (1970). *Applied Probability with Optimization Applications*. Holden-Day, Inc. (Ver p. 11).

– (1987). *Introduction to Probability and Statistics for Engineers and Scientists*. John Wiley & Sons, Inc. ISBN: 047181752X. (Ver p. 11).

Samuelsson, Christer e Atro Voutilainen (1997). "Comparing a Linguistic and a Stochastic Tagger". In: *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*. Ed. por Philip R. Cohen e Wolfgang Wahlster. Somerset, New Jersey: Association for Computational Linguistics, pp. 246–253. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.15.87. (Ver p. 6).

Santos, Cícero Nogueira, Ruy L. Milidiú e Raúl P. Rentería (2008). "Portuguese Part-of-Speech Tagging Using Entropy Guided Transformation Learning". In: *PROPOR - 8th International Workshop on Computational Processing of Written and Spoken Portuguese*. Ed. por A. Teixeira al. Vol. 5190. Lecture Notes in Artificial Intelligence. Vitória, ES, Brazil: Springer-Verlag Berlin Heidelberg, pp. 143–152. (Ver p. 20).

Schütze, Hinrich (1995). "Distributional Part-of-Speech Tagging". In: *EACL* 7, pp. 141–148. (Ver pp. 36, 40).

Shen, Libin, Giorgio Satta e Aravind Joshi (jun. de 2007). "Guided Learning for Bidirectional Sequence Classification". In: *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Prague, Czech Republic: Association for Computational Linguistics, pp. 760–767. URL: http://www.aclweb.org/anthology/P07-1096. (Ver pp. 69, 71, 74, 75).

Smith, Noah A. e Jason Eisner (2004). "Annealing Techniques for Unsupervised Statistical Language Learning". In: *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. Barcelona, Spain: Association for Computational Linguistics, p. 486. DOI: http://dx.doi.org/10.3115/1218955.1219017. (Ver pp. 36, 43).

– (2005a). "Contrastive estimation: training log-linear models on unlabeled data". In: *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Ann Arbor, Michigan: Association for Computational Linguistics, pp. 354–362. DOI: http://dx.doi.org/10.3115/1219840.1219884. (Ver p. 43).

– (2005b). "Guiding Unsupervised Grammar Induction Using Contrastive Estimation". In: *Proceedings of IJCAI Workshop on Grammatical Inference Applications*, pp. 73–82. (Ver p. 43).

Tsuruoka, Yoshimasa e Jun'ichi Tsujii (2005). "Bidirectional inference with the easiest-first strategy for tagging sequence data". In: *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Vancouver, British Columbia, Canada: Association for Computational Linguistics, pp. 467–474. DOI: 10.3115/1220575.1220634. URL: http://portal.acm.org/ft_gateway.cfm?id=1220634&type=external&coll=GUIDE&dl=&CFID=74892741&CFTOKEN=24140270. (Ver p. 69).

Vapnik, Vladimir Naumovich e Samuel Kotz (2006). *Estimation of dependences based on empirical data: Empirical inference science: afterword of 2006*. Reprint of 1982 ed. with afterword of 2006. Information science and statistics. New York, N.Y.: Springer. ISBN: 0387308652. URL: http://www.loc.gov/catdir/enhancements/fy0663/2005938355-d.html. (Ver p. 55).

Viterbi, Andrew James (abr. de 1967). "Error Bounds For Convolutional Codes And An Asymptotically Optimal Deconding Algorithm". In: *IEEE Transactions on Information Theory*, pp. 260 –269. (Ver pp. 8, 18, 56).