Carlos Eduardo Dantas de Menezes

Um método para a construção de analisadores morfológicos, aplicado à língua portuguesa, baseado em autômatos adaptativos

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia Elétrica

São Paulo

2000

Carlos Eduardo Dantas de Menezes
Engenheiro de Eletricidade, Escola Politécnica da USP, 1994

Um método para a construção de analisadores morfológicos, aplicado à língua portuguesa, baseado em autômatos adaptativos

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia Elétrica

Área de Concentração:
Sistemas Digitais

Orientador:
Prof. Dr. João José Neto

São Paulo
2000

Aos meus pais e avó e à minha querida Nelma.

Agradecimentos

Devo agradecimentos especiais ao meu orientador por toda sua paciência, ajuda e por todos os seus conselhos oportunos; agradeço também a toda a equipe que tem trabalhado na construção do Corpus Tycho Brahe, com a qual tenho interagido.

Aos muitos amigos do Laboratório de Sistemas Integráveis (LSI) da Universidade de São Paulo e da Universidade São Judas Tadeu, que me apoiaram de diversas maneiras.

À minha querida família, que sempre me incentivou a prosseguir em frente.

Resumo

Este trabalho tem por objetivo propor um método de construção de um analisador morfológico, treinável com o uso de corpus anotado, que é independente de língua, mas que foi aqui testado com textos da língua portuguesa.

Trata-se de um sistema de aprendizado automático, que infere informações lingüísticas, relativas a aspectos lexicais e contextuais de todo um corpus de treinamento. Estas informações são armazenadas, codificadas com base em autômatos adaptativos, e posteriormente utilizadas para a tarefa de classificação ou etiquetação morfológica.

Os autômatos adaptativos mostraram-se adequados tanto para o fluxo de controle da heurística de aprendizado, como também para nele codificar todos os dados necessários.

Abstract

The present work is intended to propose a new method for the construction of a morphological tagger for natural languages. This tagger is, in a large extent, language-independent and trainable with an annotated corpus, and has been tested, in this work, with texts in Portuguese language.

This project is based on automatic learning principles, and has been develop for automatically acquiring and inferring linguistic knowledge related to lexical and contextual aspects of a training corpus.

Collected and inferred information are coded into an structure based on adaptive automata. This device is used later as a basis for tagging of other texts.

In this project, adaptive automata had proven their adequacy for both the representation and the acquisition of knowledge on the mentioned aspects of the natural language and the logic of the heuristics employed to collect that information.

Sumário

Lista de Figuras

1. Introdução

1.1 Motivação

Este trabalho tem como motivação principal a construção de ferramentas de apoio a trabalhos de pesquisa na área do Processamento de Linguagens Naturais (PLN), especialmente no que tange à língua portuguesa.

Há muitas aplicações úteis desta área. Por exemplo, logo no início da Guerra Fria o governo americano começou a patrocinar o projeto de um tradutor automático russo-inglês, com o objetivo de permitir que seus cientistas conseguissem, ao menos, ter uma noção do que os russos desenvolviam. A **tradução automática, com o fim de coletar informações,** foi assim a primeira aplicação prática na área do PLN [VASCONCELLOS-93].

Este tipo de tradução funcionou a contento, devido ao fato de não se exigir como resultado textos de qualidade. Quem interpretava a saída do tradutor era um especialista na área a ser investigada.

A classe científica, contudo, sonhava com a possibilidade de traduzir textos com alta qualidade; seria possível, então, a **tradução com a finalidade de disseminar informação**. Mas cada vez mais o sonho parecia que nunca seria realizado. Percebeu-se que os problemas eram maiores do que se esperava e, em 1966, esta área foi duramente criticada no relatório ALPAC da Academia Nacional de Ciências (dos EUA). Isto resultou no corte dos patrocínios do governo americano, por um longo período de cerca de quinze anos. Só os centros de pesquisa da Europa e do Japão persistiram neste objetivo durante este período [SLOCUM-85].

Nos nossos dias, os pesquisadores que trabalham com PLN ainda buscam obter um tradutor cujas saídas apresentem alta qualidade e pouca necessidade de revisão humana. Entretanto, devido a todas as experiências de quatro décadas, seus projetos são menos ambiciosos, em sua grande maioria, e, portanto, mais realistas quanto às qualidades de seus produtos.

Recentemente, outras aplicações de PLN, diferentes da tradução automática, mostram-se muito úteis, como é o caso da **recuperação de informações** e das **interfaces em linguagem natural** para sistemas computacionais.

Com o crescimento assombroso da quantidade de dados disponíveis na Internet, especialmente na WWW (World Wide Web), durante a década de 1990, ficou muito difícil achar nela uma determinada informação desejada. Com o advento dos chamados *sites* de busca (entre outros: Yahoo, Alta Vista, Lycos, etc.) após a segunda metade da mesma década, este quadro sofreu algumas mudanças para melhor, tornando muito mais fácil fazer uma pesquisa sobre um determinado assunto. No entanto, muita informação não pertinente, como resultado de uma busca, ainda é localizada e apresentada ao consulente, devendo então ser separada manualmente pelo usuário.

Uma evolução das atuais ferramentas de busca na WWW acontecerá quando a tecnologia de entendimento automático de textos em linguagem natural tornar-se suficientemente robusta, de modo que não mais seja necessário tentar **recuperar informação** somente através de palavras-chave, mas pelo sentido.

Uma proposta muito interessante e útil como aplicação de PLN, surgida nos últimos anos, é criar um novo tipo de **interface homem-máquina, cujo meio de interagir seja a linguagem natural** [SUERETH-97]. Seria possível digitar em qualquer microcomputador "Mande por mim um *e-mail* para o Mário, cancelando a reunião de hoje." e a interface interpretaria o pedido, construiria uma mensagem com o conteúdo solicitado, e dispararia o aplicativo de e-mails com os devidos parâmetros.

Em adição, tem-se observado recentemente pesquisas que almejam incorporar em todas estas aplicações o tratamento da linguagem falada (tradutores que trabalhem a partir da voz humana, interfaces que permitam a interação com o computador e com sistemas de banco de dados através de pedidos falados, etc.).

Em vista do apresentado, pode-se constatar a vastidão de aplicações práticas do PLN. Contudo, para que tais ferramentas possam ser construídas, alguns módulos

básicos, integrantes da maioria dos sistemas de PLN, devem ser estudados, desenvolvidos e implementados. Entre outros, pode-se citar o analisador morfológico, o analisador sintático e o analisador semântico ([BIEWER-85], [BENNETT-85], [VAUQUOIS-85], [ISABELLE-85]).

O analisador morfológico tem a tarefa de atribuir a cada palavra da língua uma etiqueta, ou seja, uma classificação morfológica (por exemplo, substantivo, artigo, adjetivo, verbo, advérbio, etc.). Ao analisador sintático cabe identificar qual é o papel de uma palavra em uma sentença (por exemplo, sujeito, objeto, etc.), além de determinar a estrutura sintática da frase. Já o analisador semântico deve auxiliar na resolução de ambigüidades* em nível morfológico e sintático, determinando o significado das palavras e das orações, a fim de que a análise completa da sentença reflita melhor a informação que se pretendia passar através da sentença escrita.

Uma característica fundamental que estas ferramentas devem apresentar é a robustez. Independentemente do estilo em que uma sentença foi escrita – ou, até mesmo, se uma sentença fugir um pouco do padrão formal da língua escrita, porém sendo esta corretamente entendida por um falante da língua em questão – deveria ser corretamente analisada pelo sistema de PLN, como um todo. E isto é realmente um desafio.

O enfoque, neste trabalho, será dado ao módulo analisador morfológico, visto ser este necessário para o funcionamento de outros módulos básicos do PLN: por exemplo, um analisador sintático trabalha com base em categorias morfológicas. Pode-se afirmar que este módulo é parte integrante de praticamente qualquer sistema de PLN [DAELEMANS-96].

Na seção seguinte, alguns comentários mais detalhados serão tecidos a respeito do analisador morfológico.

1.2 Analisador morfológico

Um analisador morfológico (ou léxico) tem por função associar a cada palavra uma etiqueta, que corresponda a sua categoria morfológica.

Por exemplo, dentro de um **sistema automático de tradução**, que se utiliza do método de transferência*, encontramos o analisador morfológico integrando a primeira

* Os termos marcados com asterisco são explicados no Apêndice 1.

etapa do processamento. Um diagrama macroscópico deste tipo de sistema tradutor pode ser observado na Figura 1.

O analisador morfológico encarrega-se de, a partir das palavras na língua-fonte, fornecer palavras já associadas às respectivas etiquetas morfológicas (átomos) para o módulo seguinte, o analisador sintático. Este, com base nas palavras analisadas morfologicamente, levanta uma representação estrutural (geralmente, uma estrutura em forma de árvore) da relação de cada palavra com as outras de uma sentença. Esta estrutura, por sua vez, é mapeada, pelo módulo de transferência, em uma outra que seja equivalente na língua-alvo, sendo que as palavras pertencentes à primeira estrutura também são mapeadas nas palavras correspondentes da língua-alvo na segunda estrutura. Com base na estrutura assim obtida, o módulo gerador morfológico efetua a flexão adequada das palavras na língua-alvo. O módulo analisador semântico auxilia os módulos de análise e de transferência a decidir o correto sentido de uma palavra.

Esta forma de particionar um sistema de tradução não é nova. A arquitetura básica do sistema tradutor chamado TAUM-AVIATION [ISABELLE-85] pode ser subdividida em blocos muito similares aos da Figura 1. Também os compiladores de linguagens de programação mostram uma estrutura muito similar [JOSÉ NETO-87].

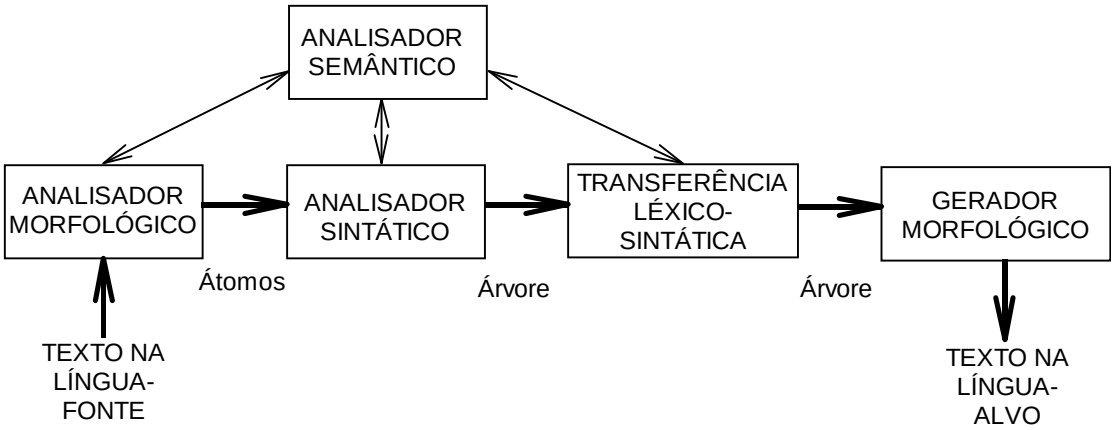


Figura 1 – Esquema macroscópico de um sistema tradutor baseado no método de transferência

Mais uma importante utilização de um analisador morfológico automático é a **criação de um grande repositório de informação lingüística**, muitas vezes chamado **corpus lingüístico**[*]. Este é constituído por uma grande quantidade de textos,

em geral com alguma forma de anotação ou marcação, como, por exemplo, anotação morfológica e anotação sintática, entre outras. Como exemplos de corpora lingüísticos podem ser citados:

- Corpus anotado Penn Treebank, do Departamento de Ciências da Computação e Informação da Universidade da Pensilvânia (EUA). É composto de cerca de 4.5 milhões de palavras em inglês americano, com anotação morfológica. Até 1993, por volta de metade do corpus continha anotação sintática. O texto é constituído por sentenças de manuais de computadores, resumos de artigos científicos de várias áreas de conhecimento, capítulos de livros de vários autores americanos, etc. [MARCUS-93]. A motivação da construção deste corpus foi a de promover uma série de pesquisas em Lingüística Computacional.
- Corpus anotado Penn-Helsinki de inglês medieval, do Departamento de Lingüística da Universidade da Pensilvânia (EUA). É um corpus de 510.000 palavras, com anotações estruturais (sintáticas), que permitem a busca no corpus não só por palavras e trechos, mas também por estruturas sintáticas. Foi construído com a motivação de facilitar o estudo e a pesquisa da variação histórica da sintaxe da língua inglesa [PPCME-98].
- Corpus anotado Tycho Brahe de português histórico, do IEL (Instituto de Estudos da Linguagem), UNICAMP, SP. Ele contém textos escritos entre os séculos XVII e XIX, permitindo a observação da evolução da língua portuguesa neste período. Até a conclusão do projeto originador deste corpus, este deverá contar com 1 milhão de palavras, com anotações morfológica e sintática. A escolha das etiquetas morfológicas tem por base o conjunto projetado para o Corpus Penn-Helsinki [TBCHP-98].

Como a quantidade de trabalho para a produção de repositórios como estes é imensa[1], a automatização de algumas das tarefas, como é o caso da anotação morfológica, torna-se muito útil.

[1] Algumas medições do tempo de etiquetação manual foram feitas pelo grupo que desenvolve o Corpus Tycho Brahe (IEL, UNICAMP, SP), e estas medidas sugerem que a velocidade de etiquetação morfológica manual é de cerca de 650 palavras/homem.hora. Ou seja, o desenvolvimento de um corpus de tamanho razoável (de 1 milhão de palavras) poderia consumir cerca de 1500 horas.homem de um

Há ainda uma realimentação deste trabalho: ferramentas automáticas são usadas para a criação de corpora anotados, enquanto estes últimos são usados para a construção automática de novas ferramentas para o PLN.

Como exemplo, em um artigo clássico da sub-área estatística do PLN [BROWN-90], os autores usam a teoria da comunicação aplicada à modelagem do processo de tradução*, com o uso de corpora, e propõem a construção de um sistema tradutor automático francês-inglês, usando este método.

Também pode ser citado o trabalho que propõe um consagrado paradigma chamado "Aprendizado Dirigido por Erros, Baseado em Transformações" [BRILL-93], com o objetivo de construir um analisador robusto de textos livres* em linguagem natural. Através de treinamento, feito com base em um corpus pequeno em comparação com o utilizado em outros métodos (o etiquetador morfológico descrito por Eric Brill foi treinado com um corpus anotado com menos de 45.000 palavras, enquanto que se relata um outro, fortemente estatístico, que necessita de um corpus de um milhão de palavras [BRILL-93, p. 33, 50, 51 e 85]), é inferido um conjunto de regras transformacionais[2] que podem ser usadas para determinar a estrutura sintática de sentenças livres. Portanto, cálculos estatísticos são feitos somente para a indução de regras; todo o restante do processamento é simbólico, ou seja, efetuado com o uso das regras construídas. A primeira aplicação dada a este paradigma foi a análise morfológica de textos em língua inglesa.

Corpora anotados são úteis também para uma série de estudos lingüísticos teóricos e outros estudos na área de PLN [MARCUS-93].

1.2.1 Dificuldades na tarefa da análise morfológica

A principal dificuldade existente na tarefa da análise morfológica encontra-se em sua susceptibilidade à ambigüidade. Um exemplo clássico da língua inglesa é a sentença:

The man can can the can (O homem pode enlatar a lata)

---

trabalho altamente especializado, isto, sem contar o tempo gasto posteriormente para a revisão do mesmo.

[2] Estas regras realizam transformações no conjunto de anotações do texto (por mudar a etiqueta morfológica de uma palavra, por exemplo), que são disparadas por determinadas condições, como por exemplo um item lexical (palavra) específico, ou algumas etiquetas de palavras adjacentes.

Nota-se que a mesma palavra, “can”, dentro de uma única sentença, manifesta-se em três categorias morfológicas diferentes, a saber:

The man <u>*can*</u>**/MODAL** <u>*can*</u>**/VERBO** *the* <u>*can*</u>**/NOME**

Pode-se perceber, no exemplo acima, que foi convencionado especificar a etiqueta morfológica logo depois da palavra em questão, precedida por uma barra (“/”). Assim, a palavra <u>*can*</u> pode ser etiquetada como **MODAL** quando representar um verbo auxiliar, modal, com o significado de poder; a palavra <u>*can*</u> pode ser etiquetada como **VERBO** quando representar um verbo com o significado “enlatar” e pode ser etiquetada como **NOME** quando representar um substantivo com o significado “lata”.

Poder-se-ia imaginar que a tarefa de analisar morfologicamente a sentença acima, de modo correto, fosse apenas uma tarefa de dicionarização. Mas não basta construir dicionários eletrônicos com todas as possíveis categorias morfológicas de todas as palavras. Isto porque, muitas vezes, é só através do contexto em que uma palavra está inserida que sua categoria morfológica pode ser determinada (isto pode ser observado no exemplo acima). E também porque não há meios práticos de dicionarizar absolutamente <u>todas</u> as palavras de uma língua, especialmente quando se considera o efeito do tempo: apesar de não possuir um vocabulário infinito, matematicamente falando, qualquer língua humana é extremamente dinâmica no tempo, possuindo um vocabulário cujo tamanho varia continuamente. Os lingüistas classificam as palavras em duas classes:

- **Classe de palavras fechada** refere-se a palavras em número bem definido e limitado, como, por exemplo, as preposições; estas são dicionarizáveis com grande facilidade.
- **Classe de palavras aberta** engloba um número ilimitado de palavras (mas finito), como, por exemplo, os substantivos (ou nomes); não é prático dicionarizar todos os nomes, pois a qualquer momento uma pessoa pode cunhar um novo termo e nomes próprios novos surgem sempre.

Então, um analisador morfológico robusto deve levar em conta informações contextuais para retirar qualquer ambigüidade da anotação morfológica de uma palavra.

1.3 Objetivos do Trabalho

Infelizmente, a língua portuguesa não tem sido alvo de pesquisas extensas no campo do processamento de linguagens naturais, em comparação com línguas como o inglês, o espanhol, o francês, o alemão e o japonês. Isto pode ser explicado por fatores tais como a maior importância econômica dos países que falam estes idiomas.

Pode-se, porém, citar alguns trabalhos recentes nas áreas de morfologia, sintaxe e tradução da língua portuguesa.

Uma tese propôs uma especificação lingüística completa de um tradutor português-inglês baseado no método de transferência e, usando um formalismo de unificação* (GPSG), construiu a gramática de análise da língua portuguesa [CHIN-96]; uma outra tese apresentou a implementação de um protótipo de tradutor inglês-português, o qual usava uma representação interna chamada "árvore de palavras" e tratava alguns fenômenos semânticos [KINOSHITA-97]; um projeto construiu um analisador morfológico e um sintático, visando a criação de um tradutor; ambos trabalham com regras escritas à mão e demonstram boa robustez [BICK-96]; e outro trabalho que, com objetivo de auxiliar a criação de um corpus, construiu um analisador morfológico parcialmente treinável para o português clássico [ALVES-99].

Como contribuição do presente trabalho, será analisado um método de construção de um analisador morfológico totalmente operacional, o qual será testado com textos da língua portuguesa.

O formalismo dos autômatos adaptativos (AA) será usado como base de implementação. Uma outra contribuição deste trabalho é a constatação da adequação dos AA para a construção dos algoritmos de aprendizado automático utilizados no desenvolvimento desta dissertação [JOSÉ NETO-94].

1.4 Plano da Dissertação

Após estas considerações iniciais, que fornecem uma idéia geral a respeito do problema que se pretende resolver, este trabalho apresenta a seguinte seqüência:

O capítulo 2 relata a fundamentação conceitual em que se baseia este trabalho: os diversos paradigmas em que se baseiam os analisadores morfológicos tradicionais e o formalismo dos autômatos adaptativos.

O capítulo 3 desenvolve a proposta desta dissertação.

O capítulo 4 descreve os experimentos realizados. Os métodos empregados, bem como as simplificações introduzidas nos experimentos, são comentadas e avaliadas. Também os aspectos mais importantes da implementação são considerados aqui, juntamente com os resultados obtidos através dos testes.

O capítulo 5 avalia todo o conjunto da dissertação: quais foram as contribuições deste trabalho para esta área de pesquisa, quais são as propostas que permitiriam melhorar as idéias apresentadas e que trabalhos se tenciona realizar futuramente.

O capítulo 6 contém as referências bibliográficas; o apêndice 1 explica o significado de alguns termos específicos do Processamento de Linguagens Naturais usados neste trabalho e o apêndice 2 contém listagens relevantes ao trabalho desenvolvido.

2. Fundamentação Conceitual

Este capítulo fornece as bases conceituais para que se possa compreender a proposta apresentada neste trabalho. Uma visão geral, a respeito do que tem sido feito no mundo em termos de análise morfológica automática, é apresentada: paradigmas que se propõem a inferir alguma forma de conhecimento de um corpus anotado, além de um paradigma baseado em regras escritas manualmente.

Uma seção é dedicada à apresentação de um formalismo computacional que servirá de base para a implementação proposta neste trabalho.

2.1 O estado da arte em analisadores morfológicos

Basicamente, serão analisados quatro paradigmas ou métodos aplicados à análise morfológica de textos em linguagem natural: o estatístico, o que se utiliza de regras escritas manualmente, o baseado em regras inferidas automaticamente e o com base em exemplos memorizados.

Será possível compará-los e perceber idéias lingüísticas comuns a vários destes métodos e paradigmas.

2.1.1 Analisadores morfológicos estatísticos

O paradigma estatístico (ou, como outros autores preferem, fortemente estatístico [BRILL-93]), utilizado para a construção de analisadores morfológicos, é baseado na análise estatística de um corpus de treinamento. A idéia central desta análise é que a ocorrência de determinada etiqueta na sentença está correlacionada

com as etiquetas das palavras vizinhas. Isto é muito bem modelado pelos chamados *n-gramas*, que introduzem a idéia aproximada de que a etiqueta de uma palavra depende apenas e tão somente das *n-1* etiquetas anteriores. Na realidade, esta dependência pode-se estender a todas as etiquetas da sentença, porém ela é cada vez mais baixa quanto mais distante estiver determinada etiqueta; desta forma, os modelos *n-gramas* procuram simplificar o modelo real, levando-se em consideração somente as etiquetas das palavras mais próximas. Esta idéia provou ser, na prática, uma simplificação bastante razoável.

Os *n-gramas* mais utilizados, por uma questão da relação custo/benefício, são os bigramas (n=2), nos quais a probabilidade de uma etiqueta é considerada dependente apenas da probabilidade da etiqueta anterior, e os trigramas, nos quais a probabilidade de uma etiqueta é considerada dependente apenas das probabilidades das duas etiquetas anteriores [CHARNIAK-93].

A implementação do modelo de n-gramas geralmente é feita através de Modelos Ocultos de Markov (ou HMM – Hidden Markov Models), que nada mais são que autômatos finitos probabilísticos , ou seja, autômatos cujas transições estão associadas a probabilidades. A Figura 2 contém um exemplo de um HMM que implementa um modelo bem simplificado do comportamento das etiquetas morfológicas de uma língua, baseado em bigramas. Cada estado representa a etiqueta da palavra anterior e as transições, com suas respectivas probabilidades associadas, as possíveis etiquetas da palavra corrente.

Uma aplicação deste paradigma para a língua portuguesa, que usa um modelo de bigramas, conseguiu atingir uma precisão de 84,5%, tendo sido treinado à base do Corpus Radiobrás, de apenas 14.000 palavras. Este valor de precisão é baixo quando comparado com o conseguido por alguns outros grupos que implementaram algoritmos baseados neste paradigma. Os autores deste trabalho citaram informações da literatura, mostrando que precisões acima de 95% foram conseguidas, usando-se textos da língua inglesa. Estes mesmos concluíram que a precisão obtida aumenta com o crescimento do corpus de treinamento. Isto justifica o baixo desempenho obtido, visto que, na época, não havia um corpus de treinamento maior [VILLAVICENCIO-95].

Um inconveniente deste paradigma é que ele é muito dependente de um corpus de treinamento grande, da ordem de um milhão de palavras, para a obtenção de uma taxa de acertos comparável ao citado acima [BRILL-93].

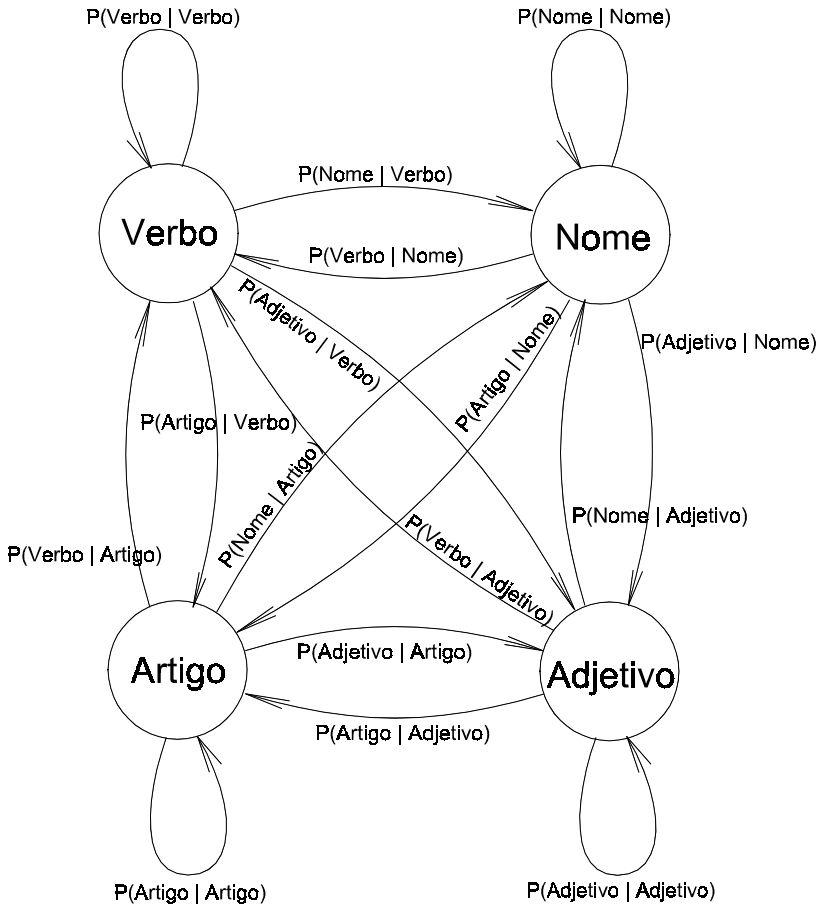


Figura 2 – Exemplo de um HMM aplicado à tarefa da análise morfológica

2.1.2 Analisadores morfológicos baseados em regras escritas manualmente

A base do chamado paradigma de dois níveis, que constitui um formalismo de regras proposto na década de 80, consiste no uso de poucas representações e relações entre estas para a modelagem computacional de alguma tarefa lingüística (análise e geração morfológica, por exemplo, como ilustrado na Figura 1 do Capítulo 1) [KOSKENNIEMI-97].

Nos sistemas baseados nas chamadas regras gerativas, que são regras de reescrita da forma x $\rightarrow$ y, onde o elemento x é reescrito ou substituído por y, várias

representações ou níveis intermediários são criados, visto que o elemento x não está mais disponível para a aplicação de outras regras. Estas regras gerativas são aplicadas uma a uma, seqüencialmente, gerando níveis intermediários que servem como entrada para outras regras. A ordem de aplicação destas regras é muito importante [ANTWORTH-91].

O autor deste paradigma propõe uma arquitetura de dois níveis apenas para a análise morfológica, conforme mostrado na Figura 3, onde a representação superficial é a forma como a palavra é escrita e usada, e a representação lexical consiste da forma canônica, dicionarizável, da palavra (que é invariante) mais as flexões morfológicas (ou as etiquetas morfológicas que representam estas flexões) [KOSKENNIEMI-97].

Devido ao fato de a arquitetura possuir apenas duas representações ou níveis, as relações entre estes são complexas. Contudo, estas relações podem ser subdivididas, encarando-se que cada regra cuidará de um fenômeno morfológico específico.

As regras não são aplicadas seqüencialmente (como é o caso das regras gerativas), gerando uma série de estados intermediários, mas conjuntamente, em paralelo. O componente denominado "regras" é implementado por um conjunto de transdutores de estado finito, e o léxico, por um outro transdutor, que armazena as formas canônicas das palavras como uma árvore de letras.

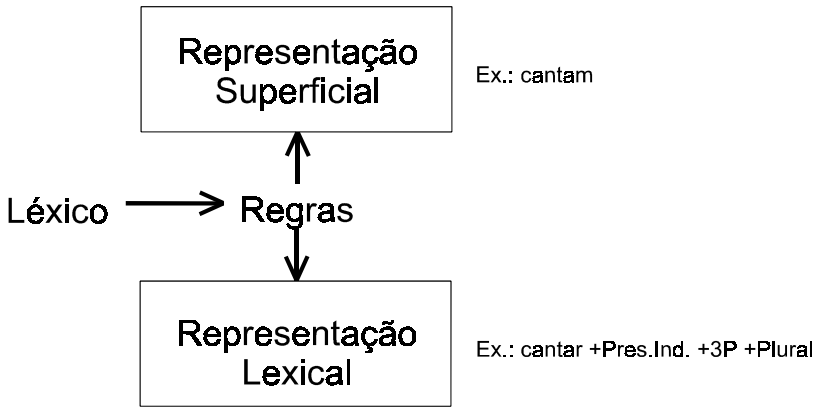


Figura 3 – Modelo de dois níveis

Deve-se ressaltar que este formalismo é completamente bidirecional, ou seja, as regras são escritas apenas uma vez e podem ser usadas para o mapeamento do nível superficial no lexical e vice-versa. Assim, pode ser usado tanto no sentido da análise

morfológica (do nível superficial para o lexical), quanto no da geração (do nível lexical para o superficial) [ANTWORTH-91].

Já foram implementados transdutores léxicos, funcionais e abrangentes, baseados neste paradigma, para pelo menos as seguintes línguas: inglês, francês, alemão, coreano e turco [KARTTUNEN-94]; também relatam-se trabalhos para o finlandês, russo, sueco, suaíli, dinamarquês, basco, estoniano e o árabe [KOSKENNIEMI-97].

Como vantagens atribuídas a este paradigma, obtém-se uma grande compactação (o transdutor para o francês pôde ser compactado em poucas centenas de Kbytes) e velocidade (o autômato e o transdutor finito são os dispositivos computacionais mais simples e eficientes possível) de processamento [KARTTUNEN-94].

Contudo, as ferramentas automáticas que são usadas para a construção destes transdutores limitam-se à compilação de regras, ao processamento de léxico e a algumas ferramentas de apoio. As regras têm de ser escritas a mão por um especialista em lingüística. Nas seções seguintes serão comentadas idéias que poderiam aprimorar este paradigma.

2.1.3 Analisadores morfológicos baseados em regras inferidas automaticamente

O paradigma já consagrado para o aprendizado automático, denominado "Aprendizado Dirigido por Erros, Baseado em Transformações", foi proposto com o objetivo de construir um analisador robusto de textos livres em linguagem natural. Através de treinamento com um corpus pequeno (se comparado com outros métodos), são inferidas regras transformacionais, as quais, posteriormente, podem ser usadas para determinar as categorias morfológicas das palavras em textos livres, bem como até mesmo a estrutura sintática de tais textos [BRILL-93].

O paradigma pressupõe duas fases distintas: a **fase de treinamento** e a **fase de uso** propriamente dito (Figura 4).

Durante a **fase de treinamento** (Figura 5), usam-se duas versões de um mesmo texto: uma não-anotada e a outra, com anotações consideradas corretas (morfológicas, sintáticas, ou quaisquer outras, dependendo da tarefa que se intenciona

dar ao etiquetador). Esta versão do texto, com anotações, está no chamado estado PADRÃO (define-se que todo texto anotado está em um estado, o qual é determinado por uma *n-upla*, o conjunto das anotações deste texto).

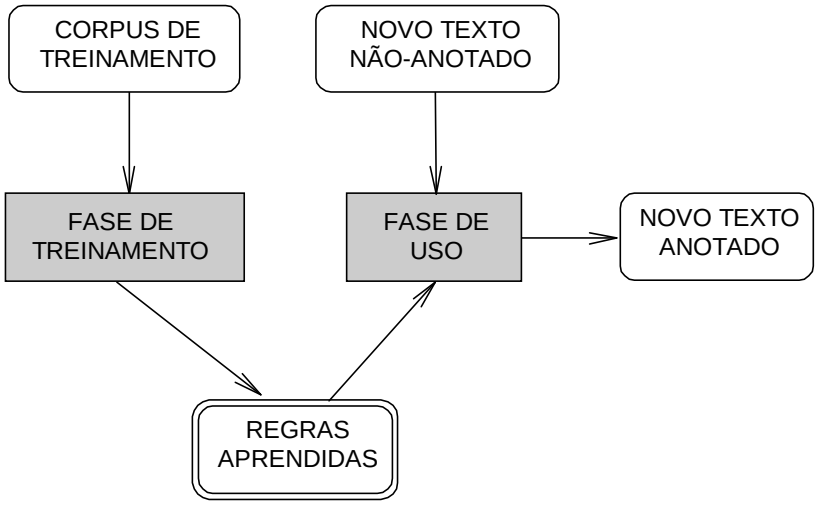


Figura 4 – As duas fases do paradigma de aprendizado

O texto não-anotado é associado a um conjunto de etiquetas, as quais, de acordo com Brill, poderiam ser geradas até aleatoriamente. Obviamente, melhor será quanto mais acertada for esta anotação inicial, mas isto, segundo o autor, não é crítico. Na verdade, o algoritmo faz uma busca heurística do tipo "subida ao monte pelo maior aclive" [*], que tenta, a cada passo, encontrar uma solução (conjunto de regras) que dê uma precisão ao processo de etiquetação acima de um limiar pré-especificado. Logo, é natural que o estado inicial influencie o processo de busca, mas, como mostram os experimentos citados, o uso de um estado inicial aleatório é responsável por uma pequena degradação na precisão da anotação [BRILL-93, p. 100, 112 e 113]. Diz-se que o texto com o conjunto de suas primeiras anotações está no ESTADO INICIAL.

A **fase de treinamento** é constituída por um laço de iterações, no qual o texto no ESTADO INICIAL evoluirá para outros estados, e que só é encerrado quando se chega ao objetivo: obter um conjunto de regras transformacionais que, quando aplicadas a um texto qualquer (livre) com uma anotação inicial, produza o texto anotado com boa precisão[3].

[3] Define-se precisão de anotação ruim como um valor abaixo de 80%, precisão mediana como algo entre 80% e 90% e precisão boa como algo maior que 90%. Estes valores são arbitrários.

Na primeira iteração do laço, as anotações feitas no ESTADO INICIAL são submetidas a uma regra transformacional, de um conjunto de regras previamente escolhido, e é medida a distância entre a anotação PADRÃO e a conseguida pela aplicação da regra. Esta distância mede quão diferentes são os dois conjuntos de anotações; um tipo de medida, talvez o mais simples, consistiria na contagem das anotações de um conjunto que diferem de suas correspondentes no outro conjunto (Figura 6). O mesmo é feito para todas as regras do conjunto pré-especificado. Aquela regra que conduzir a uma menor distância (portanto, levando o anotador mais próximo do ideal) entra para a lista de regras aprendidas. Define-se ESTADO INTERMEDIÁRIO como o estado no qual se encontra o texto depois da aplicação desta regra.

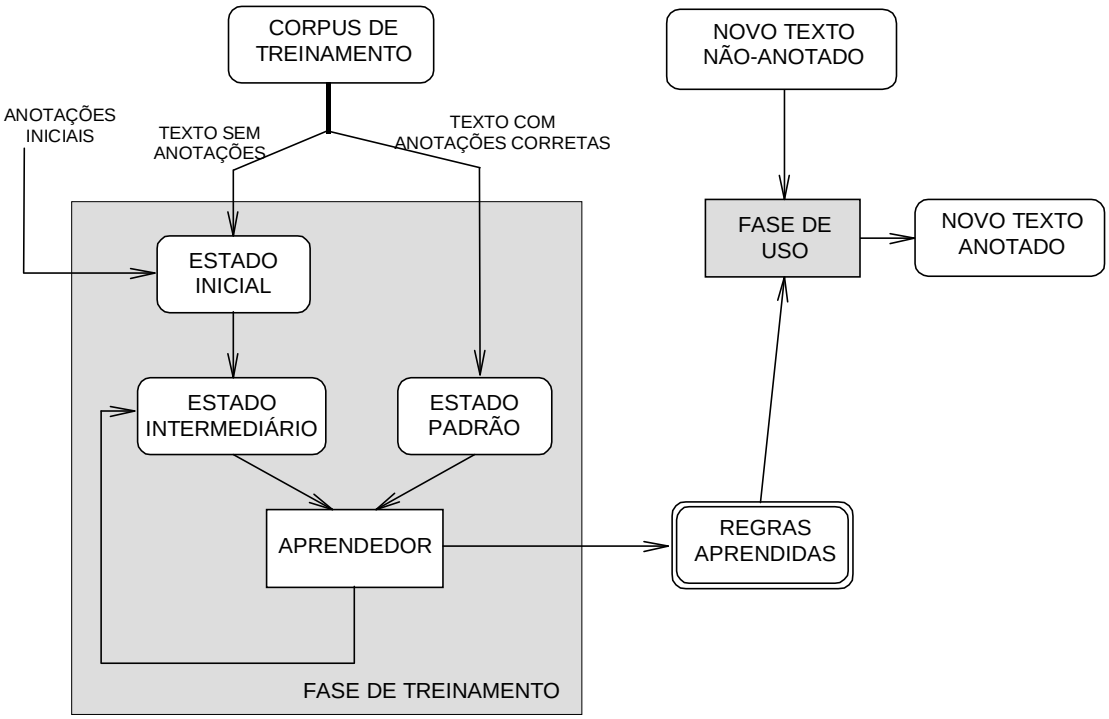


Figura 5 – Detalhamento da fase de treinamento do paradigma de aprendizado

Durante a segunda iteração do laço, o texto no ESTADO INTERMEDIÁRIO é submetido novamente a este processo, e a segunda regra aprendida é então colocada na lista. E assim por diante, até que a distância medida entre o conjunto de anotações PADRÃO e o conjunto de anotações do ESTADO INTERMEDIÁRIO caia a um valor abaixo de um limiar pré-especificado, valor este que representa o máximo erro aceitável durante a fase de treinamento (um melhor entendimento do laço de

treinamento é facilitado pela Figura 7, que ilustra sucintamente o funcionamento do bloco Aprendedor, utilizado pelo paradigma de aprendizado).

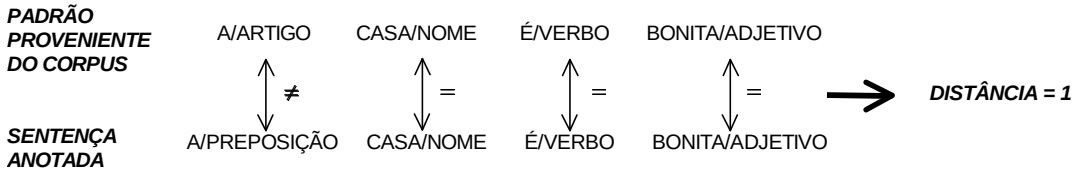


Figura 6 – Uma medida de distância entre anotações morfológicas de sentenças

Resumidamente, podemos descrever o paradigma de aprendizado através do seguinte algoritmo:

```
ESTADO_INICIAL = ANOTAÇÕES_INICIAIS(TEXTO_NÃO_ANOTADO);
ESTADO_INTERMEDIÁRIO = ESTADO_INICIAL;
ESTADO_PADRÃO = TEXTO_COM_ANOTAÇÕES_CORRETAS;
REGRAS_APRENDIDAS = {};

faça
{
   regra = aprenda_uma_regra(ESTADO_INTERMEDIÁRIO, ESTADO_PADRÃO);
   REGRAS_APRENDIDAS = REGRAS_APRENDIDAS + regra;
   ESTADO_INTERMEDIÁRIO = aplique_regra(regra, ESTADO_INTERMEDIÁRIO);
} enquanto (distância(ESTADO_PADRÃO, ESTADO_INTERMEDIÁRIO) > erro_aceitável);
```

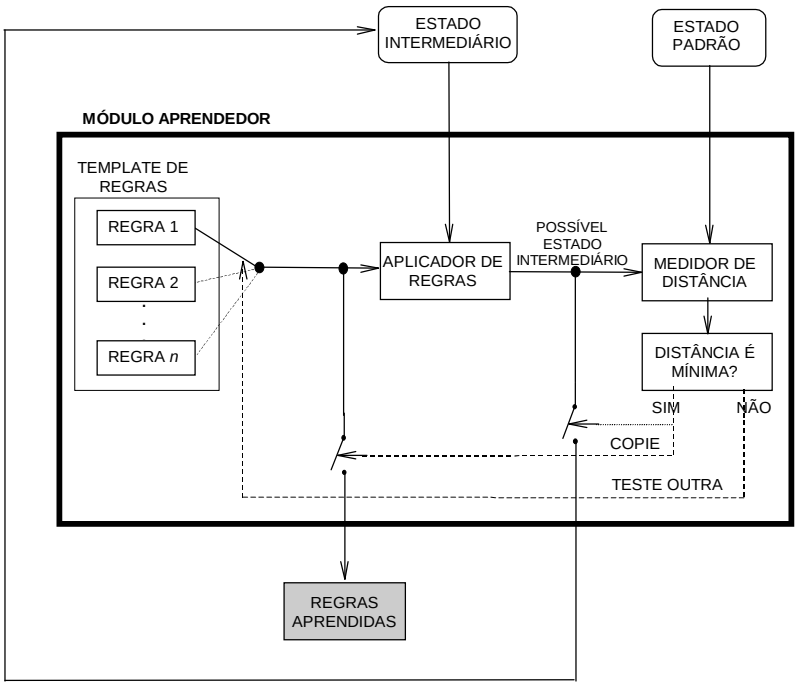


Figura 7 – Diagrama de funcionamento do módulo Aprendedor

Já a **fase de aplicação** do paradigma (Figura 8) consiste simplesmente na utilização das regras aprendidas, uma a uma seqüencialmente, sobre o conjunto formado pelo texto que se quer etiquetar acrescido de anotações iniciais (conforme descrito anteriormente, a respeito da fase de treinamento); define-se que este conjunto formado pelo texto a etiquetar e pelas suas anotações iniciais encontra-se no ESTADO INICIAL.

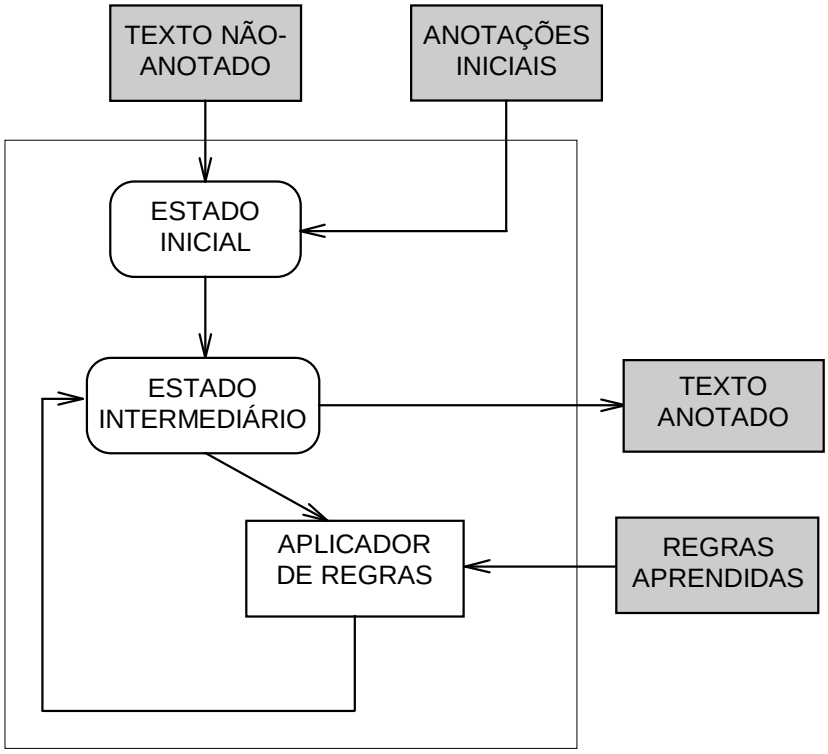


Figura 8 – Fase de aplicação do paradigma de aprendizado

Após todas as regras terem sido aplicadas, o texto de entrada, no ESTADO INTERMEDIÁRIO, estará anotado com grande precisão, visto que as regras inferidas guardam informações lexicais (por exemplo, qual é o sufixo da palavra a ser classificada) e contextuais (por exemplo, quais as etiquetas que estão em sua vizinhança) que possibilitam esta classificação.

Há três aspectos que devem ser especificados para que o paradigma de aprendizado possa ser usado [BRILL-93, p. 40]:

- O formato geral das regras que podem ser aplicadas às anotações (*template*). Por exemplo, o formato de regra utilizado neste paradigma quando aplicado à análise morfológica foi "Mude a etiqueta morfológica atual de ***X*** para ***Y***", sendo que ***X*** e ***Y*** pertencem ao conjunto de todas as etiquetas morfológicas possíveis.
- Os ambientes de disparo das regras, ou seja, quais são as informações lexicais ou contextuais que permitem a execução de uma dada regra. Como exemplos de ambientes de disparo, têm-se:
    1. A palavra atual é ***P***.
    2. A palavra anterior é etiquetada como ***R***.
    3. A palavra seguinte é etiquetada como ***R***.
- Uma função de distância (usada apenas na **fase de treinamento**) para determinar quão diferente está um conjunto de etiquetas (ESTADO INTERMEDIÁRIO) do conjunto padrão (fornecido pelo corpus de treinamento). A Figura 6 ilustrou um modo simples de contabilizar esta distância: basta realizar a contagem de quantas etiquetas de um conjunto são distintas das correspondentes no outro conjunto.

  Desta forma, possíveis regras são:
  - "Se a palavra anterior for etiquetada como '.', mude a etiqueta morfológica atual de 'CL' para 'DET'".
  - "Se a palavra seguinte for etiquetada como 'VB', mude a etiqueta morfológica atual de 'ADV' para 'ADJ'".

O mesmo paradigma é utilizado para tarefas nos mais variados níveis de anotação. Para se fazer uma análise sintática de um texto, por exemplo, é necessário [BRILL-93, p. 48]:

(1) Achar qual é a categoria morfológica mais provável das palavras no corpus de treinamento (Figura 9, item 1), através de estatísticas simples, e aplicar esta informação ao texto a ser analisado. Esta é uma primeira aproximação, a qual *não utiliza o paradigma acima descrito*, e que serve apenas para estimar a categoria morfológica de palavras que tenham sido usadas no treinamento; isto é feito para fornecer uma anotação inicial melhor que uma anotação meramente aleatória, conforme comentado anteriormente;

(2) Quanto às palavras *desconhecidas*, ou seja, *que não apareceram no corpus de treinamento*, aplicar o paradigma de aprendizado para determinar um conjunto de regras, as quais são baseadas em informações lexicais (sufixos, prefixos, etc.) provenientes das palavras que compõem o corpus de treinamento, com a finalidade de fazer uma tentativa inicial de classificação morfológica. Por exemplo, observar o sufixo de uma palavra pode conduzir a uma boa aproximação de sua categoria morfológica: em português, se uma palavra tiver o sufixo "mente", com grande probabilidade, será um advérbio; em inglês, se uma palavra termina em "ed", provavelmente será o particípio passado ou o passado simples de um verbo (Figura 9, item 2). Como já foi comentado, segundo o autor deste paradigma, esta anotação inicial poderia ser feita de qualquer outra maneira, até mesmo fornecendo etiquetas aleatórias às palavras do texto; contudo, parece claro que esta heurística conduz a uma anotação inicial mais acertada e também a uma precisão maior do processo de etiquetação como um todo.

(3) Agora, o mesmo paradigma de aprendizado é usado para se aprender outro conjunto de regras que reflitam informações contextuais sobre as categorias morfológicas (Figura 9, item 3). Uma vez que estas regras aprendidas são aplicadas, as categorias morfológicas das palavras do texto estarão determinadas, com uma grande precisão (tipicamente maior que 90% [BRILL-93]).

(4) A partir de um corpus de treinamento com anotações morfológicas e sintáticas (Figura 9, item 3), pode-se aplicar o paradigma de aprendizado para a inferência de regras que consigam determinar a estrutura sintática das sentenças em um dado texto (Figura 9, item 4). O treinamento é realizado da maneira já descrita: o corpus de treinamento é apresentado nas versões anotada e não-anotada estruturalmente, e uma etiquetagem inicial é atribuída à versão não-anotada. Então, regras que organizarão corretamente estruturas sintáticas são inferidas e serão aplicadas ao texto que se deseja anotar sintaticamente; este texto já deve estar previamente anotado morfologicamente e com anotações sintáticas iniciais.

(5) A partir do texto com anotações estruturais, pode-se aplicar o paradigma de aprendizado para a inferência de regras que sirvam para etiquetar os não-terminais das estruturas sintáticas (por exemplo, locução verbal, locução nominal, etc.). O treinamento e a aplicação são realizados de modo idêntico ao dos módulos anteriores (Figura 9, item 5).

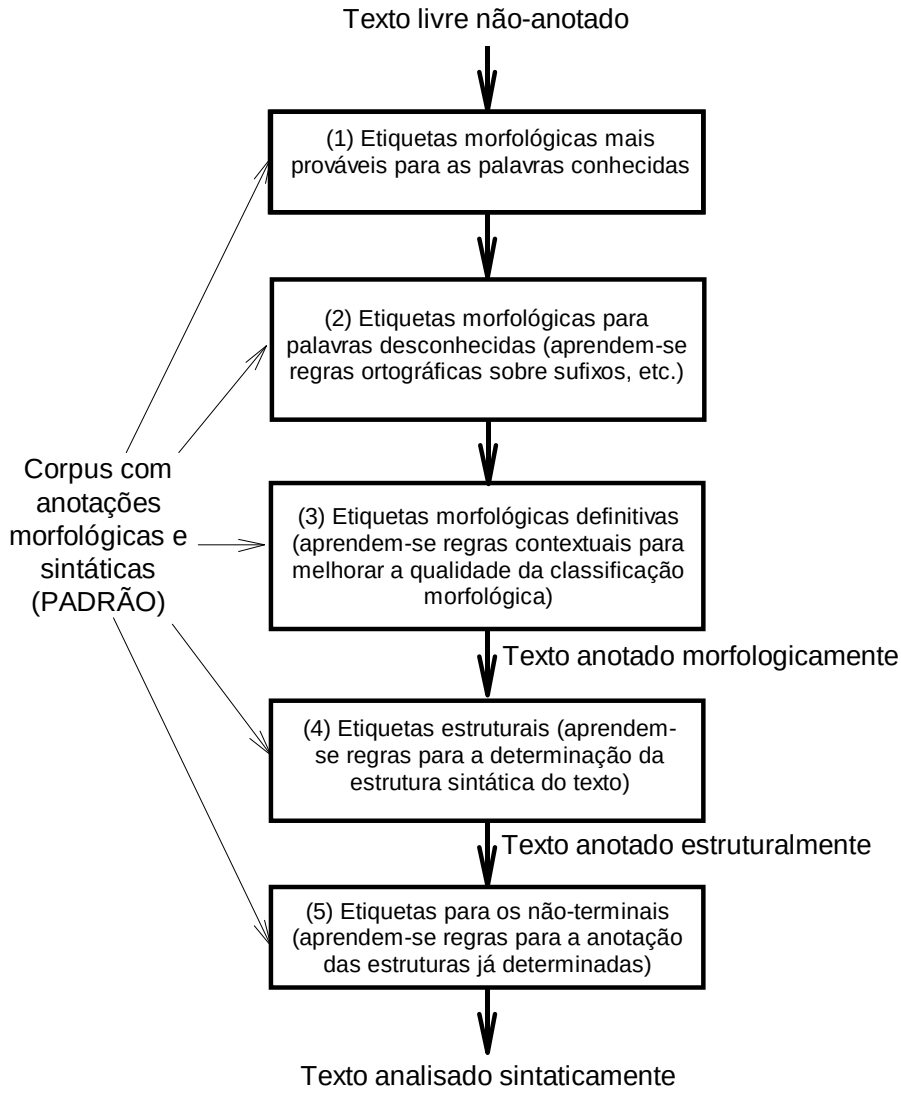


Figura 9 – Diversos níveis de utilização do paradigma de aprendizado "dirigido a erro baseado em transformações"

É interessante notar o motivo de o autor ter proposto fazer a análise sintática em vários passos ao invés de em um único passo. Basicamente, é por causa do problema de dados esparsos, ou seja, é muito mais fácil aprender que um artigo qualquer pode vir antes de um substantivo qualquer, que aprender toda uma lista de pares de artigo com substantivo, sem qualquer generalização [BRILL-93, p. 50-1]. Em outras palavras, todas estas passagens são necessárias para que o módulo de aprendizado possa abstrair classes através do seu treinamento e codificá-las num conjunto conciso de regras. Se alguns destes passos fossem omitidos, o número de

regras aprendido nos módulos restantes seria muito maior, justamente pela dificuldade na abstração de classes (Figura 10).

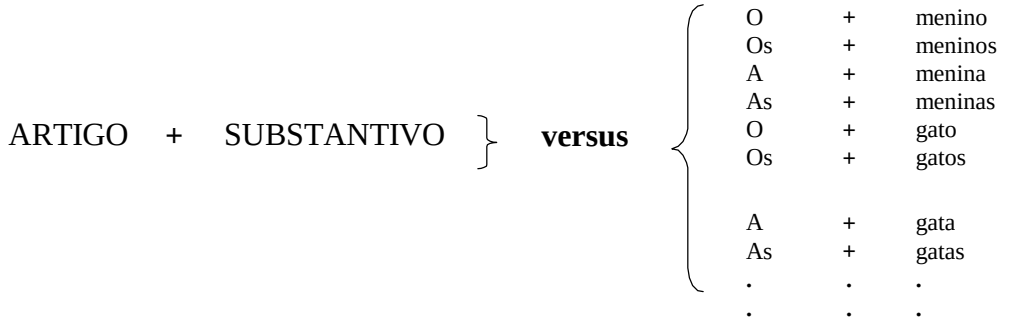


Figura 10 – Abstração de classes versus lista com todas as combinações possíveis de palavras

A Figura 11 resume a arquitetura completa de um analisador morfológico que segue o paradigma de aprendizado "dirigido a erro baseado em transformações"; as três caixas com bordas arredondadas nesta figura representam as três etapas necessárias para se efetuar uma análise morfológica com boa precisão: **etiquetagem inicial de palavras conhecidas** (eqüivale ao item 1 da Figura 9), **etiquetagem inicial de palavras desconhecidas** (eqüivale ao item 2 da Figura 9) e **refinamento contextual da etiquetagem** (eqüivale ao item 3 da Figura 9).

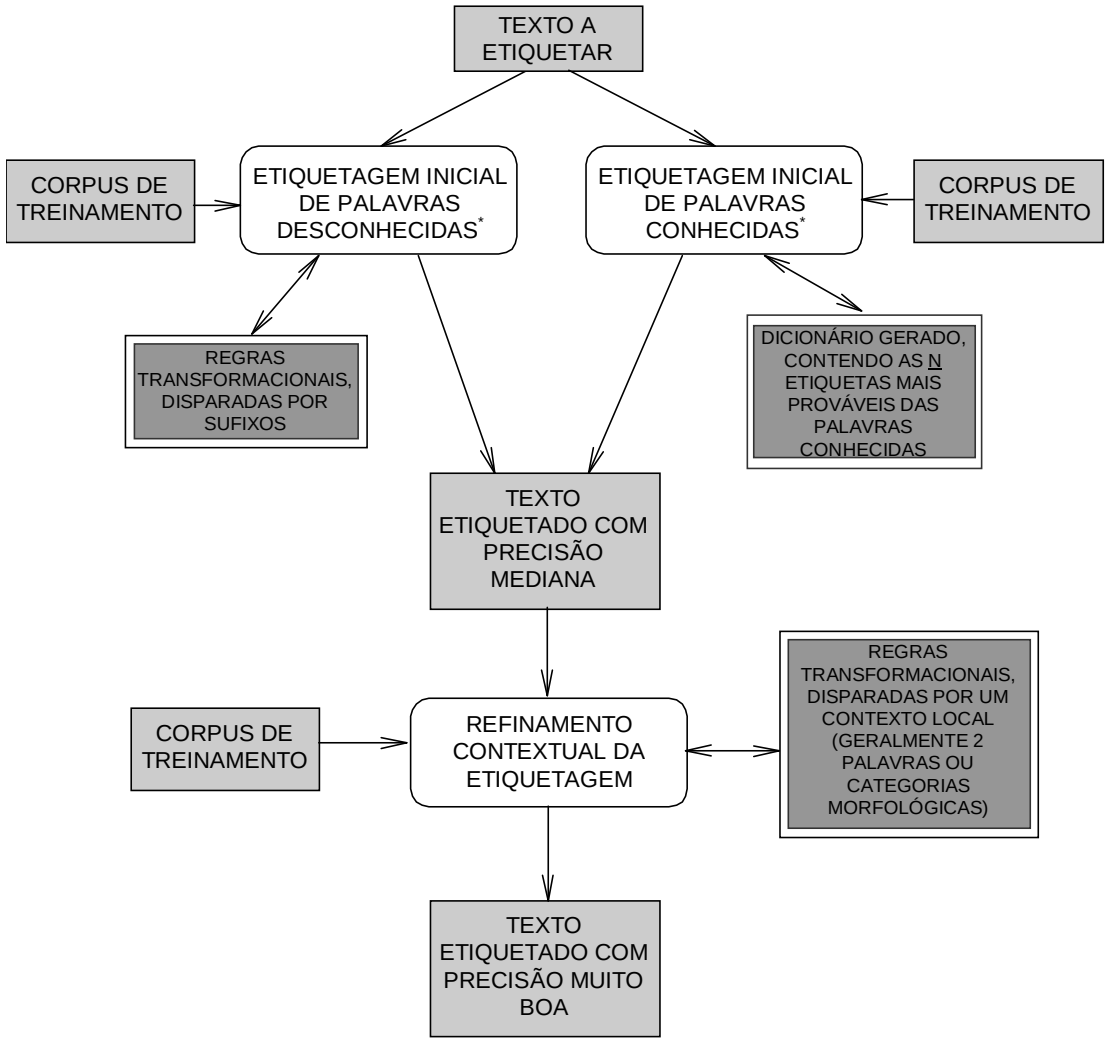
As experiências de E. Brill para a língua inglesa usaram três conjuntos de etiquetas: o do *Penn Treebank*, com 36 etiquetas morfológicas (sem considerar as pontuações), o do *Old English*, com 19 etiquetas, e o do *Brown*, com 80 etiquetas. O tamanho dos corpora usados nas experiências variou de 45.000 a 200.000 itens lexicais aproximadamente.

O desempenho do processo de etiquetação morfológica ficou entre 85% e 96% de acertos, dependendo do corpus de treinamento utilizado, do tamanho do mesmo, do conjunto de etiquetas escolhido e da consonância entre o corpus de treinamento e o de aplicação [BRILL-93].

Relata-se também um outro trabalho que usa este paradigma como base de um etiquetador para a língua portuguesa clássica, cuja arquitetura pode ser vista na Figura 12 [ALVES-99].

A grande dificuldade em adaptar o paradigma de Brill para a língua portuguesa reside em sua maior riqueza morfológica em comparação com a língua inglesa; isto

faz com que os lingüistas tenham de especificar um conjunto de etiquetas maior para anotar informativamente um texto. Por exemplo, o Corpus Tycho Brahe, que foi usado para as experiências com este etiquetador, usa um conjunto com um total de 231 etiquetas: são 36 etiquetas primárias com mais quatro subníveis de anotação, como pode-se observar na etiqueta VB-AN-G-P+SE, onde VB é a etiqueta primária. E a complexidade computacional da etapa de treinamento do etiquetador de Brill é da ordem do tamanho do conjunto de etiquetas usadas pelo corpus elevado à quarta potência; isto faz com que a fase de treinamento deste algoritmo fique computacionalmente inviável quando o número de possíveis etiquetas é grande [ALVES-99].



*Palavras conhecidas são aquelas que apareceram no corpus de treinamento; palavras desconhecidas, portanto, são aquelas que não apareceram

Figura 11 – Modelo macroscópico de um analisador morfológico, conforme proposto por Eric Brill

A idéia deste trabalho é usar, numa primeira fase, apenas o núcleo das etiquetas, reduzindo-se, assim, o conjunto delas para apenas 36 etiquetas, e, em seguida, refinar esta saída. Este processo de refinamento consiste em adicionar os complementos às etiquetas primárias (núcleos) através de regras, **escritas à mão**, que levam em conta prefixos e sufixos da palavra cuja etiqueta será refinada.

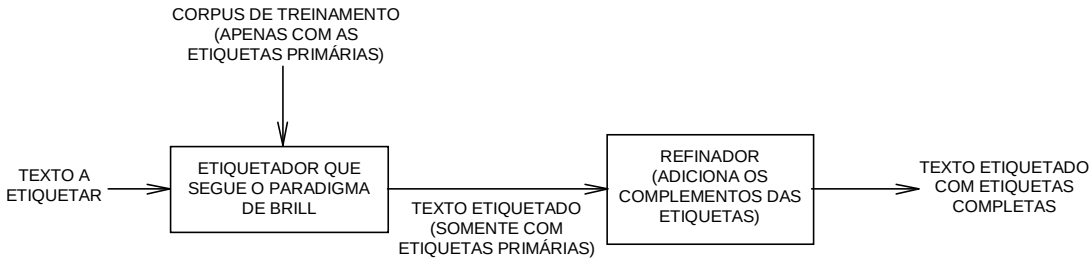


Figura 12 – Arquitetura de um etiquetador, que segue o paradigma de Brill, para a língua portuguesa

Este artigo relata experimentos iniciais, nos quais foi usado um corpus pequeno de 5.000 palavras, e que conseguiu uma taxa de acerto na anotação morfológica de 78,28%. Este valor pode ser razoável apenas se for considerado o tamanho do corpus utilizado.

2.1.4 Analisadores morfológicos baseados em exemplos memorizados

Esta proposta é uma variante do paradigma de aprendizado automático baseado em exemplos memorizados, denominado *lazy learning* (aprendizado preguiçoso). Este paradigma de aprendizado, assim como todos os outros, pressupõe duas fases distintas: a de treinamento, na qual alguma forma de informação ou conhecimento é inferida e armazenada, e a de aplicação, na qual a informação inferida é utilizada para alguma tarefa de classificação [MITCHELL-97].

O nome *lazy* (preguiçoso) vem do fato de que todo o processamento computacionalmente intensivo é postergado para a fase de aplicação ou classificação do paradigma, sendo que a fase de treinamento se resume em armazenar exemplos. É apenas quando há a necessidade de classificar um item lexical num dado contexto que

ocorrem cálculos para achar qual é o exemplo, dentre os armazenados na memória, mais semelhante a este.

W. Daelemans e outros pesquisadores propuseram um método para utilizar tal paradigma de aprendizado na construção de analisadores morfológicos [DAELEMANS-96a].

A **fase de treinamento** deste método consiste em manter armazenado em uma memória um conjunto de casos, sendo que cada um consiste de:

- um **item lexical** (uma palavra ou uma pontuação), o qual será denominado Foco; deste, serão realmente utilizadas a primeira e as últimas três letras, conforme explicado adiante.
- a **categoria morfológica ou etiqueta** atribuída ao Foco.
- o **contexto que precede** o Foco (as categorias morfológicas dos dois itens lexicais imediatamente anteriores ao Foco).
- o **contexto que sucede** o Foco (a categoria morfológica do item lexical imediatamente posterior ao Foco).

Todas estas informações que compõem um caso são retiradas do corpus de treinamento. Desta forma, pode-se dizer que haverá tantos casos quantos forem os itens lexicais do corpus.

A **fase de aplicação** consiste em usar esta base de casos para a finalidade de inferir a etiqueta de um novo item lexical, dentro de uma sentença.

Algo que é de utilidade para o algoritmo de aplicação é uma lista dos itens lexicais que apareceram no corpus de treinamento, sendo que cada um destes itens é associado a uma ou mais etiquetas morfológicas, gerando, assim, um **léxico** anotado morfologicamente.

A heurística proposta segue duas possibilidades, dependendo de o item lexical (Foco) ser conhecido (já ter aparecido no corpus de treinamento) ou desconhecido.

Se o Foco é **conhecido**, são examinados:

- o contexto anterior ao Foco (duas etiquetas anteriores, as quais, presumivelmente, são corretas).
- o contexto posterior ao Foco (uma etiqueta posterior, a qual é adquirida do léxico, visto que o item lexical posterior ao Foco ainda não foi etiquetado; possivelmente, esta é uma etiqueta ambígua).

- uma etiqueta ambígua associada ao próprio Foco (também adquirida do léxico).

Buscar-se-á na base de casos qual é aquele cujas características mais se aproximem (ou seja, com menor distância, conforme explicado à frente) das características acima citadas associadas ao Foco; a categoria morfológica referente a este caso mais próximo é usada como etiqueta para o Foco (o item lexical a ser etiquetado).

Se o Foco é **desconhecido**, são examinados:

- os contextos anterior (uma etiqueta) e posterior (também uma etiqueta, a qual vem do léxico e é possivelmente ambígua).
- a primeira e as três últimas letras do Foco.

Novamente, o algoritmo buscará qual é o caso cujas características estão à menor distância destas características acima; a categoria morfológica referente a este caso mais próximo é usada como etiqueta para o Foco.

Nota-se que há a necessidade de uma **medida de distância** para comparar as informações relativas ao item lexical a ser etiquetado (será denominado *X*) com os diversos casos armazenados (um caso qualquer será denominado *Y*). O corrente método utiliza a seguinte definição de distância entre *X* e *Y*:

$$\Delta(X,Y) = \sum_{i=1}^{n} G_i \cdot \delta(x_i, y_i) \quad \text{, onde:}$$

x_i é uma das informações referentes ao item lexical que será etiquetado (por exemplo, etiqueta imediatamente anterior). Estas informações são também denominadas **características**.

y_i é uma **característica**, correspondente a x_i, em um dos casos armazenados.

$\delta(x_i, y_i) = 0$ se $x_i = y_i$, senão $\delta(x_i, y_i) = 1$

G_i é chamado Ganho de Informação; este é um cálculo estatístico que mede a relevância de uma **característica** para com a medida de distância. Ou seja, a característica que tiver maior Ganho de Informação é aquela que influenciará mais a medida de distância (nota-se que esta é uma espécie de média ponderada, onde os Ganhos de Informação das *n* características representam os pesos de ponderação). E sabe-se intuitivamente que o contexto mais próximo de uma

palavra afeta sua categoria morfológica mais que o contexto mais distante, e isto foi detectado pelas medidas dos autores deste trabalho: o Ganho de Informação referente à primeira etiqueta anterior ao Foco foi de 0,22 e o referente à segunda etiqueta anterior foi de 0,06.

Uma observação importante a ser feita é que este método, bem como o paradigma do qual ele é derivado, pressupõem que todas as decisões são tomadas com base no reaproveitamento direto de exemplos armazenados numa memória (ou base de dados). Não há regras inferidas automaticamente, como no caso do paradigma citado na seção 2.1.3, e também não há um conjunto de regras escritas por seres humanos (em geral, lingüistas com grande experiência), como no paradigma comentado na seção 2.1.2.

Os autores deste método comentam que o mesmo compartilha as vantagens dos outros e ainda exibe algumas características peculiares que, entre outras, são [DAELEMANS-96a]:

- A necessidade de um corpus de treinamento relativamente pequeno.
- Aprendizado incremental, ou seja, a qualquer momento pode-se acrescentar novos exemplos à memória de casos, sem a necessidade de refazer o treinamento.
- Capacidade de explicar suas decisões.
- Treinamento e Aplicação rápidos.

As três primeiras são típicas do paradigma *lazy learning*, mas não a última: neste, o treinamento é realmente muito rápido, contudo a aplicação é lenta, já que envolve uma extensa pesquisa num banco de dados e cálculos para determinar a maior similaridade.

O que diferencia a proposta corrente é o uso de um algoritmo baseado em estruturas de árvores, chamado IGTREE, para a indexação e busca de informações em grandes bases de casos; seu uso reduziu o tempo de classificação de 100 a 200 vezes, em comparação com a implementação tradicional do paradigma baseado em exemplos memorizados, e usou cerca de 95% menos espaço em memória [DAELEMANS-96b].

Resumidamente, este algoritmo trabalha da seguinte maneira: o Ganho de Informação para cada uma das características é calculado; a característica que possuir maior Ganho de Informação é usada na seleção existente no primeiro nível da árvore (Figura 13), a que possuir o segundo maior Ganho de Informação, na seleção no segundo nível, e assim por diante. Deste modo, este algoritmo prioriza as decisões mais relevantes para as comparações realizadas com os dados do banco de casos. Pode-se fazer facilmente uma poda em alguns níveis da árvore quando a relevância ou Ganho de Informação destes for pequena: isto representa um processo de generalização de conhecimento.

A arquitetura do analisador é muito semelhante à de outros [BRILL-93]; é formada por um módulo que cuida da etiquetagem das palavras conhecidas e outro que cuida das palavras desconhecidas; também o contexto é observado dentro de cada um destes módulos, com o objetivo de encontrar a anotação mais correta possível (Figura 14).

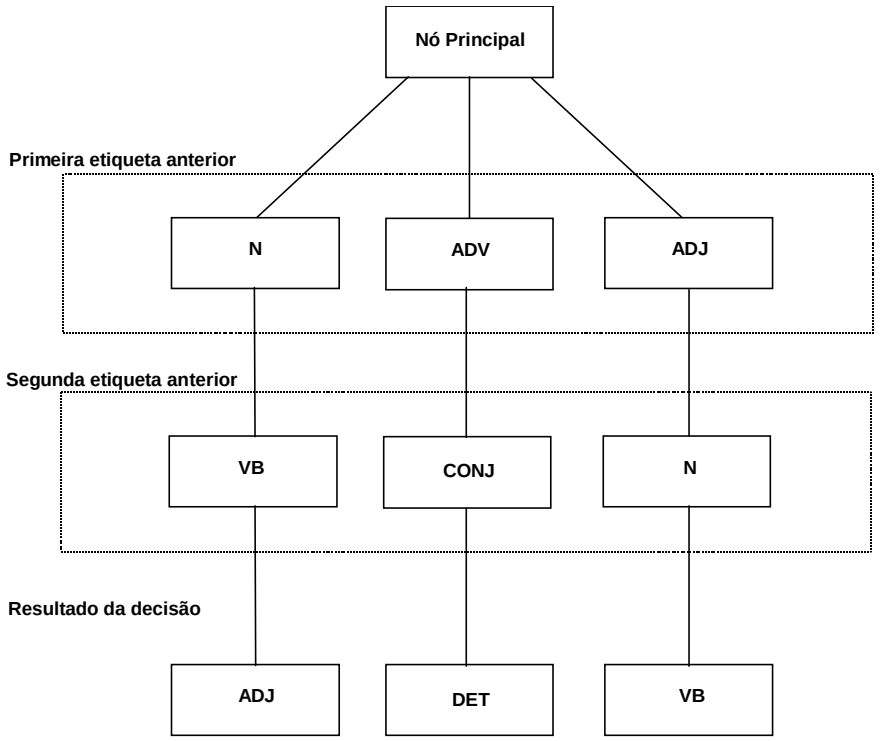


Figura 13 – Estrutura em árvore para armazenamento de grande base de casos (IGTree)

A título de comparação, é importante saber que o corpus de treinamento usado em seus experimentos era composto por um conjunto de 44 possíveis etiquetas.

Segundo os autores, seu analisador não necessita de um corpus tão grande quanto o que é exigido pelos métodos estatísticos. Eles sugerem que o tamanho mínimo para a produção de resultados razoáveis seria de 300 mil palavras; mesmo assim, alguns de seus experimentos usaram um corpus de 3 milhões de casos (grandes corpora para a língua inglesa são comuns). O desempenho na tarefa de etiquetação morfológica chegou a aproximadamente 96% de acerto.

Por todos estes dados, é possível concluir que o desempenho deste método, em termos de acerto na anotação, é muito bom e é similar ao que os métodos estatísticos e o baseado em regras inferidas automaticamente alcançaram.

Uma característica bastante interessante e útil deste método é a possibilidade de explicar uma decisão tomada, com base na maior proximidade de um determinado exemplo memorizado.

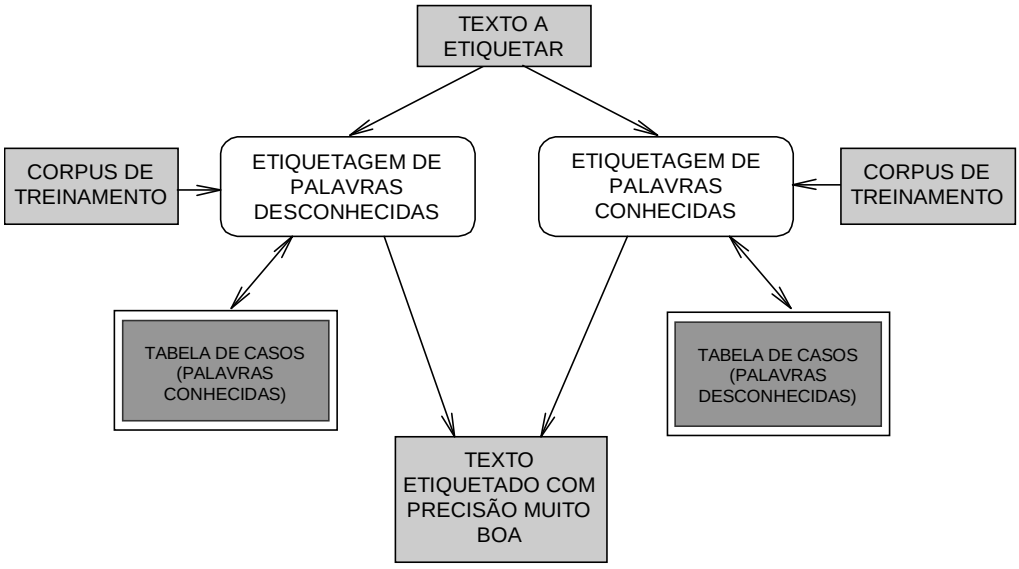


Figura 14 – Modelo macroscópico de um analisador morfológico, conforme proposto por W. Daelemans e outros pesquisadores

2.2 Autômatos Adaptativos

Os autômatos adaptativos (AA) constituem um formalismo para a representação de linguagens dependentes de contexto [JOSÉ NETO-94].

A base estrutural de um AA é um autômato de pilha [JOSÉ NETO-87]; o que os diferencia é que um AA pode ter, associado a cada uma de suas transições, funções adaptativas, anteriores e posteriores, conforme explanado adiante (Figura 15).

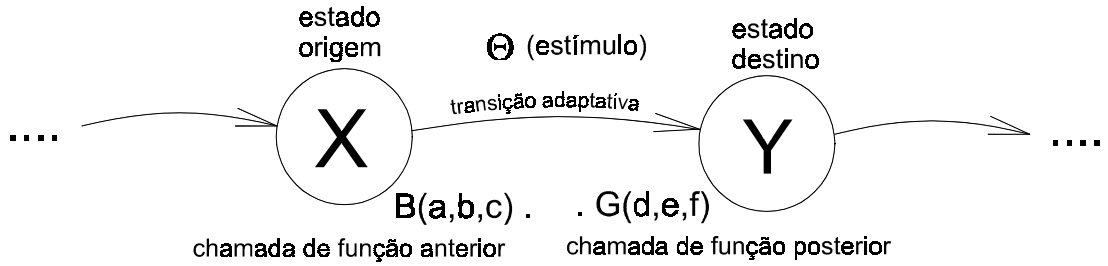


Figura 15 – Uma transição de Autômato Adaptativo (notação gráfica)

As funções adaptativas são constituídas de um conjunto de **ações adaptativas elementares** que possibilitam modificar o autômato como decorrência da execução de uma transição, através do acréscimo e retirada de estados e transições.

As **ações adaptativas elementares** podem ser de três tipos:

- **Inspeção**: serve para determinar alguma informação útil do autômato em questão como, por exemplo, de qual estado sai uma transição com determinado estímulo, chegando num dado estado, ou qual é o estímulo que permite a transição de um determinado estado para outro estado, etc. Independentemente da posição em que as ações de inspeção apareçam na declaração da função adaptativa, elas sempre são as primeiras a serem executadas, em relação aos três tipos de ações adaptativas elementares.
- **Eliminação**: serve para a modificação da topologia do autômato pela retirada de uma transição. Independentemente da posição em que as ações de eliminação apareçam na declaração da função adaptativa, elas sempre são executadas após a execução das ações de inspeção, e antes das de inserção.
- **Inserção**: serve para a modificação da topologia do autômato pela inserção de uma nova transição. Independentemente da posição em que as ações de inserção apareçam na declaração da função adaptativa, elas sempre são executadas por último.

São estes dois últimos tipos de ações adaptativas elementares que dão aos AA o poder computacional para manipular linguagens dependentes de contexto [JOSÉ NETO-94].

As **chamadas de funções adaptativas** podem ser de dois tipos:

- **Anterior:** é efetuada sempre antes de uma transição ocorrer; se o estímulo referente à transição foi encontrado na entrada, a chamada de função anterior é acionada, sendo que o estado atual neste momento ainda é o estado origem, já que a transição ainda não ocorreu.

  Se a transição em questão for excluída durante a execução desta função (isto é possível através da execução eventual de uma ação adaptativa elementar de **Eliminação**, explanada anteriormente), a alteração é mantida, mas a transição é abortada, sendo procurado um novo caminho para realizar a transição a partir do estado origem, até que isto seja conseguido, ou que seja impossível realizar uma transição (o símbolo encontrado na entrada não foi consumido nesta situação). Caso a transição em questão seja mantida, o símbolo encontrado na entrada é consumido e o estado destino da transição se torna o novo estado corrente do autômato.
- **Posterior:** é efetuada sempre depois que a mudança de estado é realizada; o estado corrente, no momento em que esta função é chamada já é o estado destino da transição.

Deve-se ressaltar que dentro de uma função adaptativa é possível realizar a chamada de uma função adaptativa qualquer antes da execução do conjunto das ações adaptativas elementares que a compõem (isto é chamado **ação adaptativa inicial**) e também uma outra chamada qualquer depois da execução desse conjunto de ações elementares (isto é chamado **ação adaptativa final**).

Também, como primeira parte do corpo das funções adaptativas, pode-se declarar identificadores que representam elementos dentro do escopo do corpo da função adaptativa. São eles de dois tipos: **geradores** (que recebem um valor único, que nenhum outro elemento possui, logo ao início da execução da função adaptativa, e permanece com este valor até o seu término) e **variáveis** (que recebem seus valores como resultado da execução das ações adaptativas elementares de inspeção e

eliminação, permanecendo até o término da execução da função com esse mesmo valor).

A declaração das funções adaptativas segue a forma mostrada na Figura 16.

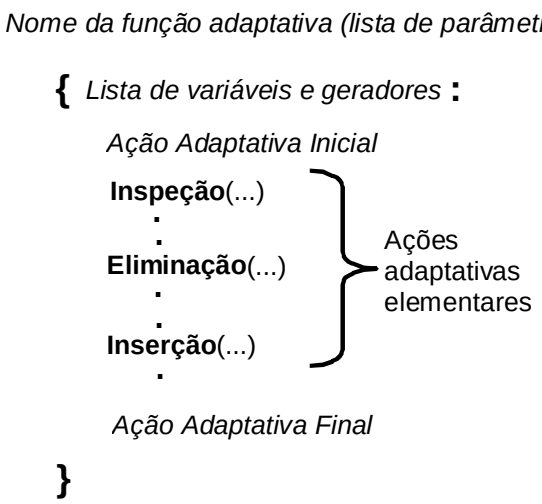


Figura 16 – Formato da declaração de uma função adaptativa

A característica de poder alterar sua própria topologia, peculiar aos autômatos adaptativos, faz com que eles sejam bastante adequados à modelagem de sistemas de aprendizado automático: um conjunto de exemplos poderia ser inserido em um AA (treinamento) na forma de novas transições; deste modo um AA pode incorporar conhecimento.

2.3 Comentários

Este capítulo citou os trabalhos considerados como pertencendo ao estado-da-arte na área do processamento de linguagens naturais.

Quanto ao aspecto de desempenho de anotação, os três paradigmas treináveis, aplicados à anotação morfológica (o estatístico, o baseado em regras inferidas automaticamente e o baseado em exemplos memorizados), conseguem uma taxa de acerto em torno dos 96%. Uma vantagem destes três paradigmas treináveis, em relação ao baseado em regras escritas manualmente, é que eles não necessitam que um humano escreva regras, pois todo o conhecimento necessário é inferido.

Dos três paradigmas treináveis, aquele cujo treinamento é mais simples é o de W. Daelemans, que consiste apenas no armazenamento de um caso numa base de dados [DAELEMANS-96a]. Quanto a qual dos paradigmas precisa do tamanho mínimo de corpus de treinamento, para obter bons resultados, a literatura indica o trabalho de E. Brill [Brill-93].

Os autômatos adaptativos constituem uma boa ferramenta para a implementação da proposta desta dissertação.

Além de ser um autômato, podendo ter uma implementação eficiente, seus recursos de inserir e apagar transições são adequados para o modelamento de algoritmos de aprendizado automático.

3. Proposta

Conforme comentado no primeiro capítulo, comparativamente com o que ocorre, por exemplo, com a língua inglesa, a quantidade de trabalhos desenvolvidos na área do Processamento de Linguagens Naturais para a língua portuguesa é relativamente pequena.

Como uma das contribuições deste trabalho, será analisado um método de construção de um etiquetador morfológico treinável, que possibilite uma boa precisão na anotação (algo superior a 90%, conforme estabelecido no Capítulo 2). Terá por base os pontos fortes de vários dos paradigmas e métodos citados no capítulo anterior. Embora seja totalmente independente da língua para a qual é aplicado, será testado com textos na língua portuguesa.

Uma outra contribuição deste trabalho é a constatação da adequação dos autômatos adaptativos, que foram adotados como formalismo de implementação, para a descrição de algoritmos de aprendizado automático (na verdade, não estão sendo utilizados exatamente os AA, mas uma variante adequada a este caso particular, conforme explanado a seguir).

3.1 Interpretador de Autômatos Adaptativos

Como se optou pelo uso dos autômatos adaptativos como base de implementação das várias etapas do analisador morfológico, seria natural a construção de um interpretador ou de compilador que execute de fato tal autômato em um computador.

Este trabalho descreve a construção de um interpretador para um formalismo variante dos AAs, que será batizado de Autômatos Adaptativos E (“E” vem de Estendido), consistindo de um grande subconjunto dos autômatos adaptativos originais, com algumas extensões (Figura 17).

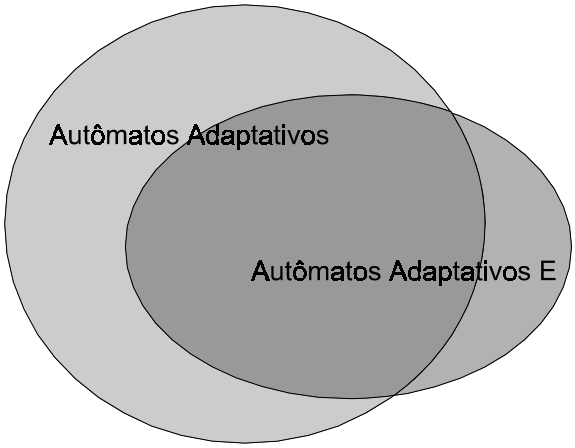


Figura 17 – Autômatos adaptativos E constituem um subconjunto dos autômatos adaptativos, com algumas extensões

Define-se autômato adaptativo E como:

- um AA que não faz uso de pilha.
- estritamente determinístico (o autômato inicial deve ser determinístico e todas as alterações adaptativas devem mantê-lo determinístico). Isto, deveras, contribui para que esta implementação seja mais simples e também para que um futuro compilador não tenha de tirar os indeterminismos.

Também é preciso propor uma extensão ao formalismo dos AA, para que estes possam dar subsídios ao levantamento de dados estatísticos muito simples (apenas contagens), que serão imprescindíveis para a implementação de algoritmos de aprendizado automático.

Esta extensão permite associar a cada transição uma variável inteira, automaticamente iniciada com o valor zero quando a transição é inserida, e criar mais dois tipos de **ações elementares** (estas não serão chamadas **adaptativas** pois não estão diretamente envolvidas nos mecanismos adaptativos):

- **Soma um:** esta ação elementar automaticamente incrementa de um o valor da variável associada a uma transição.
- **Comparação:** esta ação elementar compara o valor da variável associada a uma transição com o valor de outra variável, associada a outra transição, ou então com um valor fixo. É efetuada com base nos operadores básicos de comparação, a saber: maior, maior ou igual, menor, menor ou igual, igual.

Deve-se ressaltar que estes dois tipos de ações poderiam ser realizados estritamente através dos mecanismos intrínsecos dos AA (por exemplo, **soma um** poderia ser representada pela inserção de uma transição em vazio), mas, por questões ligadas à eficiência e praticidade, estes dois tipos de operação não foram implementados desta forma, mas por meio das ações elementares que foram acrescentadas ao formalismo.

Utilizou-se a linguagem de programação Prolog para a implementação deste interpretador por vários motivos práticos, entre os quais:

- a definição de transições é bastante aderente ao paradigma declarativo; estas transições transformam-se em cláusulas de Prolog.
- a unificação, que é uma das características mais marcantes do Prolog, é um meio natural de implementar a ação adaptativa elementar **inspeção**.

Como se pode notar na Figura 18, que mostra a sintaxe típica da descrição de uma transição inicial, e na Figura 19, descrevendo a declaração de uma função adaptativa, a sintaxe de descrição do autômato está muito próxima da sintaxe da linguagem Prolog; e isto advém dos recursos de implementação oferecidos por esta linguagem.

O funcionamento do quarto parâmetro da transição na Figura 18 é o seguinte: o parâmetro é escrito na cadeia de entrada, porém, como sendo algo já lido.

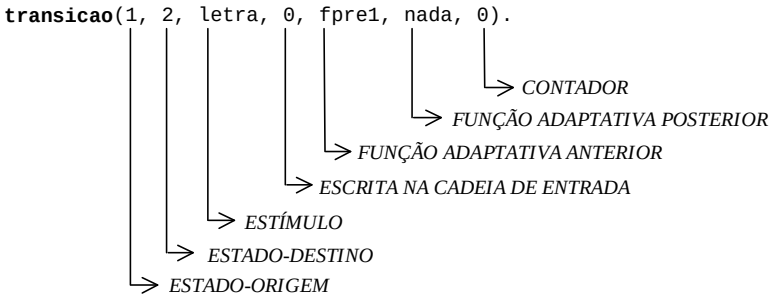


Figura 18 – Sintaxe típica da descrição de uma transição

Na declaração da função fpre1 (Figura 19) notam-se alguns aspectos:

- O interpretador passa automaticamente, como parâmetros, para todas as funções adaptativas, três informações referentes à transição em questão: ESTADO-ORIGEM, ESTADO-DESTINO e CADEIA DE ENTRADA. Esta forma de implementar não segue a definição original dos autômatos adaptativos.
- Usou-se a extensão **soma_um**: o segundo parâmetro conterá o valor do primeiro parâmetro incrementado de um.

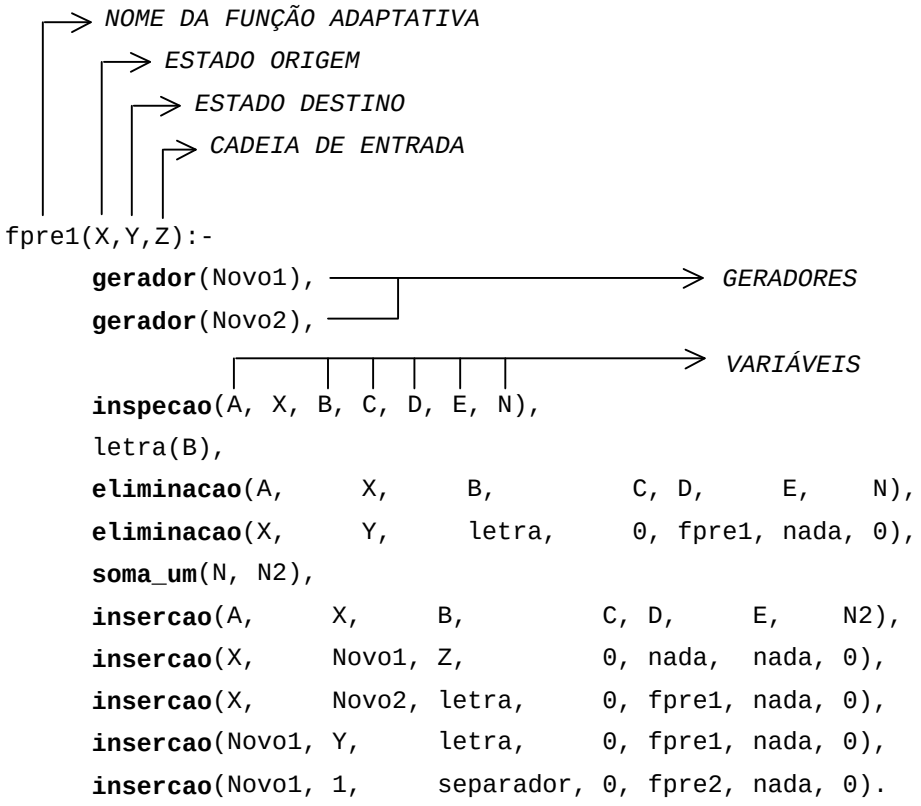


Figura 19 – Sintaxe típica da declaração de uma função adaptativa

Foram definidos diversos estímulos, que serão usados pelos autômatos adaptativos E:

- **estímulo simples:** considera-se um **estímulo simples** um caracter, uma cadeia de caracteres ou uma lista de cadeias de caracteres explícitos. Exemplos: "A", "CONJ", "ADV/N/PREP". Considera-se que o estímulo será bem-sucedido quando houver na primeira posição da cadeia de entrada o referido estímulo. Se a transição à qual este estímulo está associado ocorrer, este mesmo será consumido da cadeia de entrada.
- **barra:** o estímulo barra será bem-sucedido quando houver na primeira posição da cadeia de entrada o caracter '/'. Se a transição à qual este estímulo está associado ocorrer, este mesmo será consumido da cadeia de entrada.
- **separador:** o estímulo **separador** será bem-sucedido quando houver na primeira posição da cadeia de entrada o caracter " "(espaço em branco) ou um caracter que represente a mudança de linha. Se a transição à qual este estímulo está associado ocorrer, este mesmo será consumido da cadeia de entrada.

- **símbolo:** o estímulo **símbolo** será bem-sucedido quando houver na primeira posição da cadeia de entrada qualquer caracter que não seja nem uma barra, nem um separador. Se a transição à qual este estímulo está associado ocorrer, este mesmo será consumido da cadeia de entrada.
- **etiqueta:** o estímulo **etiqueta** será bem-sucedido quando houver na primeira posição da cadeia de entrada qualquer cadeia de caracteres que não inclua separadores ou barras. Se a transição à qual este estímulo está associado ocorrer, este mesmo será consumido da cadeia de entrada.
- **conj_tag:** o estímulo **conj_tag** será bem-sucedido quando houver na primeira posição da cadeia de entrada uma lista qualquer de pelo menos duas **etiquetas**, separadas por uma barra. Se a transição à qual este estímulo está associado ocorrer, este mesmo será consumido da cadeia de entrada.
- **lap:** o estímulo **lap** (*look-ahead* de pertinência) é sempre acompanhado de um parâmetro que constitui um **estímulo simples**. Será bem-sucedido em duas situações: (1) quando houver na primeira posição da cadeia de entrada o **estímulo simples** que é o parâmetro do **lap** ou (2) quando houver na primeira posição da cadeia de entrada uma lista de etiquetas, e o parâmetro do **lap** for uma das etiquetas desta lista. De qualquer modo, nada será consumido da cadeia de entrada.
- **vazio:** o estímulo **vazio** será bem-sucedido sempre e nada será consumido da cadeia de entrada.

Como já comentado anteriormente, o comportamento de um autômato adaptativo E é estritamente determinístico.

Portanto, precisa-se especificar uma ordem na qual os diferentes tipos de estímulos são examinados pelo interpretador (Figura 20). Define-se a precedência na qual se observarão os estímulos como sendo a seguinte:

1. Estímulo simples.
2. Lap.
3. Símbolo, barra, etiqueta, conj_tag (todos com igual prioridade).
4. Vazio.

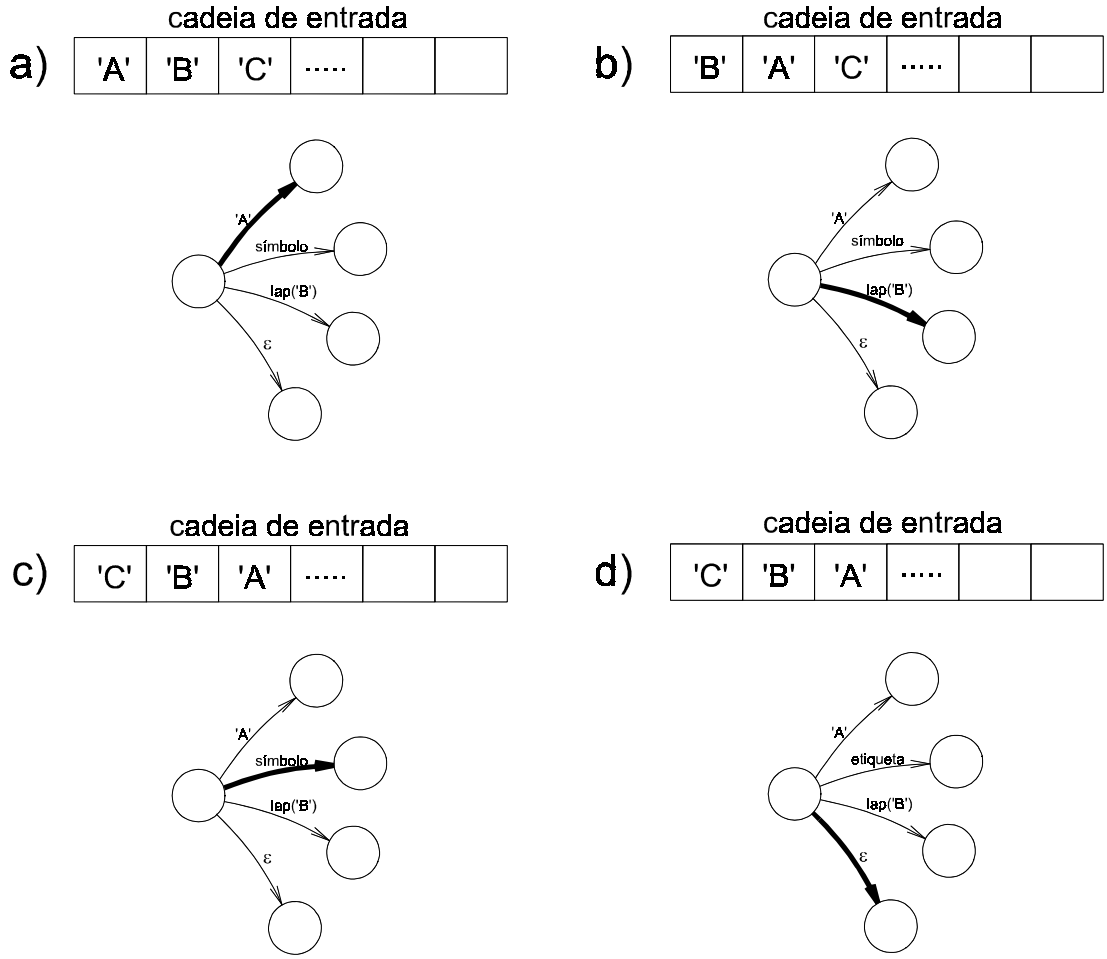


Figura 20 – Determinismo na escolha da próxima transição em um autômato adaptativo E

No item a) da Figura 20, observa-se que, com o caracter "A" na primeira posição da cadeia de entrada, três das quatro possíveis transições seriam elegíveis: a transição com o **estímulo simples** "A", a com o estímulo **símbolo**, e a transição em **vazio**. Pela definição acima, elege-se a transição com o estímulo "A".

No item b), observa-se o caracter "B" na primeira posição da cadeia de entrada. Logo, três das quatro possíveis transições seriam elegíveis: a transição com o estímulo **símbolo**, a com o estímulo **lap**("B"), e a transição em **vazio**. Pela definição acima, a transição com **lap**("B") é eleita.

No item c), observa-se que duas das quatro possíveis transições seriam elegíveis, já que o caracter "C" está na primeira posição da cadeia de entrada: a transição com o estímulo **símbolo** e a em **vazio**. Pela definição acima, elege-se a transição com estímulo **símbolo**.

E no item d), observa-se que, com o caracter "C" na primeira posição da cadeia de entrada, apenas uma das quatro possíveis transições é elegível: a transição em **vazio**. Nota-se que uma transição em **vazio** sempre é elegível, mas só é escolhida quando não há outra opção.

Para ilustrar o funcionamento do interpretador de autômatos adaptativos E, foi idealizada uma aplicação simples: um levantador de léxico, ou seja, um autômato que codifica todo o vocabulário encontrado num corpus (sem quaisquer anotações) no formato de uma árvore de letras, com a contabilização da freqüência de cada palavra no texto.

Como é possível observar através da listagem a seguir, foi preciso definir apenas duas funções adaptativas anteriores para a realização deste experimento. E deve-se ressaltar que o autômato inicial (Figura 21) é extremamente simples, consistindo de dois estados e duas transições. Um resumo da finalidade de cada uma dessas funções adaptativas pode ser encontrado na Tabela 1.

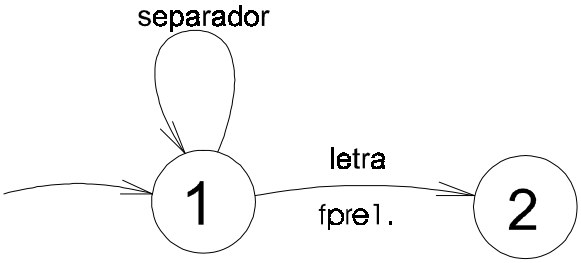


Figura 21 – Autômato inicial referente ao primeiro experimento

Função adaptativa	Tipo	Funcionalidade
fpre1	Anterior	É responsável pelo crescimento da árvore de letras.
fpre2		É responsável pela atualização do contador que existe associado a toda última letra de uma palavra. Assim, é possível contabilizar quantas ocorrências de cada palavra foram encontradas no corpus.

Tabela 1 – Funções adaptativas usadas no levantador de léxico

```
/* Lista de transições do autômato original */
transicao(1, 2, letra, 0, fpre1, nada, 0).
transicao(1, 1, separador, 0, nada, nada, 0).

/* Lista das funções adaptativas */
nada(X,Y,Z):- true.
fpre1(X,Y,Z):-
```

```
      gerador(Novo1),
      gerador(Novo2),
      eliminacao(X,     Y,     letra,     0, fpre1, nada, 0),
      insercao(  X,     Novo1, Z,         0, nada,  nada, 0),
      insercao(  X,     Novo2, letra,     0, fpre1, nada, 0),
      insercao(  Novo1, Y,     letra,     0, fpre1, nada, 0),
      insercao(  Novo1, 1,     separador, 0, fpre2, nada, 0).

fpre2(X,Y,Z):-
      inspecao(A, X, B, C, D, E, N),
/* Parâmetros para inspeção: inspecao(ESTADO ORIGEM, ESTADO DESTINO, ESTÍMULO,
ESCRITA PARA CADEIA DE ENTRADA, FUNÇÃO ADAPTATIVA PRÉ, FUNÇÃO ADAPTATIVA PÓS,
CONTADOR) */
      letra(B),
      eliminacao(A, X, B, C, D, E, N),
      soma_um(N, N2),
      insercao(A, X, B, C, D, E, N2).
```

Será usado o corpus hipotético mostrado abaixo (somente de duas palavras) para simular o comportamento do levantador de léxico. A Figura 22 exemplifica o crescimento do autômato inicial depois de submetido à cadeia de caracteres "GATO " na entrada. Fica claro que o autômato, depois de consumir esta primeira palavra vai parar no estado 1, devido a presença de um espaço em branco após a mesma.

```
GATO  GATUNO
```

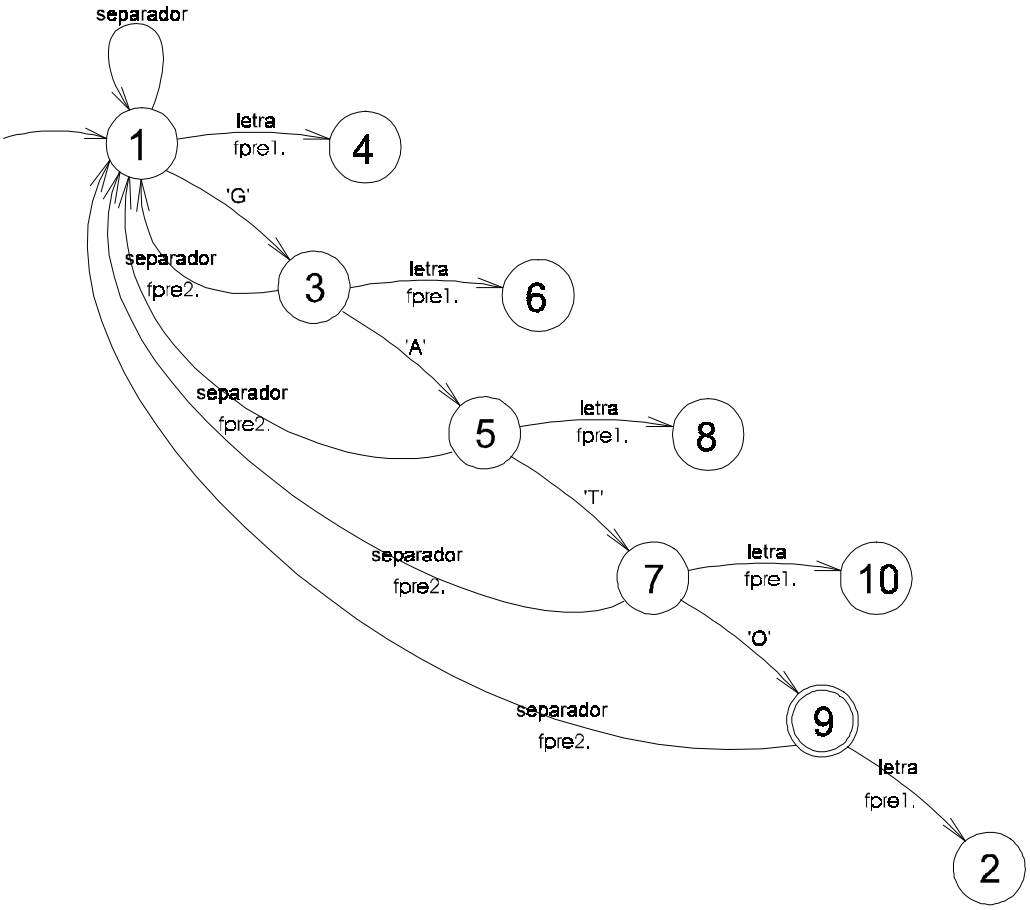


Figura 22 – Exemplo de crescimento do autômato inicial, após a entrada da cadeia de caracteres "GATO "

Às transições foi associado um parâmetro numérico (o último da lista) que tem uma dupla finalidade: indicar se o estado que segue é um estado final (convencionou-se que, se este número for diferente de zero, o estado que suceder a esta transição será um estado final) e contar quantas vezes um estado final foi atingido (indiretamente, este será o número de vezes que a palavra terminada neste estado apareceu até então no texto). Isto foi possível através da extensão do formalismo dos autômatos adaptativos proposta nesta seção.

A Figura 23 dá uma idéia mais clara da topologia que vai sendo assumida pela estrutura de dados montada, à medida que a próxima palavra presente na entrada, "GATUNO ", for consumida: uma árvore de letras.

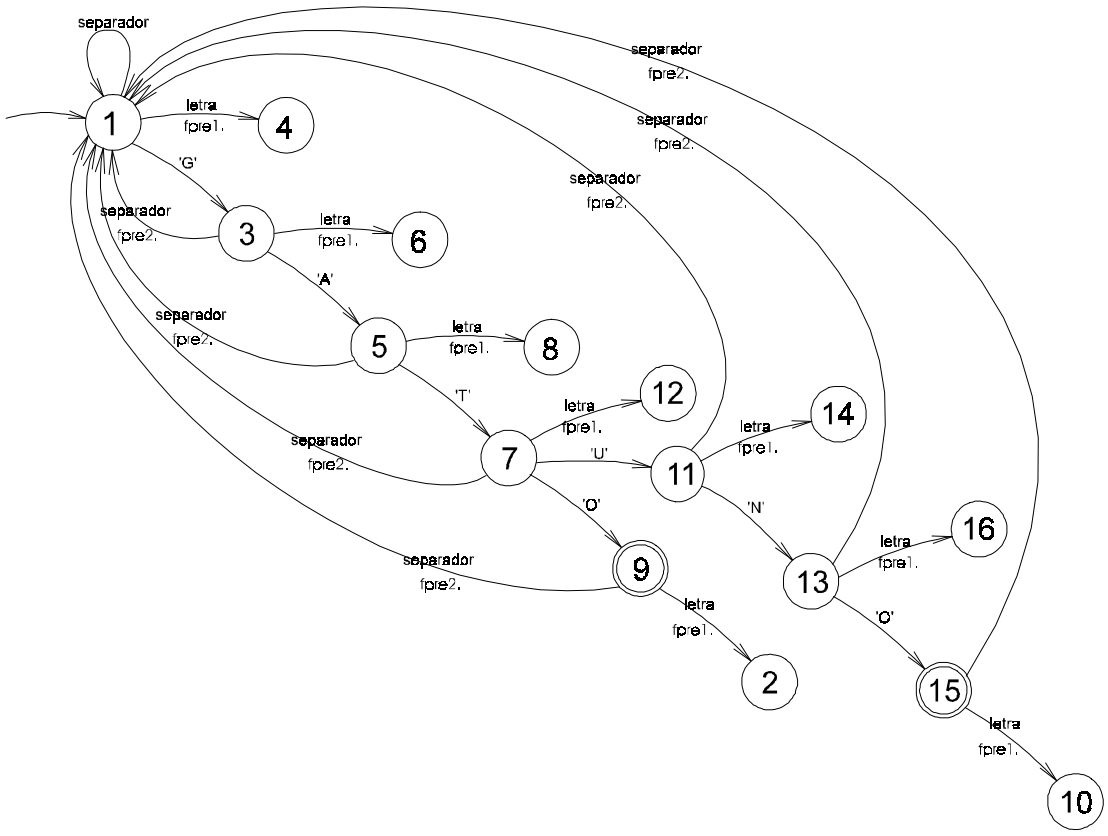


Figura 23 – Exemplo de crescimento do autômato inicial, após a entrada da cadeia de caracteres "GATO GATUNO "

3.2 Especificação do etiquetador morfológico

É possível observar idéias lingüísticas semelhantes em todos os paradigmas de analisadores morfológicos treináveis citados no capítulo 2. Por exemplo, os analisadores morfológicos baseados no paradigma estatístico, no de regras inferidas automaticamente e no de exemplos memorizados utilizam-se de três fontes de informação lingüística, todas extraídas de um corpus de treinamento [BRILL-93, DAELEMANS-96a, CHARNIAK-93, VILLAVICENCIO-95]:

- os sufixos de palavras, como parte do processo de inferência da etiqueta morfológica de palavras desconhecidas;
- uma lista de palavras associadas à categorias morfológicas (léxico), para fornecer informações sobre palavras conhecidas;
- contexto próximo do item lexical que se quer etiquetar (2 ou 3 etiquetas ao redor), para refinar a escolha de sua etiqueta.

Este trabalho propõe, entre outras coisas, um método para a construção de um etiquetador morfológico, que possa ser usado para um número muito grande de línguas (apesar que se propõe testá-lo apenas para a língua portuguesa), que seja treinável com o uso de corpora e que possibilite uma boa precisão na anotação (conforme já definido anteriormente).

Os pré-requisitos para que este etiquetador possa ser treinado numa determinada língua são os seguintes:

- disponibilidade de um corpus com anotações morfológicas;
- as palavras desta língua devem ter uma estrutura do tipo:

  PALAVRA = PREFIXOS + RADICAL + SUFIXOS
- a língua não deve ser AGLUTINANTE (como é o caso do alemão, por exemplo), visto que este método ainda não leva em conta a existência de itens lexicais com mais de um radical.

A arquitetura básica do analisador morfológico treinável proposto neste trabalho segue o publicado por E. Brill, que é uma modelagem em três módulos: o primeiro, que cuida da etiquetação inicial de palavras conhecidas, o segundo, que cuida da etiquetação inicial de palavras desconhecidas, e um terceiro e último, que promove um refinamento contextual (Figura 24) [BRILL-93]. Cada um dos módulos armazenará informações extraídas de um corpus de treinamento, e, a partir destas informações, procederá a etiquetação sem inferir regras, o que está mais próximo da proposta de W. Daelemans [DAELEMANS-96a]. Usar-se-ão autômatos adaptativos (AA) como base de implementação e como estrutura de dados para o armazenamento das informações necessárias a cada módulo [JOSÉ NETO-94]. Desta forma, será demonstrada a viabilidade e avaliada a aplicabilidade dos AA para a modelagem de paradigmas de aprendizado automático.

Assuntos referentes à eficiência da implementação não constituem o foco desta dissertação. Contudo, como um AA pode ser visto como um autômato finito por trechos [JOSÉ NETO-94], e como autômatos finitos podem ser implementados de modo extremamente eficiente [ROCHE-97], pode-se esperar bons resultados, sob este aspecto, de uma implementação que seja formulada com o uso dos AA. Seria

necessário, entretanto, gerar um compilador que suporte a extensão de autômatos adaptativos E, aqui proposta, o que foge ao escopo deste trabalho.

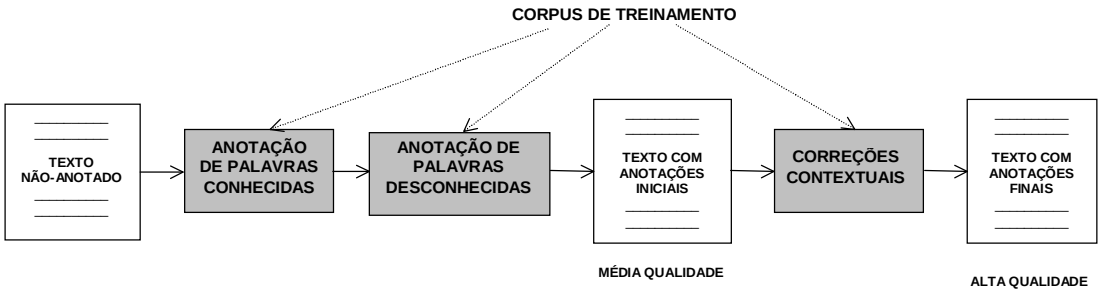


Figura 24 – Visão macroscópica do etiquetador morfológico proposto

3.2.1 Primeiro módulo: obtenção da etiqueta mais provável para as palavras conhecidas

A estrutura de dados concebida como base para este módulo é a de uma árvore *n*-ária de letras, utilizada para armazenar o léxico, contendo uma lista ligada associada a cada uma de suas folhas (a qual representa o final de uma seqüência completa que compõe um item lexical). Esta lista é utilizada para armazenar as várias etiquetas morfológicas possíveis, em ordem decrescente de freqüência de aparecimento. Uma vantagem, inerente a esta estrutura em forma de árvore, é que ocorre naturalmente uma compressão do tamanho da base de dados pelo fato de todos os prefixos serem armazenados apenas uma vez na estrutura.

Após a fase de treinamento, um texto sem anotações pode ser fornecido ao autômato e este determinará as etiquetas mais prováveis, em ordem decrescente de freqüência, para cada palavra que tenha aparecido no corpus de treinamento (ou seja, uma palavra conhecida); caso uma palavra deste texto não tenha aparecido neste corpus (palavra desconhecida), ela não receberá etiqueta alguma.

Foram necessárias cinco funções adaptativas anteriores, duas posteriores e uma auxiliar (que não está associada diretamente a uma transição) para a descrição deste módulo, conforme se pode observar na Tabela 2 e na listagem constante do Apêndice 2.

Função adaptativa	Tipo	Funcionalidade
fpre1	Anterior	É responsável pelo crescimento da árvore de léxico.
fpre2		É responsável pelo aprendizado da primeira etiqueta de uma palavra.
fpre3		É também responsável pelo aprendizado da primeira etiqueta de uma palavra.
fpre4		Armazena uma etiqueta e prepara o autômato para receber uma outra etiqueta, se houver, para a mesma palavra.
fpre5		Função que sugere todas as etiquetas possíveis para uma palavra não etiquetada, caso haja exemplos anteriores. Faz uso da função fx.
fx	Auxiliar	Função que não está associada a qualquer transição, mas que é chamada pela fpre5. Ela serve para resgatar todas as etiquetas possíveis para uma dada palavra, a partir da segunda.
fpos1	Posterior	É responsável pela contabilização das freqüências das etiquetas de cada palavra e pela ordenação destas etiquetas em função da freqüência (ordem decrescente). Trabalha recursivamente.
fpos2		Função que auxilia fpre5.

Tabela 2 – Funções adaptativas usadas no primeiro módulo do etiquetador

Apenas para ilustrar o funcionamento deste módulo, a partir da Figura 25 até a Figura 30, são mostradas representações gráficas do autômato em diversos momentos do processamento, quando este é alimentado por um corpus de treinamento hipotético que só apresenta a palavra "A", algumas vezes etiquetada como ARTIGO, outras como PREPOSIÇÃO e ainda outras vezes como PRONOME, cujo formato segue a expressão regular (PALAVRA (("/" ETIQUETA) | ε) SEPARADOR*)*.

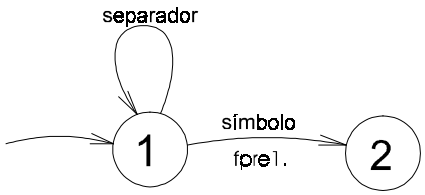


Figura 25 – Autômato inicial referente ao primeiro módulo do etiquetador

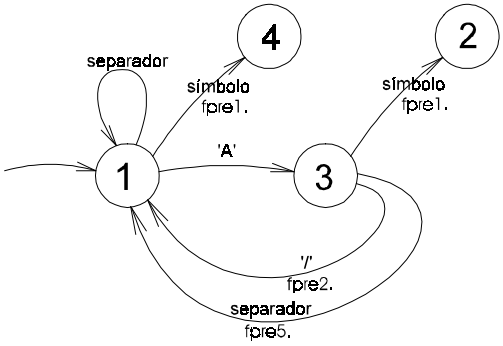


Figura 26 – A primeira palavra "A" foi consumida

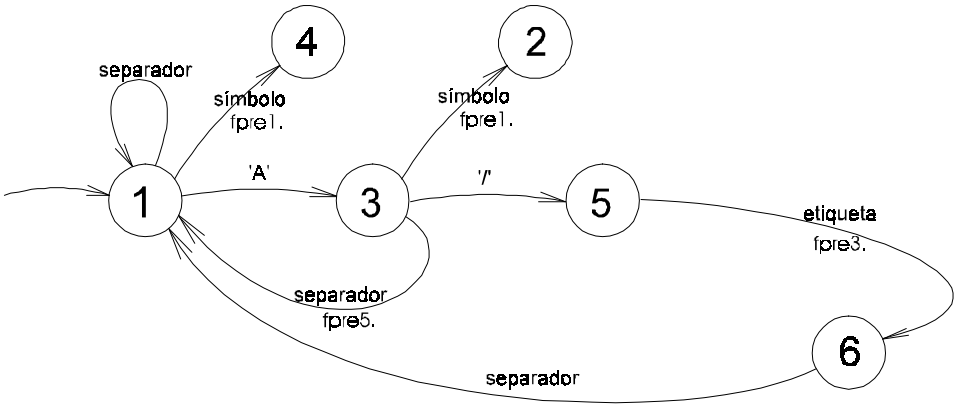


Figura 27 – A palavra "A", seguida de uma barra ("/"), foi consumida

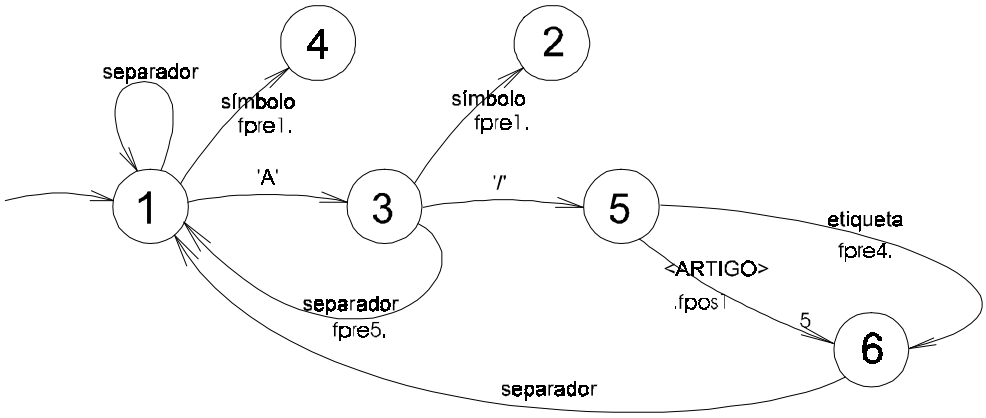


Figura 28 – A palavra "A", etiquetada como artigo, apareceu cinco vezes no corpus de treinamento

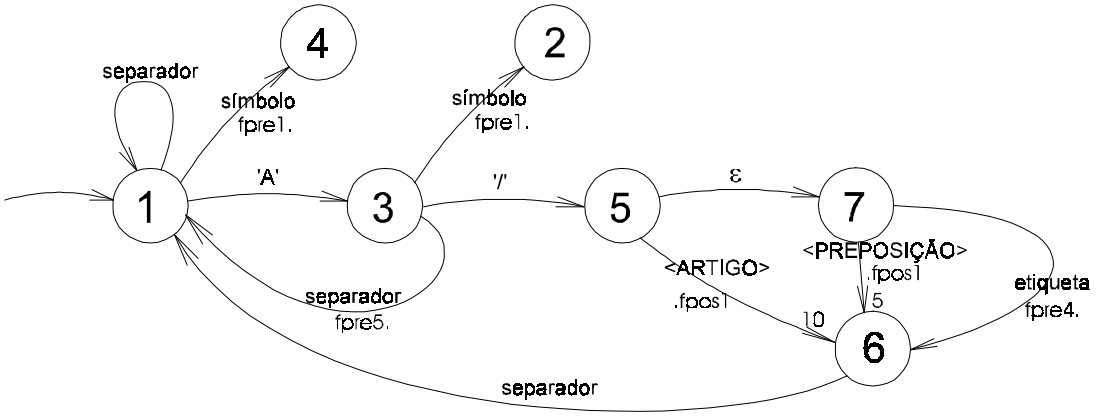


Figura 29 – A palavra "A", etiquetada como artigo, apareceu dez vezes no corpus de treinamento e, etiquetada como preposição, cinco vezes

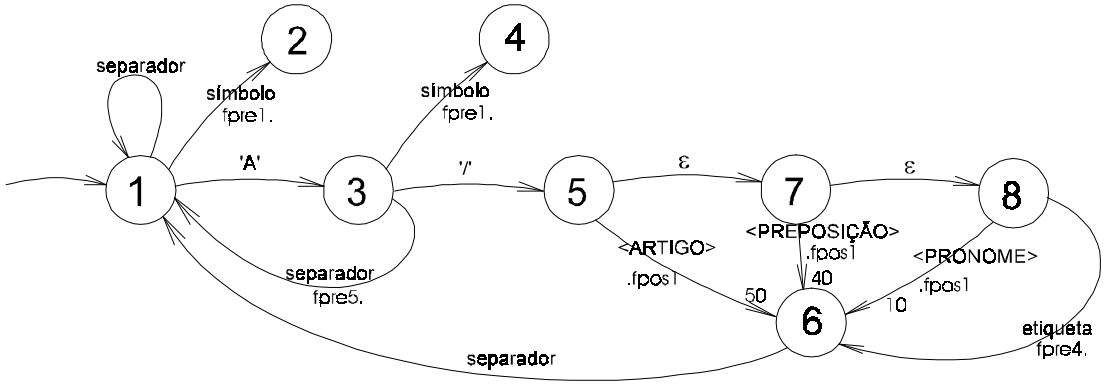


Figura 30 – A palavra "A", etiquetada como artigo, apareceu cinqüenta vezes no corpus de treinamento, etiquetada como preposição, quarenta vezes, e, como pronome, dez vezes

O resultado final, como pode ser observado na Figura 30, não lembra, de fato, a estrutura de uma árvore, visto que todo o léxico de nosso corpus hipotético resume-se a apenas uma palavra. Considerando-se todas as transições deste autômato, inclusive aquelas que processam os separadores do texto de entrada, o autômato, na verdade, é um grafo orientado cíclico.

Se for submetida uma palavra sem anotação ao autômato treinado, e se esta palavra existir no corpus, será sugerida a ela uma lista de anotações, as quais representam todas as anotações possíveis para a mesma, dispostas em ordem decrescente de freqüência de ocorrência, conforme levantado no treinamento efetuado.

Pode-se observar que com este autômato é possível efetuar uma anotação mesmo durante a fase de aprendizagem: se no corpus de treinamento houver palavras não anotadas, a estas serão sugeridas as correspondentes etiquetas mais prováveis. Na verdade, neste módulo não há distinção física entre a fase de treinamento e a de aplicação; estas são coexistentes, sendo assim denominadas por motivos didáticos e conceituais.

3.2.2 Segundo módulo: etiqueta para palavras desconhecidas, com base em sufixos

Com base nas últimas letras dos itens lexicais encontrados no corpus de treinamento e nas etiquetas morfológicas associadas a cada um deles, este módulo

infere um mapeamento que é usado na etiquetação de itens lexicais que nunca apareceram no corpus de treinamento, também ditas palavras desconhecidas.

A heurística por trás deste módulo tem um embasamento lingüístico; é sabido que, nas línguas cujas palavras apresentam a estrutura PREFIXO + RADICAL + SUFIXO, o sufixo de uma palavra tem uma forte correlação com a sua categoria morfológica.

Pode-se ter uma idéia geral do funcionamento deste módulo através da sua arquitetura, mostrada na Figura 33.

Em princípio, deve-se fazer um pré-processamento no corpus de treinamento (veja os elementos INVERSÃO, PODA e ALTERNÂNCIA na Figura 33), para que o mesmo seja reduzido a apenas terminações de palavras com as respectivas etiquetas. Optou-se por reduzir cada item lexical a apenas suas três últimas letras (elemento PODA); é verdade que na língua Portuguesa há sufixos menores e maiores que 3 letras, contudo, a escolha deste número é arbitrária. Nos trabalhos de E. Brill e W. Daelemans, observa-se esta mesma escolha [BRILL-93 e DAELEMANS-96a]. Também este número pode ser facilmente alterado, pois é um parâmetro do pré-processamento.

Deve-se também levar em conta que este módulo propricia uma forma de **extrapolação** quando faz uma comparação de sufixos, ou seja, qualquer comparação a ser feita não necessita ser exata, podendo ser parcial. Assim, uma palavra que tenha o sufixo "mente" receberia a etiqueta ADV (advérbio), mesmo que o módulo de palavras desconhecidas tenha associado somente as últimas três letras "nte" com esta etiqueta.

Optou-se também por ignorar a existência de palavras com até 3 letras para o treinamento deste módulo, visto que ela, por inteiro, seria considerada um sufixo (elemento PODA na Figura 33). A justificativa para esta atitude vem da consideração de um problema que certamente afetará este módulo. É o problema dos **falsos sufixos**: aparecendo o nome próprio "Joana", por exemplo, associado à etiqueta NPR (nome próprio) no corpus de treinamento, o módulo pode inferir que "ana" é um sufixo associado a esta etiqueta.

Também, pode acontecer que o módulo infira o sufixo "ar", associado à etiqueta VB (verbo no infinitivo), e tente etiquetar as palavras (considerando-as desconhecidas) "açúcar" e "par" com esta etiqueta.

De fato, esta heurística é naturalmente susceptível a este fenômeno. Contudo, se além disso ainda for considerado que as palavras com três ou menos letras poderão ser deixadas no corpus de treinamento, mais falsos sufixos serão aprendidos.

Apesar desta argumentação, é apropriado que se faça algum experimento que confirme ou anule esta heurística aqui citada (seção 4.1.1.2).

Abaixo, pode-se visualizar o efeito do pré-processamento em algumas palavras. Os caracteres que formam os sufixos[4] e as etiquetas são armazenados, nesta nova versão do corpus, da direita para a esquerda (elementos INVERTER e ALTERNAR na Figura 33):

PRÉ-PROCESSAMENTO

canta**vam**/VPAS	⇒	**mav**/SAPV
cora**ção**/N	⇒	**oãç**/N
bond**osa**/ADJ	⇒	**aso**/JDA
naturalme**nte**/ADV	⇒	**etn**/VDA
fala**rão**/VFUT	⇒	**oãr**/TUFV

Foram definidas nove funções adaptativas anteriores, cinco posteriores e quatro auxiliares para a descrição deste módulo, conforme se pode observar na Tabela 3 e na listagem constante do Apêndice 2.

Para ilustrar o processo de treinamento, usar-se-á um pequeno corpus hipotético, mostrado abaixo, que é, de fato, um subconjunto do corpus real. Nota-se que este já foi pré-processado.

```
ava/D-BV    ava/D-TE    aso/F-JDA
```

Este corpus traduz os seguintes fatos: que o sufixo "ava" (talvez advindo de palavras como "trabalhava", "estava", por exemplo) pode estar associado às etiquetas "VB-D" e "ET-D", e que o sufixo "osa" (por exemplo, das palavras "carinhosa" e "custosa") pode estar associado à etiqueta "ADJ-F".

[4] Muitas vezes neste texto, o termo **sufixo** é utilizado não com seu sentido estritamente lingüístico, mas significando **terminação de uma palavra**.

A Figura 31 descreve o autômato inicial deste segundo módulo, o qual tem apenas dois estados e duas transições.

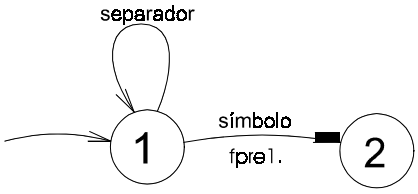


Figura 31 – Autômato inicial do segundo módulo do paradigma

Antes mesmo de a primeira letra do corpus (“a”) ser consumida, a função adaptativa fpre1 é disparada e promove mudanças no autômato original, para que este possa acomodar uma letra (neste caso, a última) de um sufixo, conforme se pode observar na Figura 32 (deve-se notar que, como o corpus está invertido, a primeira letra a ser processada é a última de um sufixo).

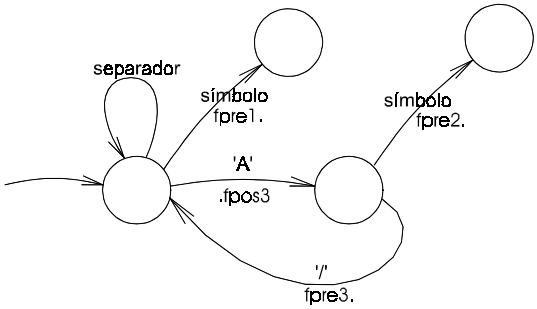


Figura 32 – A letra ‘A’ é consumida (palavra que termina em ‘A’)

Antes da penúltima letra deste sufixo ser consumida, a função fpre2 é disparada, transformando o autômato de acordo com a Figura 34.

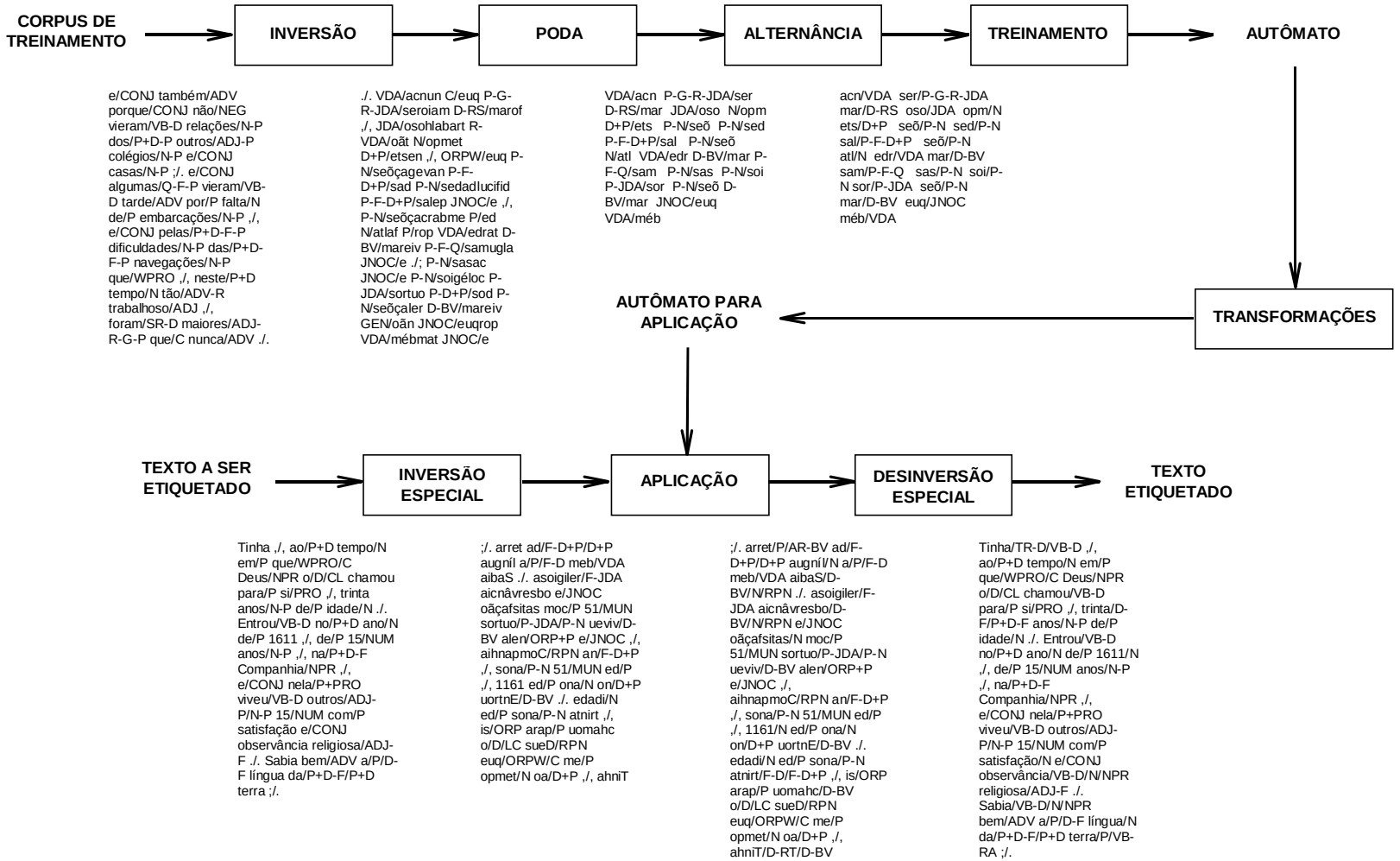


Figura 33 – Arquitetura do segundo módulo do etiquetador

Função adaptativa	Tipo	Funcionalidade
fpre1	Anterior	É responsável pelo aprendizado da última letra de uma série de letras de um sufixo.
fpre2		É responsável pelo aprendizado da penúltima letra e das anteriores (ante-penúltima, etc.) de um sufixo.
fpre3		É responsável pelo aprendizado da primeira etiqueta de um sufixo.
fpre4		É um caso particular da função fpre2.
fpre5		É um caso particular da função fpre6.
fpre6		É responsável pelo aprendizado de uma etiqueta associada a um sufixo, e de preparar o autômato para receber uma outra etiqueta.
fpreAcheMaisProvaveis		É utilizada somente na fase de aplicação. Coloca entre as transições 6 e 7 todas as etiquetas do **sufixo mais longo** encontrado.
fpreAcheMaisProvaveis2		Idem fpreAcheMaisProvaveis. Ela foi definida, juntamente com fposVoltar, para simular uma transição através do estímulo lap(separador).
fpreEscrever		É utilizada somente na fase de aplicação. Coloca as transições existentes entre os estados 6 e 7 no caminho para sejam atravessadas.
fAchePrimeiroVazio	Auxiliar	Função que não está associada a qualquer transição, mas que é chamada pela fpreAcheMaisProvaveis. Ela serve para achar a primeira de uma seqüência de transições em vazio, a partir de um dado estado. Trabalha recursivamente.
fAchePrimeiroExemplo		Função que não está associada a qualquer transição, mas que é chamada pela fpreAcheMaisProvaveis. Ela serve para encontrar no autômato uma transição através do estímulo "bloqueio", ligada ao sufixo que se está analisando, pois esta transição aponta para a melhor classificação para o referido sufixo. Trabalha recursivamente.
fCopieTags		Função que não está associada a qualquer transição, mas que é chamada pela fpreAcheMaisProvaveis. Ela serve para copiar uma seqüência de transições a partir de um determinado estado para um outro estado. Trabalha recursivamente.
fApagueTudo		Função que não está associada a qualquer transição, mas que é chamada pela fposAjeitar e fposAjeitar2. Ela serve para eliminar uma seqüência de transições que se encontram entre dois estados. Trabalha recursivamente.
fpos1	Posterior	É responsável pela contabilização das freqüências das etiquetas de cada sufixo e pela ordenação destas etiquetas em função da freqüência (ordem decrescente). Trabalha recursivamente.
fpos2		É responsável pelo incremento do contador associado a uma transição (transição correspondente a cada letra de um sufixo).
fposAjeitar		É responsável por restaurar o autômato à situação anterior à execução das funções fpreAcheMaisProvaveis e fpreEscrever.
fposAjeitar2		Idem fposAjeitar, só que deixa no caminho a ser atravessado uma transição que escreve no *stream* de saída a etiqueta *default*.
fposVoltar		É responsável por restaurar uma situação provocada por fpreAcheMaisProvaveis2. Ilustrativamente, esta função "destrói a ponte construída após atravessá-la".

Tabela 3 – Funções adaptativas usadas no segundo módulo do etiquetador

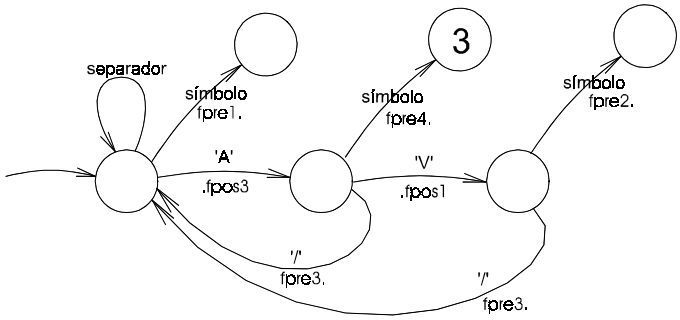


Figura 34 – A letra 'V' é consumida (palavra que termina em "VA")

Pode-se observar na Figura 35 que novamente a função fpre2 será acionada, imediatamente antes do consumo da ante-penúltima letra do sufixo ("a").

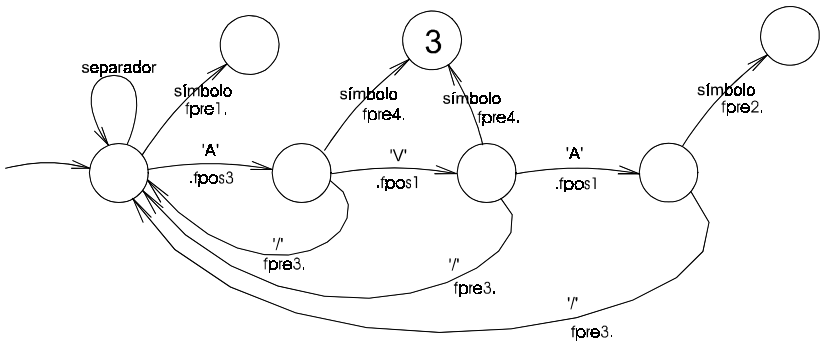


Figura 35 – A letra 'A' é consumida (palavra que termina em "AVA")

A Figura 36 e a Figura 37 descrevem o modo como a etiqueta "VB-D" é associada ao sufixo "ava", através das ações das funções fpre3 e fpre5.

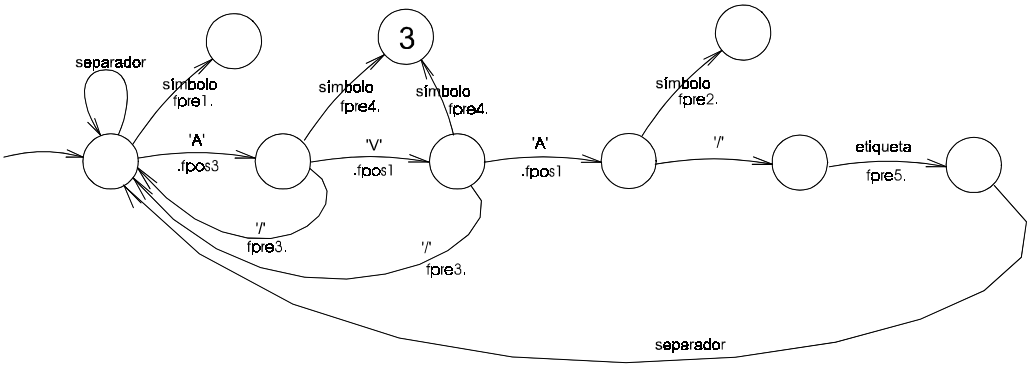


Figura 36 – O símbolo '/' é consumido (em seguida virá uma etiqueta)

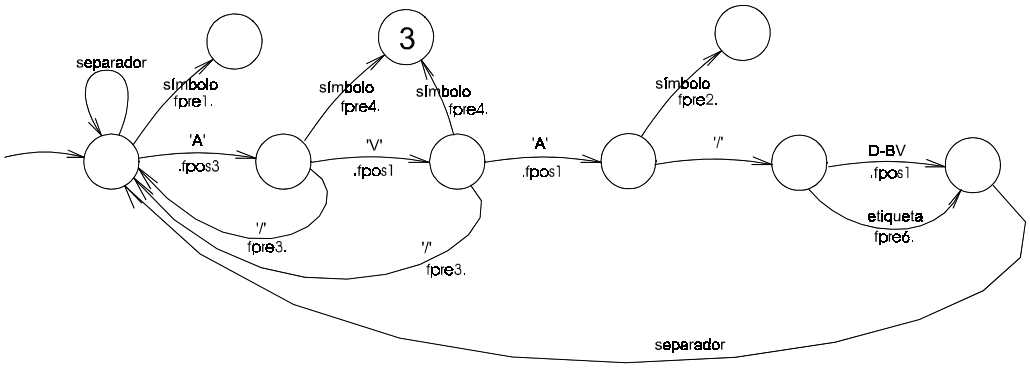


Figura 37 – A etiqueta "VB-D" é consumida (associada à palavra que termina em "AVA")

A seguir, o autômato processará o segundo sufixo do corpus (também "ava") com sua etiqueta "ET-D" (Figura 38). A expansão do autômato para acomodar este segundo exemplo de etiqueta do mesmo sufixo ocorre através da função adaptativa fpre6.

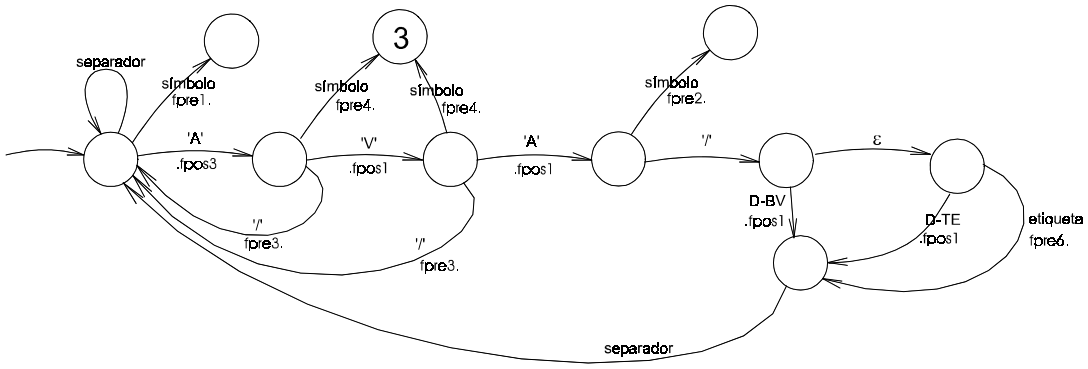


Figura 38 – Outra palavra é encontrada no corpus de treinamento que termina em "AVA" e que é associada à etiqueta "ET-D"

O crescimento do autômato, neste ponto, poderia acontecer através de uma dentre três funções adaptativas diferentes (fpre1, fpre4 ou fpre2), como pode ser observado na Figura 38, dependendo se o próximo sufixo não compartilha qualquer letra com os sufixos já armazenados (é o caso de fpre1), se este compartilha alguma letra, mas não todas, com um dos sufixos armazenados (fpre4) ou se compartilha todas as letras de um sufixo (é o caso de fpre2, que só faria sentido se houvesse algum sufixo com mais de três letras).

Em nosso exemplo hipotético, o terceiro sufixo presente no corpus de treinamento é "osa". Nota-se que sua última letra ("a") é comum a última do prefixo "ava" já armazenado; portanto, este sufixo começará a ser incorporado pelo autômato através da função fpre4.

Da Figura 39 até a Figura 42 vê-se, passo-a-passo, o modo pelo qual o autômato aprende mais este exemplo.

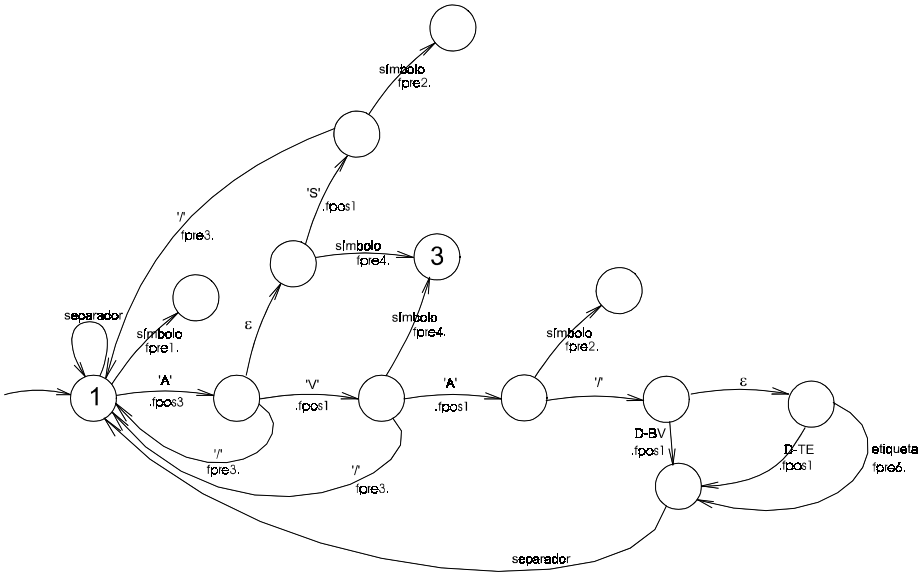


Figura 39 – As letras 'A' e 'S' são consumidas (palavra que termina em "SA")

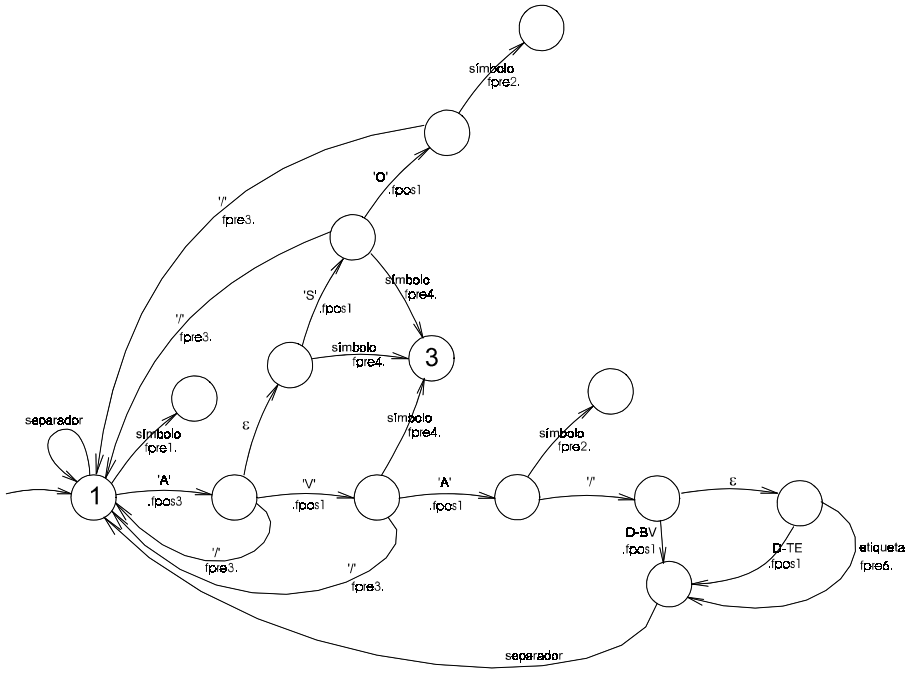


Figura 40 – A letra 'O' é consumida (palavra que termina em "OSA")

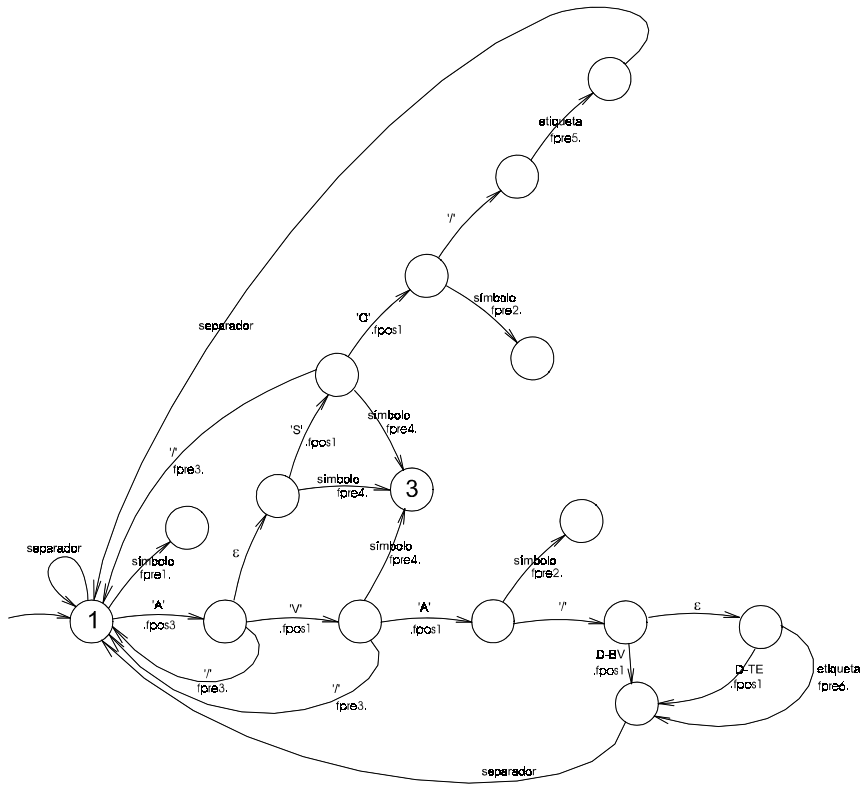


Figura 41 – O símbolo '/' é consumido (em seguida virá uma etiqueta associada ao sufixo "OSA")

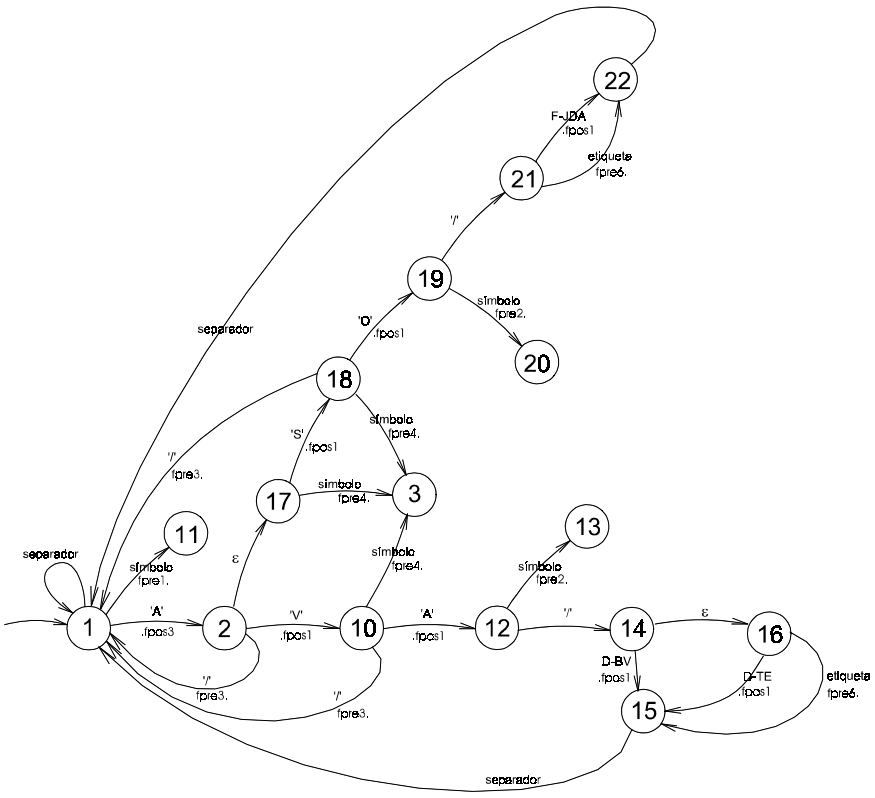


Figura 42 – A etiqueta "ADJ-F" é consumida (associada à palavra que termina em "OSA")

Pode-se observar na Figura 42 que todos os possíveis sufixos e seus relacionamentos com as correspondentes categorias morfológicas são automaticamente inferidos, gerando-se assim mapeamentos entre os sufixos e as etiquetas morfológicas mais prováveis das palavras correspondentes. Este mapeamento é codificado num autômato adaptativo que tem a estrutura de uma árvore *n*-ária de letras, para armazenar o sufixo, com uma lista ligada em cada folha desta árvore, para armazenar as várias etiquetas morfológicas possíveis, em ordem decrescente de freqüência de aparecimento no corpus de treinamento.

Dos três módulos, este é o único em que há uma real distinção entre a fase de aprendizagem (ou treinamento) e a de aplicação. Nos demais, os autômatos trabalham indiferentemente nos dois modos.

Por exemplo, no primeiro módulo, quando o autômato encontra na cadeia de entrada uma palavra seguida de uma etiqueta, ele incorpora esta seqüência como um exemplo. Por outro lado, quando uma palavra sem etiqueta é encontrada, o autômato se encarrega de sugerir a etiqueta mais provável, se, é claro, houver algum exemplo com esta palavra armazenado no autômato.

Para que o autômato do segundo módulo possa ser usado no processo de etiquetação, é proposto um conjunto de transformações neste que podarão as transições ligadas ao seu crescimento (processo de aprendizagem) e acrescentarão novas. Estas novas transições encarregar-se-ão de encontrar a etiqueta mais provável para o respectivo sufixo (Figura 33).

Abaixo, vê-se uma listagem com o conjunto das transformações propostas para o autômato, depois de treinado.

```
transformacoes:-
        substitue1,
        substitue2,
        substitue3,
        substitue4,
        eliminacao(1, _, simbolo,   0,        fpre1,        nada,         0),
        insercao(  1, 3, simbolo,   0,        nada,         nada,         0),
        insercao(  3, 3, simbolo,   0,        nada,         nada,         0),
        insercao(  3, 1, separador, 0,        fpreEscrever, nada,         0),
        insercao(  3, 4, barra,     0,        nada,         fposAjeitar2, 0),
        insercao(  4, 5, etiqueta,  0,        nada,         nada,         0),
        insercao(  5, 1, separador, 0,        nada,         nada,         0),
        insercao(  6, 7, vazio,     ["/N"],   nada,         nada,         0).
```

Os quatro conjuntos de substituições (substitue1 a substitue4) podem ser observados graficamente na Figura 43 e significam o seguinte:

- **substitue1**: substitua todas as transições com estímulo “barra” e sem quaisquer funções adaptativas por duplas de transições, tal que uma das transições da dupla seja idêntica à substituída, a menos do estímulo, que passa a ser “bloqueio”, e a outra também seja idêntica à substituída, a menos do estado-destino, que passa a ser 4.
- **substitue2**: substitua todas as transições com estímulo “símbolo” e função adaptativa anterior fpre2, porém sem função adaptativa posterior, por duplas de transições, de tal forma que ambas as transições da dupla passam a ter o mesmo estado-origem da substituída, e estado destino 3; uma delas terá por estímulo “símbolo” e função adaptativa anterior fpreAcheMaisProvaveis, e a outra terá “separador” por estímulo e fpreAcheMaisProvaveis2 por função adptativa anterior; ambas as transições não possuirão função adaptativa posterior.
- **substitue3**: substitua todas as transições com estímulo “símbolo”, função adaptativa anterior fpre4, estado destino 3 e sem função adaptativa posterior, por duplas de transições com o mesmo estado-origem e estado-destino da transição substituída; uma delas terá por estímulo “símbolo” e função adaptativa anterior fpreAcheMaisProvaveis, e a outra terá “separador” por estímulo e fpreAcheMaisProvaveis2 por função adptativa anterior; ambas as transições não possuirão função adaptativa posterior.
- **substitue4**: substitua todas as transições com estímulo “barra”, função adaptativa anterior fpre3, estado destino 1 e sem função adaptativa posterior, por transições com estado-origem idêntico ao da substituída, só que com estado-destino 4, estímulo “barra” e sem quaisquer funções adaptativas.

As outras etapas da transformação são simples de entender, pois tratam-se de apenas uma eliminação e algumas inserções de transições. O resultado pode ser visto na Figura 44.

O estímulo “bloqueio”, usado apenas aqui, constitue, na realidade, um artifício. Nada que possa existir na cadeia de entrada corresponde a “bloqueio”, e isso é proposital. O objetivo neste ponto é justamente impedir que algumas transições sejam executadas, podendo ser apenas inspecionadas.

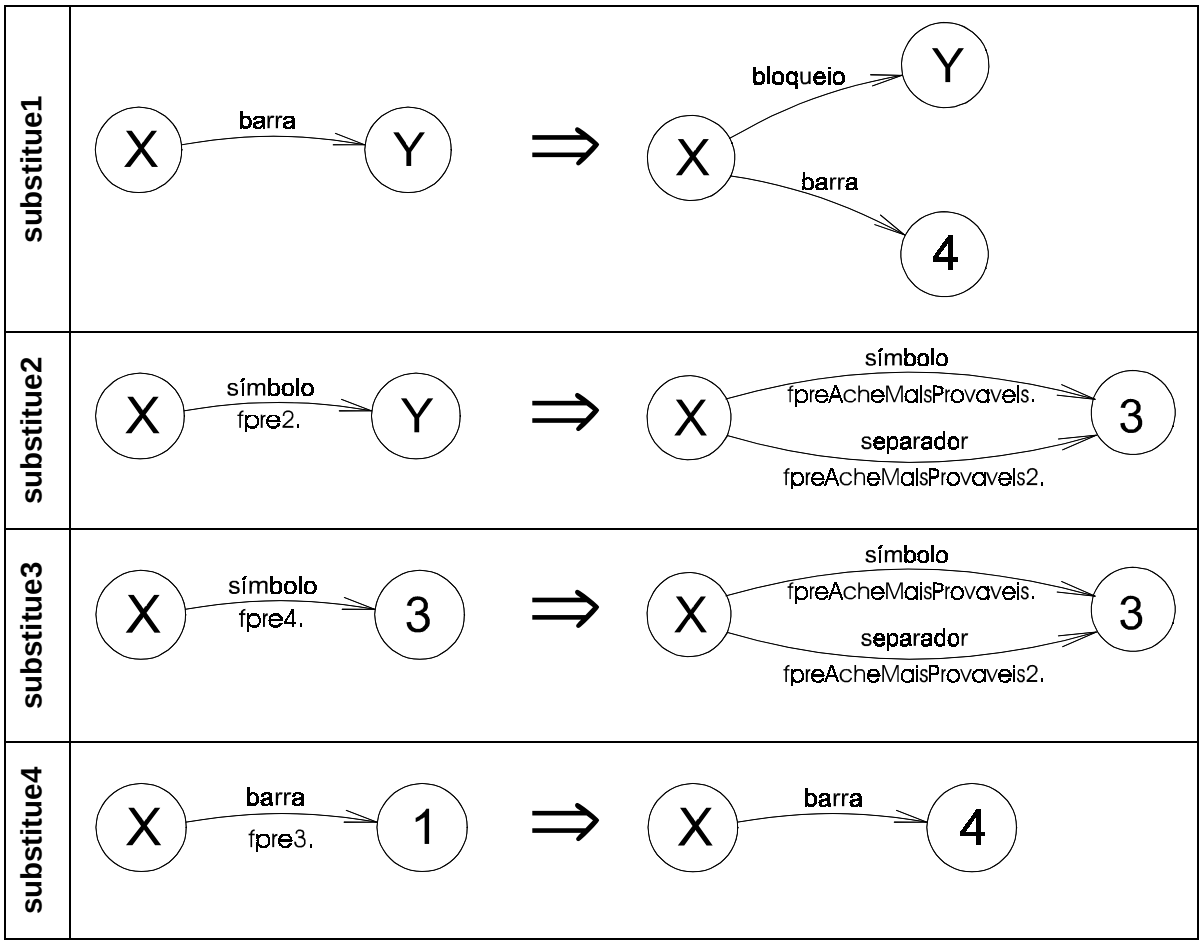


Figura 43 – Conjunto de substituições feitas após a etapa de treinamento

De fato, é somente neste módulo que se vê uma separação um pouco mais clara entre código e dados, que não existe nos outros módulos. Todas as transições que ficaram isoladas pelas transições com estímulo "bloqueio" constituem apenas dados. As outras, que são realmente executadas, formam uma mistura de código e dados.

Assim que essas transformações forem efetuadas, o autômato já poderá ser usado para o processo de etiquetação. Seguem algumas simulações que demonstram a habilidade de **generalização** ou **extrapolação** presentes neste autômato (todas as referências a estados dizem respeito ao autômato da Figura 44):

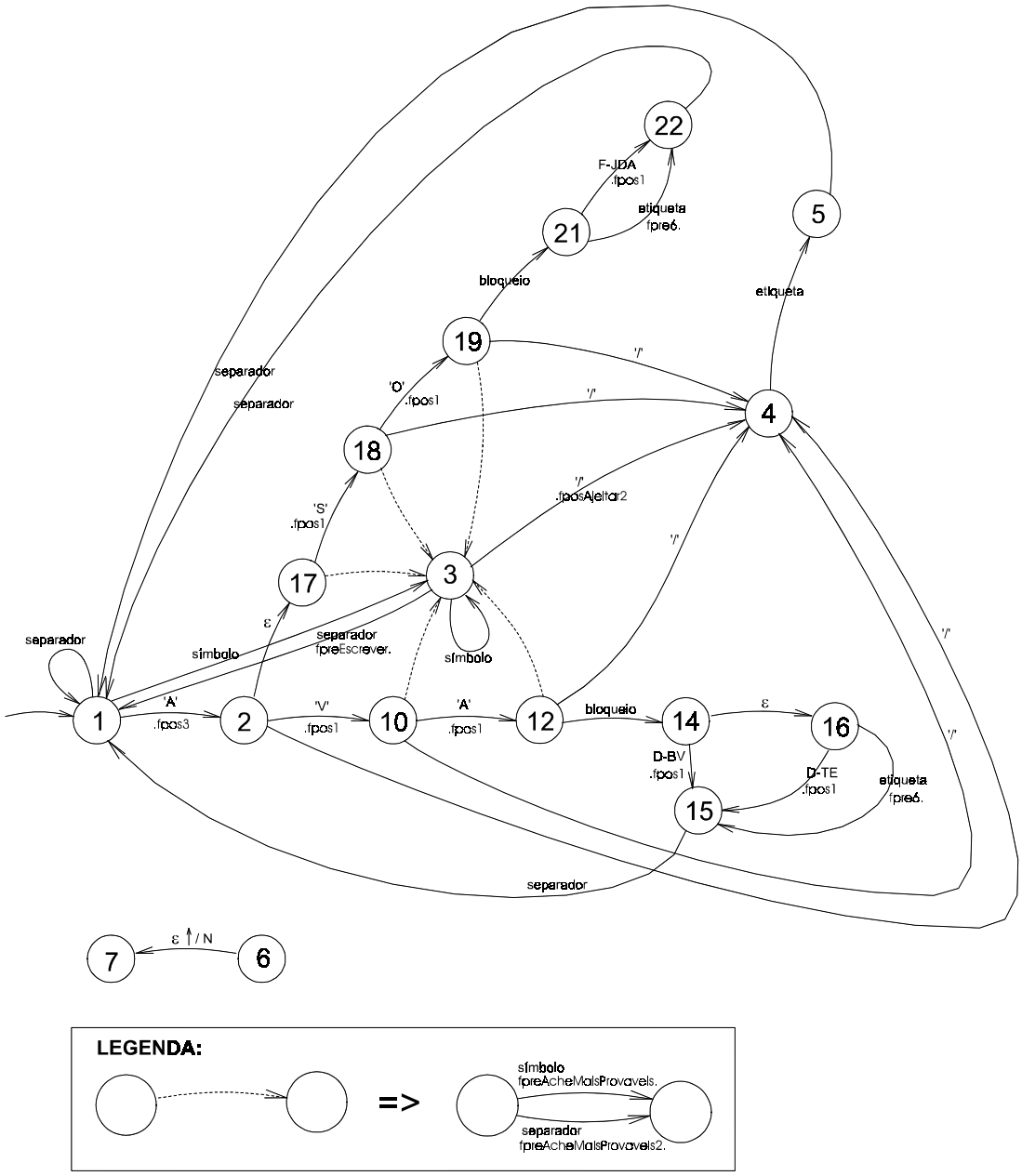


Figura 44 – Autômato após treinamento e transformações

- Suponha-se que a palavra hipotética "XXXXXYA" está para ser etiquetada, sendo ela desconhecida; o autômato receberá a mesma invertida, ou seja, "AYXXXXX" (veja o elemento INVERSÃO ESPECIAL na Figura 33). O autômato evoluirá do estado 1 para o 2, consumindo a letra "A" (a função adaptativa fpos3 só servirá para incrementar o contador associado a esta transição); do estado 2, a única opção é transitar em vazio para o estado 17, e deste para o estado 3 com estímulo símbolo (a letra "Y" foi consumida), sendo que a função fpreAcheMaisProvaveis é executada antes desta última transição, enquanto o estado-origem ainda é o 17. O

efeito da execução desta função pode ser visto entre os estados 6 e 7 da Figura 45; considerou-se que VB-D e ET-D eram as etiquetas mais prováveis para a palavra em questão, visto estas serem as de maior probabilidade para palavras terminadas pela letra "A" (como o restante do sufixo era desconhecido, não foi levado em consideração), e isto foi codificado em transições em vazio que escrevem na cadeia de entrada as etiquetas encontradas. As próximas cinco transições serão para o próprio estado 3 por símbolo, até que todas as letras restantes da palavra sejam consumidas. Então, como certamente existirá um separador após a palavra (um espaço, um *line feed* ou um *carriage return*, por exemplo), será tentada uma transição para o estado 1 por separador (veja Figura 44), porém, a função fpreEscrever é executada primeiro e ela elimina esta transição do estado 3 para o 1, inserindo uma espécie de desvio para que as transições entre os estados 6 e 7 sejam atravessadas, conforme a Figura 45. Desta forma, o autômato seguirá do estado 3 para o 6, em vazio, e sucessivamente pelas outras quatro transições em vazio, sendo que as duas primeiras destas escreverão as etiquetas mais prováveis no *stream* de saída, passando pelo estado 7, até chegar ao estado 1 novamente. Após a última destas transições ter sido percorrida, a função fposAjeitar é disparada e reverte todas as alterações que a função fpreEscrever promoveu no autômato.

- Suponha-se agora que a palavra hipotética "XXXXYSA" está para ser etiquetada, sendo ela também desconhecida; o autômato receberá a mesma invertida, ou seja, "ASYXXXX" (elemento INVERSÃO ESPECIAL na Figura 33). O autômato evoluirá do estado 1 para o 2, consumindo a letra "A"; do estado 2, a única opção é transitar em vazio para o estado 17, deste para o 18, consumindo a letra "S", e deste para o estado 3 com estímulo símbolo (a letra "Y" foi consumida), sendo que a função fpreAcheMaisProvaveis é executada antes desta última transição, enquanto o estado-origem ainda é o 18. O efeito da execução desta função pode ser visto entre os estados 6 e 7 da Figura 46; considerou-se que ADJ-F era a etiqueta mais provável para a palavra em questão porque, segundo este critério, esta é a etiqueta com maior probabilidade para palavras terminadas pelas letras "SA" (como o restante do sufixo era desconhecido, não foi levado em consideração), e isto foi codificado em uma transição em vazio que escreve na

cadeia de entrada esta etiqueta. As próximas quatro transições serão para o próprio estado 3 com estímulo símbolo, até que todas as letras restantes da palavra sejam consumidas. Para consumir o separador que certamente existirá um após a palavra, será tentada uma transição para o estado 1 por separador (Figura 44), porém, a função fpreEscrever é executada primeiro e ela elimina esta transição do estado 3 para o 1, inserindo um desvio para que as transições entre os estados 6 e 7 sejam atravessadas (Figura 46). Desta forma, o autômato seguirá do estado 3 para o 6, por vazio, e sucessivamente pelas outras três transições em vazio, sendo que a primeira escreverá a etiqueta mais provável na cadeia de entrada, passando pelo estado 7, até chegar ao estado 1 novamente. Após a última destas transições ter sido percorrida, a função fposAjeitar é disparada e reverte todas as alterações que a função fpreEscrever promoveu no autômato.

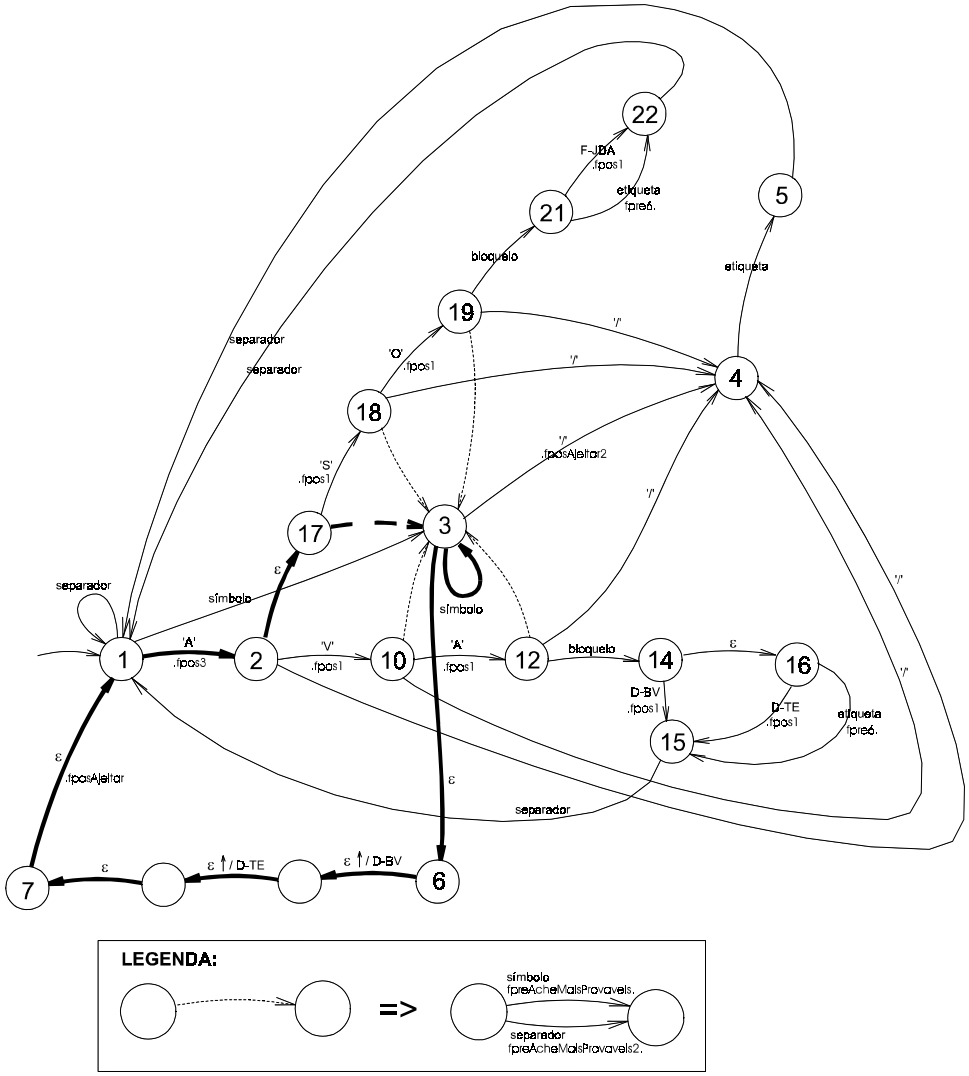


Figura 45 – Autômato durante a fase de aplicação na palavra hipotética "XXXXXYA"

Para que seja possível ter o corpus anotado em um formato legível, tornou-se necessário um pós-processamento para a saída deste autômato, com o objetivo de desfazer as inversões realizadas anteriormente. Isto pode ser observado, macroscopicamente, no elemento DESINVERSÃO ESPECIAL da Figura 33.

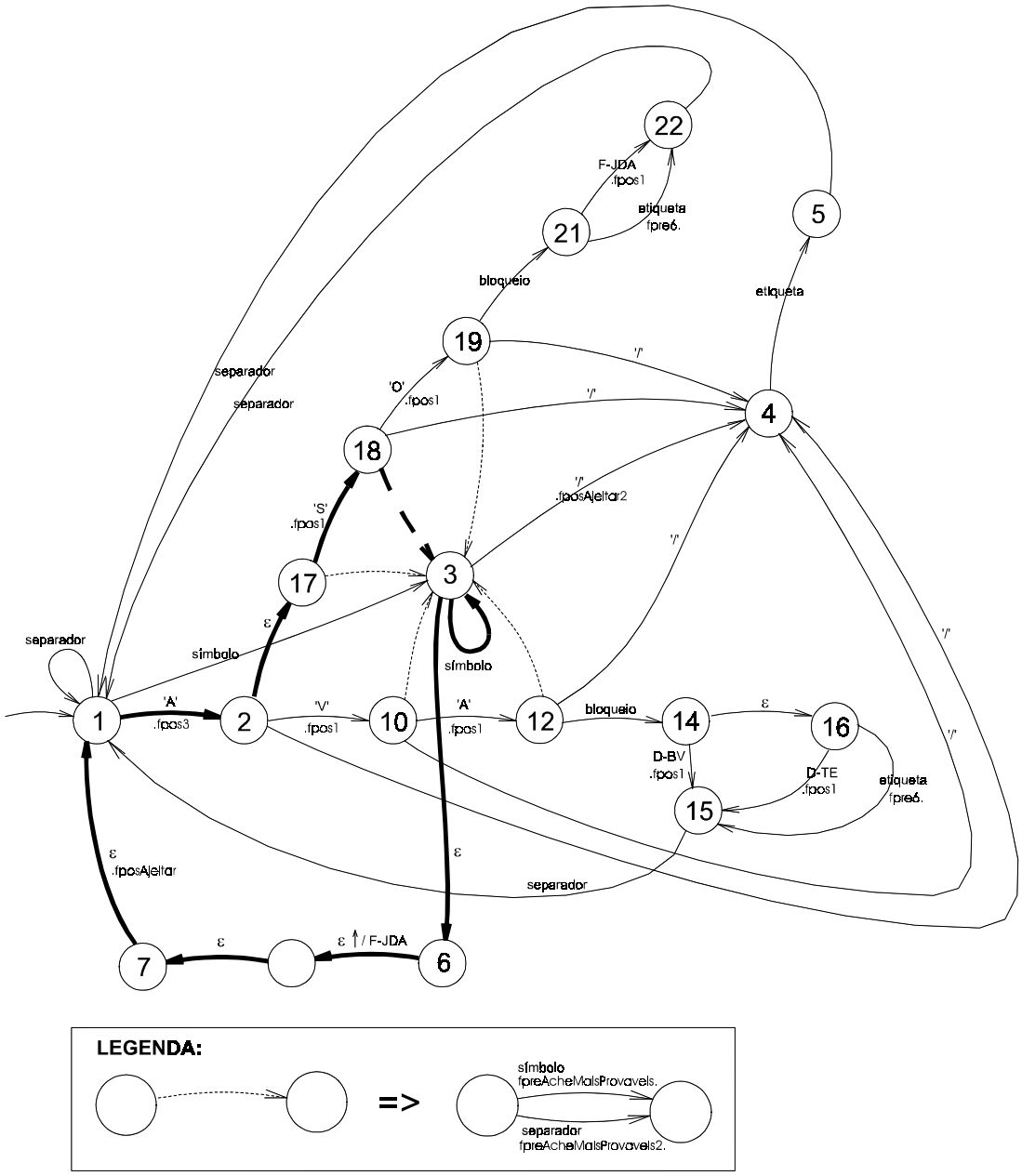


Figura 46 – Autômato durante a fase de aplicação na palavra hipotética "XXXXYSA"

3.2.3 Terceiro módulo: refinador contextual

As duas fases anteriores (primeiro e segundo módulos) dizem respeito a informações meramente lexicais extraídas de um corpus: o módulo de palavras conhecidas promove a construção de um léxico, no qual há uma lista de categorias morfológicas associada a cada item lexical (esta lista é ordenada pela freqüência de ocorrência destas etiquetas associadas ao respectivo item lexical); e o módulo de

palavras desconhecidas armazena sufixos e relaciona-os a categorias morfológicas (ordenadas segundo o mesmo critério existente para o primeiro módulo).

Já este terceiro módulo serve como um refinamento do serviço que os dois primeiros prestam. Ele é responsável por escolher, dentre as várias etiquetas possíveis para uma dada palavra, aquela que mais se adapte ao contexto em que esta palavra se encontra, na ocasião.

Este módulo é treinado com base em informações referentes à seqüência relativa em que se encontram as etiquetas no corpus de treinamento. Desta forma, propõe-se a criação de uma nova versão do corpus de treinamento, contendo apenas as etiquetas do corpus original (Figura 48). Abaixo, encontra-se uma pequena amostra deste:

```
P SR ADV-R CONJ ADV-R VB-AN D N P+D NPR , CONJS D-F-P N-P CL VB-D Q-F-P ADJ-G-P
N-P , NEG CL VB-R SENÃO P+D-F N P N-P . CONJ ADV P N VB-D N-P P ADJ-P N-P ADJ-R-
G-P . P+D-F N CONJ N SR-D ADJ , CONJ , P P P N SR D-UM N P N ADJ-F , CONJS VB-D
VB ADV VB-D+SE+CL P+D N D-UM ADJ-G N , VB-D+SE CONJS D-UM-F N , Q SE VB-D .
```

É esta nova versão do corpus de treinamento que será usada, de fato, para a inferência de informações contextuais pelo módulo refinador.

A Figura 47 mostra o autômato inicial usado neste terceiro módulo do etiquetador. Este autômato consiste de apenas duas transições e três estados.

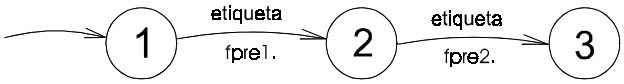


Figura 47 – Autômato inicial usado no terceiro módulo do etiquetador

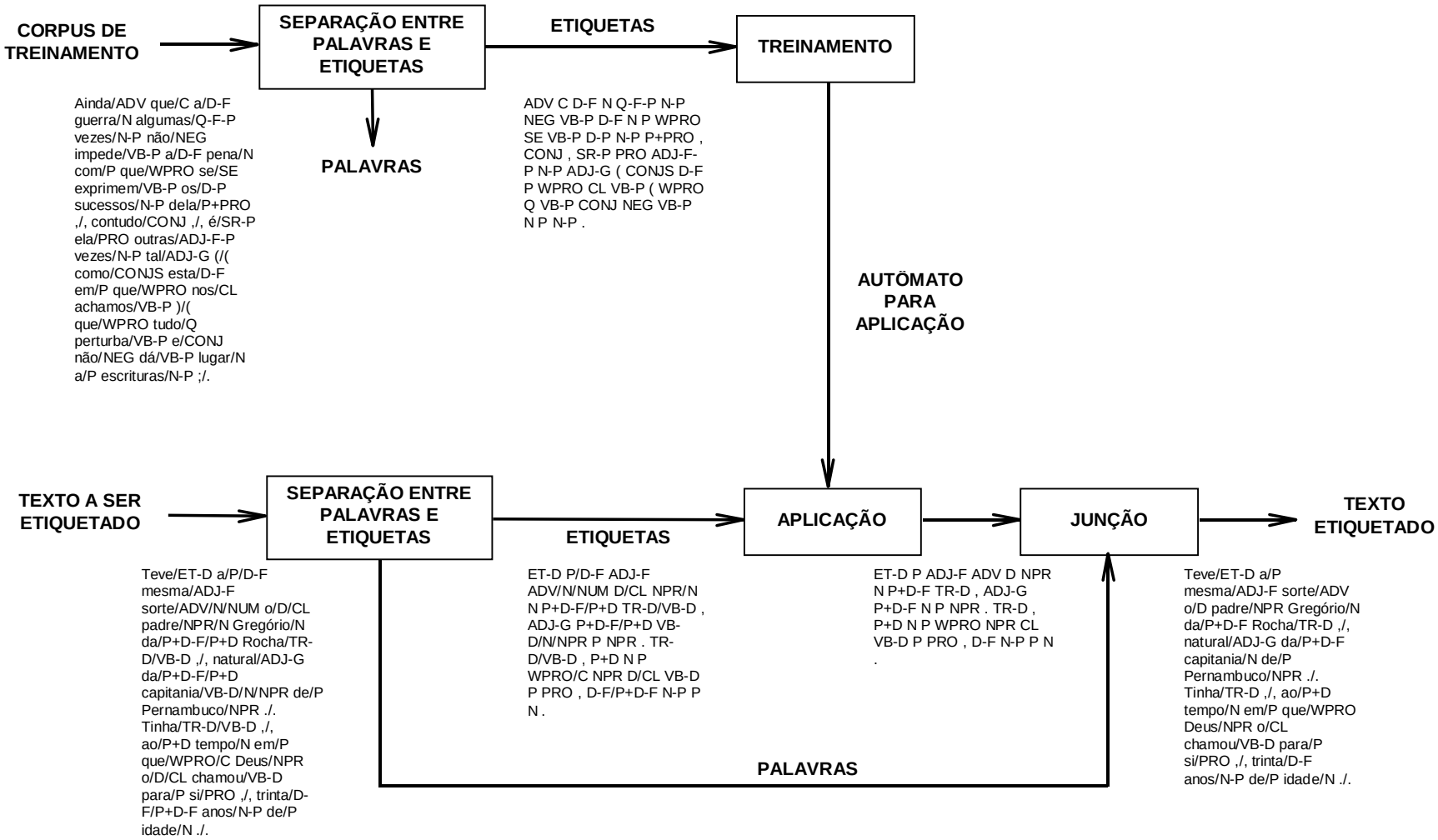


Figura 48 – Arquitetura do terceiro módulo do etiquetador

Sete funções adaptativas anteriores, quatro posteriores e seis auxiliares foram utilizadas para modelar o refinador contextual (Tabela 4 e listagem presente no Apêndice 2).

Este autômato armazenará informações correspondentes a contextos formados por trios ou trigramas de etiquetas. Supor-se-á o corpus hipotético abaixo para ilustrar a heurística de treinamento adotada.

```
P   SR   ADV-R   CONJ   ADV-R    ....    P   N   VB-D    ....    P   N   SR
```

A idéia central do método baseia-se na utilização de uma janela de três posições. Percorre-se com ela a seqüência de etiquetas previamente extraída do corpus de treinamento; a primeira posição da janela refere-se a uma etiqueta já consumida anteriormente, mas que é memorizada; a segunda, refere-se à etiqueta que está sendo consumida na ocasião, e a terceira, à etiqueta seguinte, que é apenas consultada, sem ser consumida (um *look-ahead*), conforme ilustrado na Figura 49.

Esta janela desloca-se um passo por vez, sendo que a cada passo da janela, o correspondente trigrama é considerado.

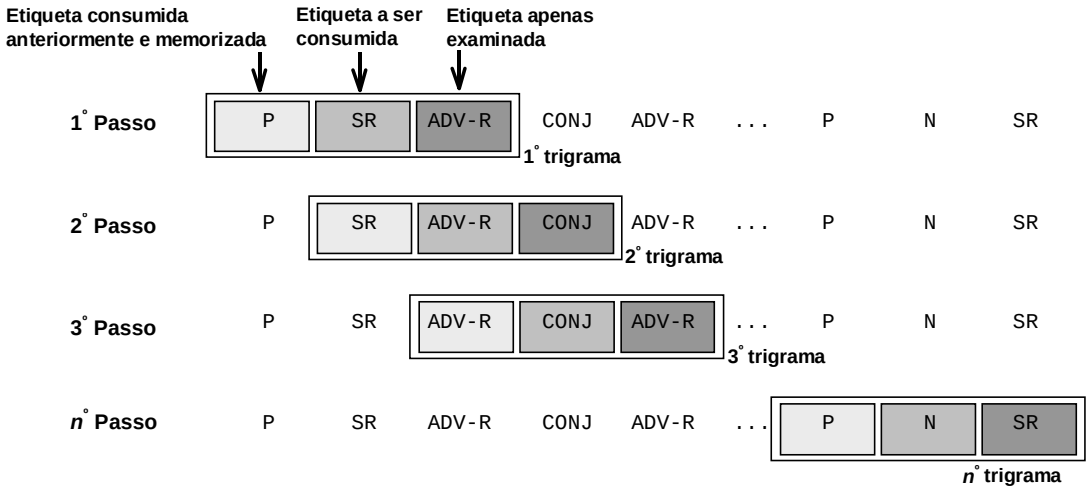


Figura 49 – Trigramas, representados pelas janelas, sendo armazenados durante treinamento

Função adaptativa	Tipo	Funcionalidade
fpre1	Anterior	É responsável pelo aprendizado da primeira etiqueta de um trigrama.
fpre2		É responsável pelo aprendizado da segunda etiqueta de um trigrama.
fpre3		É responsável pelo aprendizado da terceira etiqueta de um trigrama.
fpre4		Esta função é um caso particular da fpre2. É responsável pelo aprendizado de uma outra etiqueta na segunda posição de um trigrama.
fpre5		Esta função é um caso particular da fpre3. É responsável pelo aprendizado de uma outra etiqueta na terceira posição de um trigrama.
fpre6		Esta função é usada somente no processo de Aplicação do conhecimento adquirido. Pesquisa a opção mais provável para tirar a ambigüidade de um grupo de etiquetas que são todas possíveis, através do contexto próximo (etiqueta da palavra anterior, já sem ambigüidades, e conjunto de etiquetas possíveis para a palavra seguinte).
fpre7		Simula uma transição pelo estímulo lap(conj_tag), ou seja, a transição ocorre se o estímulo for um conjunto de etiquetas, mas este conjunto não é consumido.
fAchePrimeiro	Auxiliar	Esta função é chamada pela fpre6. Ela serve para encontrar a primeira de uma seqüência de transições em vazio. Trabalha recursivamente.
fMonteTransicoes		Esta função é chamada pela fpre6. É responsável por montar a estrutura usada na aplicação do conhecimento adquirido, que vai determinar qual das etiquetas ambíguas é a melhor para ser usada no referido contexto. Trabalha recursivamente.
fCopieTransicoes		Esta função é chamada pela fMonteTransicoes. Ela serve para copiar uma seqüência de transições a partir de um determinado estado para um outro estado. Trabalha recursivamente.
pertence		Esta função é chamada pela fMonteTransicoes. Determina se uma etiqueta pertence ou não a um conjunto de etiquetas. Trabalha recursivamente.
fOpcaoDefault		Esta função é chamada pela fMonteTransicoes. Durante a montagem da estrutura usada na aplicação do conhecimento adquirido, esta função serve para adotar, como última opção de etiquetagem, a etiqueta que teve maior freqüência para a referida palavra em um dos módulos anteriores (o de palavras conhecidas ou o de palavras desconhecidas).
fApague		Esta função é chamada pela fposRemoveTudo. Elimina uma seqüência de transições, as quais constituem parte daquela estrutura temporária usada na aplicação do conhecimento adquirido. Trabalha recursivamente.
fpos1	Posterior	É responsável pela contabilização das ocorrências das etiquetas que estão na segunda e terceira posições em um trigrama e pela ordenação destas em ordem decrescente de freqüência. Trabalha recursivamente.
fpos2		Função que auxilia fpre7.
fpos3		É responsável pela contabilização das ocorrências das etiquetas que estão na primeira posição em um trigrama. Ela apenas incrementa o contador associado a uma transição.
fposRemoveTudo		É responsável por remover a estrutura, que foi construída por fMonteTransicoes, e que foi usada na aplicação do conhecimento adquirido.

Tabela 4 – Funções adaptativas usadas no terceiro módulo do etiquetador

Durante o primeiro passo, são encontradas as etiquetas P SR ADV-R. A Figura 50 ilustra a estrutura na qual a heurística proposta neste trabalho armazena os trigramas. No exemplo da referida figura, as duas primeiras etiquetas foram consumidas, enquanto que a terceira foi apenas examinada.

A Figura 51 ilustra o crescimento do referido autômato na ocasião em que a etiqueta ADV-R for consumida e a etiqueta CONJ for examinada (segundo passo); a Figura 52 apresenta o mesmo quando a quarta etiqueta (CONJ) for, de fato, consumida e uma próxima (ADV-R), examinada (terceiro passo).

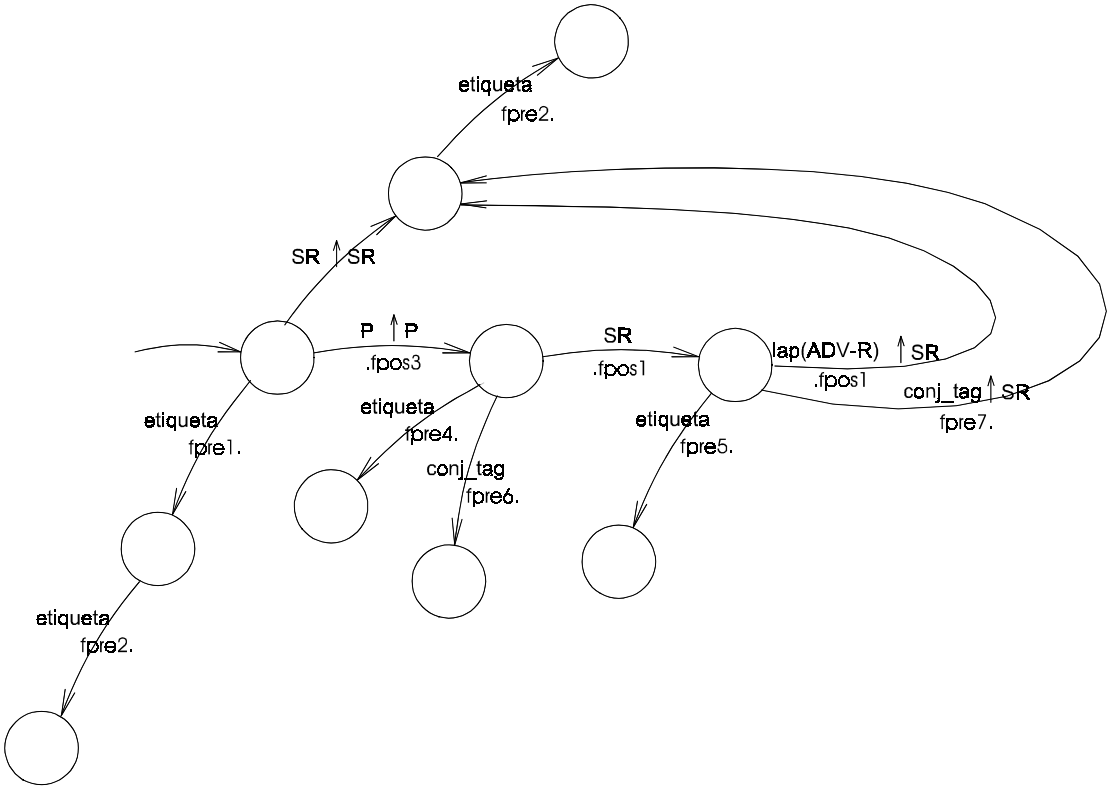


Figura 50 – Autômato sendo treinado para realizar a refinação contextual: foram consumidas as etiquetas P e SR; a etiqueta ADV-R foi apenas consultada

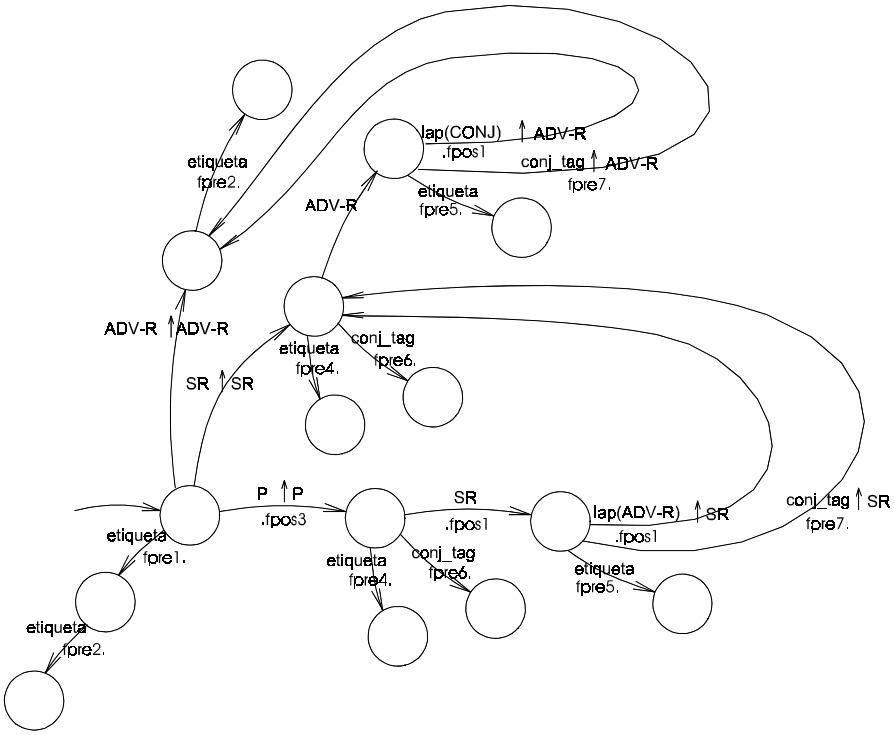


Figura 51 – Autômato sendo treinado para realizar a refinação contextual: foram consumidas as etiquetas P, SR e ADV-R; a etiqueta CONJ foi apenas consultada

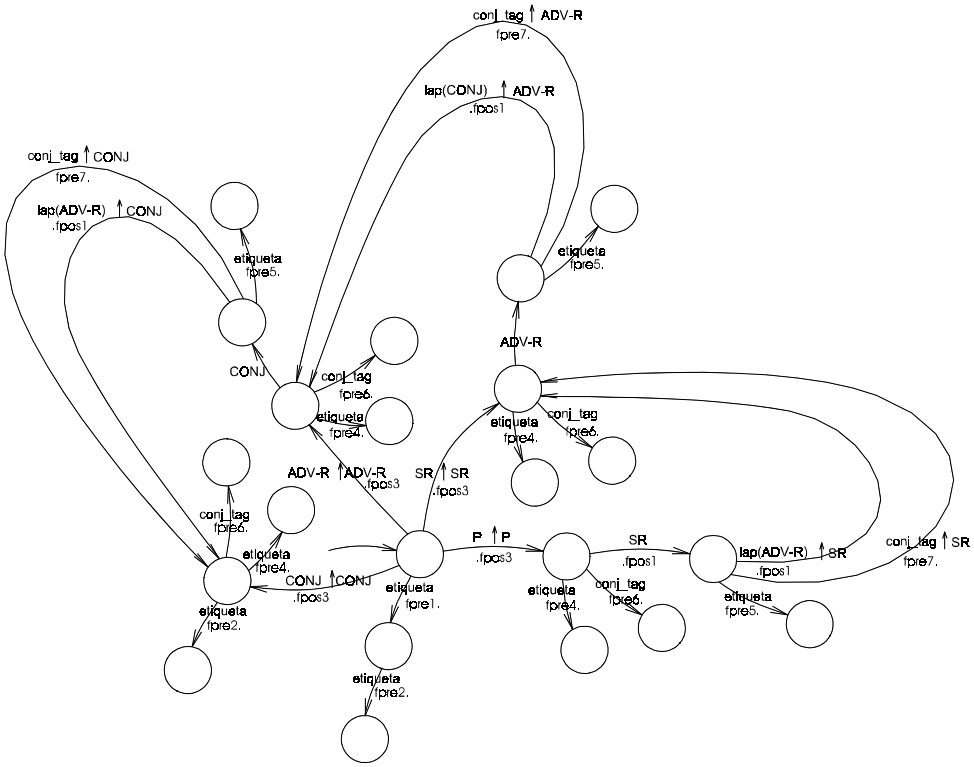


Figura 52 – Autômato sendo treinado para realizar a refinação contextual: foram consumidas as etiquetas P, SR, ADV-R e CONJ; a próxima etiqueta (ADV-R) foi apenas consultada

Seguindo a simulação da heurística de aprendizado, a Figura 53 e a Figura 54 mostram como o autômato armazena os trigramas P N VB-D e P N SR em suas transições, de modo que seja mantida uma estrutura de dados similar a uma árvore.

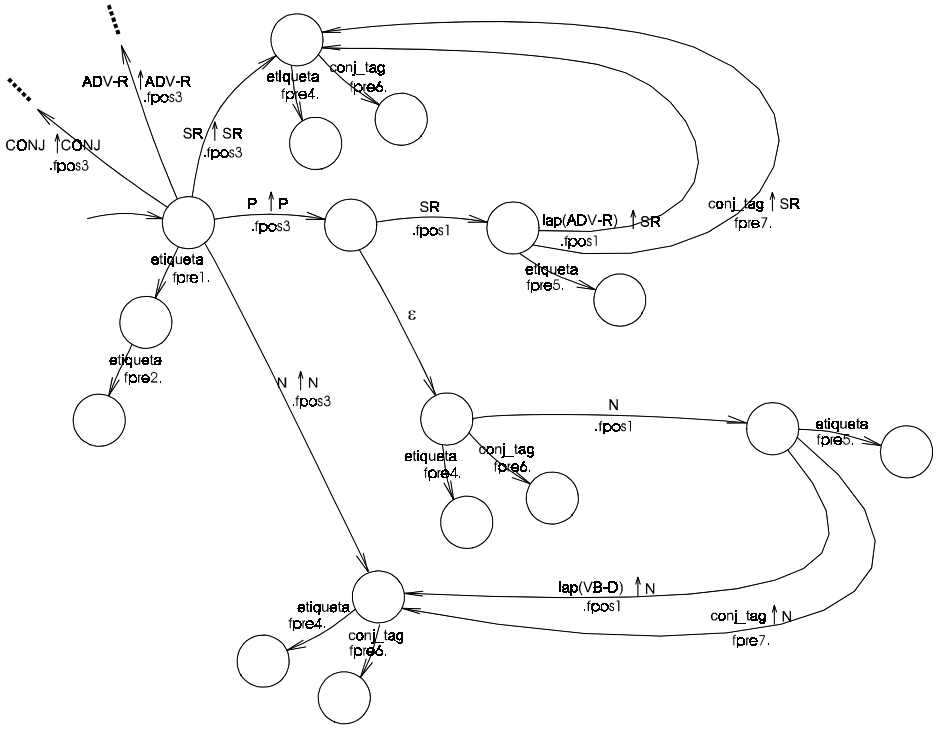


Figura 53 – Autômato sendo treinado para realizar a refinação contextual: foram consumidas as etiquetas P e N; a próxima etiqueta (VB-D) foi apenas consultada

Na verdade, os AA presentes neste trabalho não têm estrutura de árvore. Entretanto, se forem removidas as transições que servem apenas para controle, o que resta é realmente uma árvore, como pode ser visto na Figura 55, que, de fato, é apenas uma reformulação da Figura 54.

Como já foi comentado em seções anteriores, esta é uma estrutura que permite uma boa compressão no tamanho dos dados, e um sensível aumento na velocidade de acesso a estes dados.

Nota-se que o segundo e terceiro níveis de ramos (transições) desta árvore, a partir de sua raiz (estado inicial), estão em ordem decrescente de freqüência de aparecimento no corpus de treinamento (Figura 54 e Figura 55). Isto foi idealizado desta maneira para que se possa saber qual é o trigrama mais provável, a partir da primeira etiqueta fixada.

Neste módulo, diferentemente do que acontece no segundo módulo, não há distinção entre as fases de treinamento e de aplicação. Após ter incorporado todos os trigramas possíveis como exemplos de contextos, o autômato resultante pode ser usado para o refinamento de etiquetas ambíguas (quando há mais de uma etiqueta possível para determinada palavra).

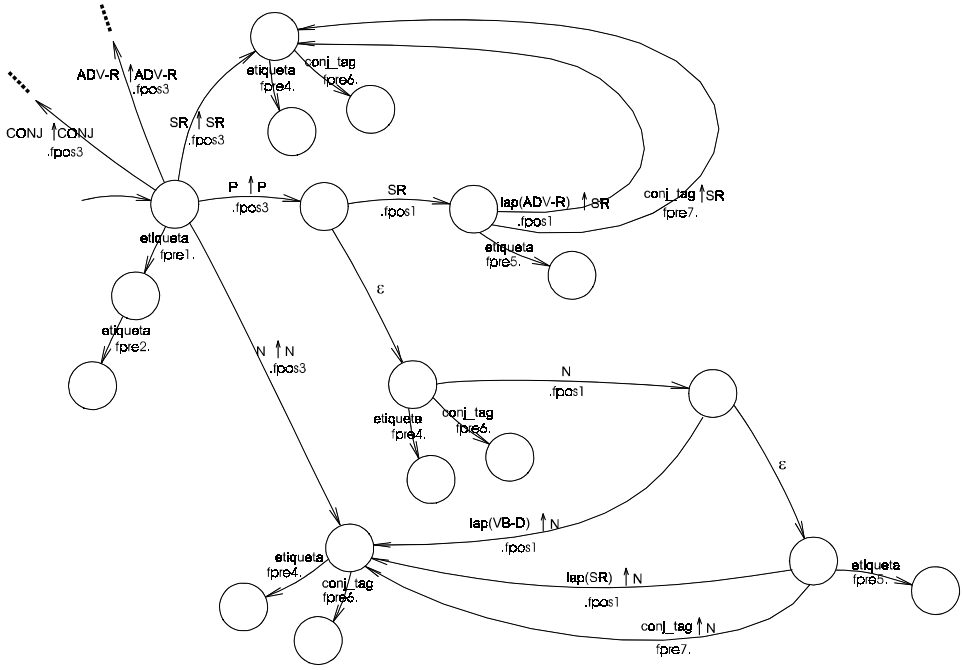


Figura 54 – Autômato sendo treinado para realizar a refinação contextual: foram consumidas as etiquetas P e N; a próxima etiqueta (SR) foi apenas consultada

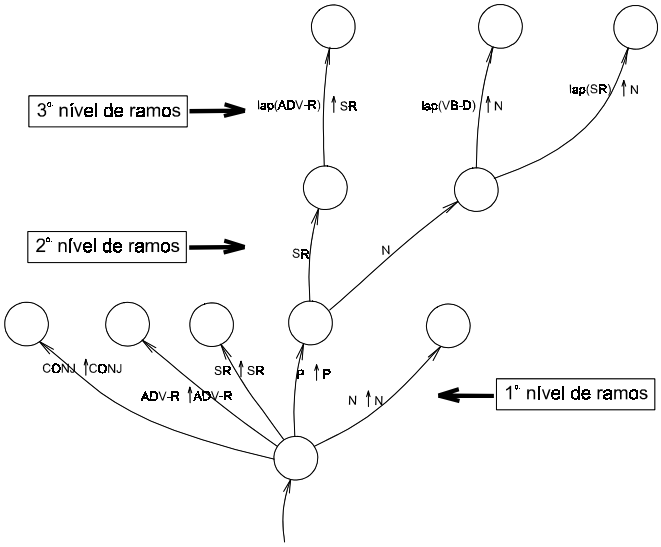


Figura 55 – Autômato anterior sem algumas transições de controle: destaca-se a estrutura em forma de árvore

A heurística de aplicação do conhecimento contextual adquirido é resumido na Figura 56. Pressupõe-se que o processo se inicie a partir de uma etiqueta não ambígua; a etiqueta seguinte, a qual será chamada de Foco, é a que será refinada, tendo em vista as etiquetas anterior e posterior (esta última pode ser ambígua ou não). Portanto, será escolhida uma dentre as várias etiquetas possíveis, de acordo com o contexto, para substituir o Foco.

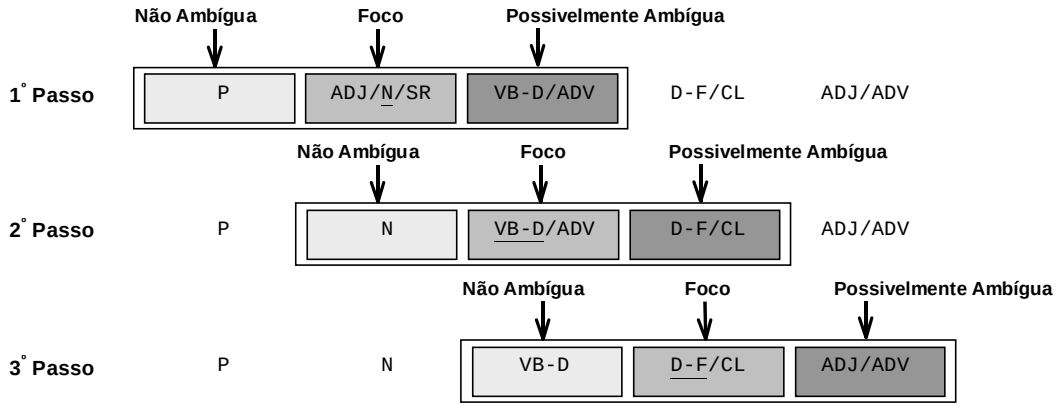


Figura 56 – Janela de três posições para resolver as ambigüidades pelo contexto

Olhando-se para a primeira janela da Figura 56 (1° Passo), percebe-se que existem 6 ($1 \times 3 \times 2$) possibilidades de trigramas sem ambigüidades, conforme listado abaixo:

P ADJ VB-D
P ADJ ADV
P N VB-D
P N ADV
P SR VB-D
P SR ADV

Suponha-se que apenas o trigrama ressaltado acima com negrito (P N VB-D) apareça no corpus de treinamento. Seria, então, natural esperar que o módulo de refinação contextual optasse pela etiqueta N para substituir a etiqueta ambígua ADJ/N/SR.

Estas decisões são tomadas dinamicamente pelo autômato através da montagem de novas transições, como destacado na Figura 57, com linhas mais grossas.

Como se pode observar, foi criada uma transição cujo estímulo é a própria etiqueta ambígua, e, em sua seqüência, um conjunto de transições com estímulo **lap**; estes estímulos testarão a etiqueta seguinte, que possivelmente é ambígua, sem entretanto consumi-la, transitando pela primeira que for bem sucedida. Esta transição propiciará a escrita, na cadeia de entrada, da etiqueta refinada contextualmente, que no caso desta simulação seria a etiqueta N.

A última das transições deste conjunto tem como estímulo a chamada Opção Default, a qual é apenas uma transição em vazio que envia para a cadeia de entrada a etiqueta mais provável dentre as componentes do Foco (etiqueta ambígua a ser corrigida), sem levar em consideração o contexto. No referido exemplo, esta é a etiqueta ADJ. Ou seja, se o contexto não oferecer informações suficientes (talvez por causa do treinamento com um corpus de tamanho pequeno, pouco significativo), opta-se por utilizar as informações lexicais vindas dos módulos anteriores (o de palavras conhecidas e o de palavras desconhecidas), considerando-se apenas a etiqueta mais provável.

Pode-se notar que todas estas transições que visam resolver ambigüidade estão associadas à função adaptativa posterior **fposRemoveTudo**, a qual tem a atribuição de desmontar completamente esta estrutura de refinamento contextual, depois de a mesma ter sido utilizada (Figura 57).

Considerando-se a segunda janela da Figura 56 (2º Passo), pode-se observar que em sua primeira posição está a etiqueta que foi o produto do refinamento anteriormente citado, e esta etiqueta é utilizada, na presente ocasião, como parte do contexto que irá colaborar para o refinamento da próxima etiqueta ambígua (VB-D/ADV). Neste caso, existem 4 ($1 \times 2 \times 2$) possibilidades de trigramas sem ambigüidades, conforme listado abaixo, sendo que o primeiro é o que veio a ser escolhido, por ser o mais provável. Isto implica na escolha da etiqueta VB-D como refinamento de VB-D/ADV.

N	**VB-D**	**D-F**
N	VB-D	CL
N	ADV	D-F

N ADV CL

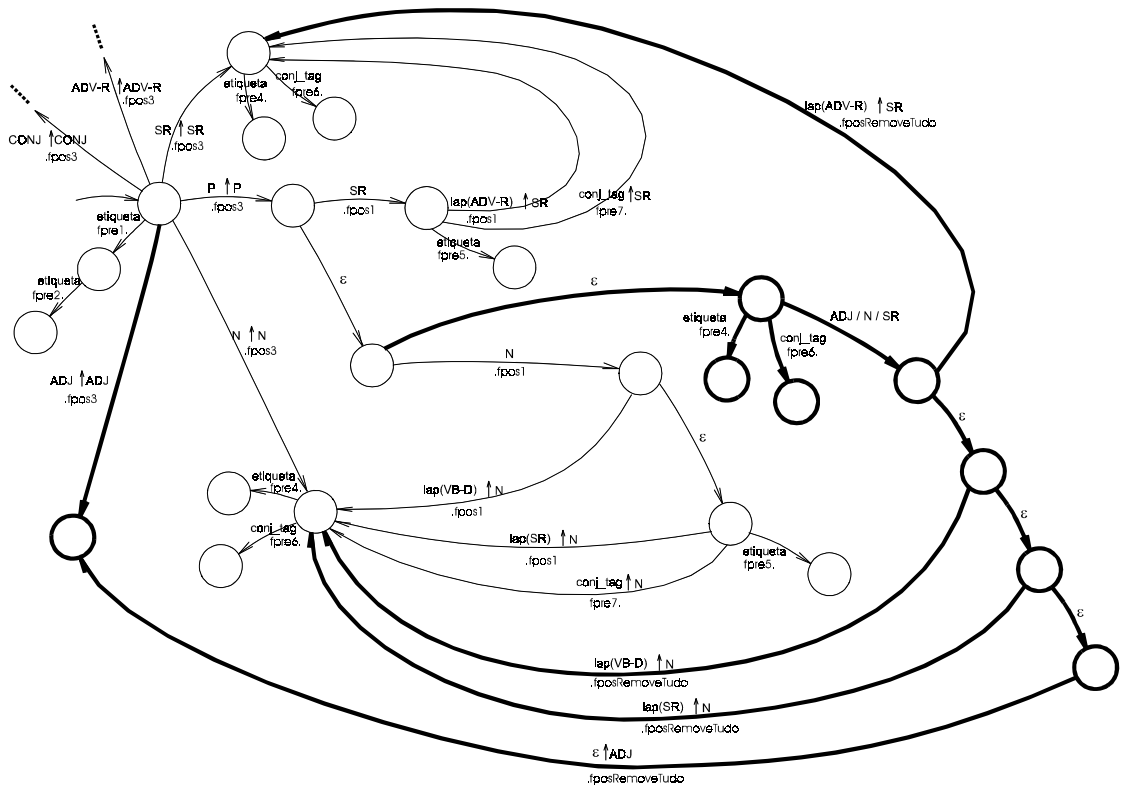


Figura 57 – Fase de Aplicação do refinador contextual

Algo similar ocorre com a terceira janela da Figura 56 (3º Passo), onde o Foco é a etiqueta ambígua D-F/CL, que será refinada e dará lugar à etiqueta D-F.

Levando-se em consideração esta estratégia de montagem e desmontagem de estruturas auxiliares, este AA apresenta uma estrutura bastante dinâmica e com o número de transições utilizadas sempre minimizado, já que estas estruturas de refinamento de etiquetas ambíguas são descartadas após a decisão ter sido tomada.

Contudo, se a prioridade fosse a velocidade de etiquetagem e não houvesse limitações quanto ao número de transições (caso haja espaço suficiente em memória física ou em memória virtual) a heurística poderia ser alterada de modo a não remover estas estruturas criadas para a resolução de ambigüidade. Havendo outro trigrama idêntico em outra parte do texto, esta estrutura não precisaria ser novamente montada, sendo imediatamente aproveitada.

4. Aspectos de Implementação

Este capítulo tem por objetivo descrever os experimentos realizados e comentar os resultados obtidos nestes.

Todos os experimentos realizados aderiram aos métodos e heurísticas propostos no capítulo 3 e farão o papel de demonstrar a viabilidade destes para a tarefa de etiquetar morfologicamente um texto livre da língua portuguesa.

A dificuldade central desta tarefa, em comparação com a anotação morfológica em línguas como o inglês, reside no fato de que há a necessidade de um número bem maior de etiquetas para representar a maior riqueza morfológica da língua portuguesa (o corpus *Penn Treebank*, com textos em inglês, usa um conjunto de 36 etiquetas morfológicas, a menos de pontuações, enquanto que o corpus Tycho Brahe, com textos em português, usa 231 etiquetas, conforme comentado na seção 2.1.3).

O método para a etiquetação morfológica proposto neste trabalho não é afetado por esta dificuldade.

4.1 Experimentos Realizados

Resumidamente, serão realizados dois experimentos: um inicial, utilizando-se de um corpus de treinamento realmente muito pequeno, porém, que tem como objetivo mostrar a viabilidade do projeto; e o segundo experimento, que estará mais próximo da realidade por utilizar um corpus de treinamento maior e por se esperar dele um bom desempenho quanto à taxa de acerto.

Cada um destes experimentos é subdividido em três partes, que correspondem aos três módulos do etiquetador morfológico.

4.1.1 Primeiro experimento

O primeiro experimento realizado é um tanto quanto limitado do ponto de vista de validade prática, no entanto, tem uma função conceitual importante. Os diversos módulos foram treinados com o uso de um pedaço que não faz mais parte do corpus Tycho Brahe, composto de 1.812 palavras e dividido em duas partes (foi usado o que era disponível na época da realização do experimento) [TBCHP-98]:

- corpus de treinamento: 1.684 itens lexicais (palavras e pontuações).
- corpus de aplicação: 128 itens lexicais.

O chamado corpus de aplicação é usado para a realização de uma medição do desempenho do etiquetador, em termos de acertos, bem como de cada um de seus módulos.

Como o corpus de aplicação é parte integrante do corpus inicial, ele contém anotações morfológicas corretas. Necessita-se, então, separar os itens lexicais das etiquetas a eles associadas. Pode-se dizer que foi criado um nova versão do corpus de aplicação, formada apenas pelos seus itens lexicais.

Esta nova versão do corpus de aplicação será usada como texto a ser etiquetado. Para proceder à aferição dos módulos do etiquetador, basta realizar uma comparação entre as etiquetas atribuídas por estes módulos e aquele conjunto de etiquetas pertencente ao corpus de aplicação original. A porcentagem de etiquetas coincidentes corresponde à taxa de acerto do módulo em análise. Esta forma de medir o desempenho foi usada por outros pesquisadores, como E. Brill; a única diferença entre sua medida e a adotada neste trabalho é que Brill preferiu medir a **distância** entre duas anotações, enquanto que aqui foi útil medir a **coincidência** entre duas anotações (Figura 6) [BRILL-93].

A seguir, pode-se examinar o referido corpus de aplicação:

```
Teve a mesma sorte o padre Gregório da Rocha , natural da capitania de
Pernambuco . Tinha , ao tempo em que Deus o chamou para si , trinta anos de
idade . Entrou no ano de 1611 , de 15 anos , na Companhia , e nela viveu outros
15 com satisfação e observância religiosa . Sabia bem a língua da terra ; e
```

melhor a exercitava nas aldeias , cultivando os índios . Era enfermo de ordinário , e nas enfermidades mui animoso e paciente ; em especial na última , que foi mui trabalhosa , e de grandíssimas dores entre as quais , com muita devoção e consolação da sua alma , recebeu todos os Sacramentos e faleceu a 9 de Maio de 1625 .

4.1.1.1 Primeiro módulo: etiqueta mais provável para as palavras conhecidas

O experimento descrito nesta seção envolve a implementação da primeira etapa do analisador morfológico explanado na seção 3.2.1 (Figura 24).

Inicialmente, o autômato é treinado com o uso do corpus de treinamento, passando a incorporar o léxico encontrado neste corpus e todas as possíveis etiquetas morfológicas para cada item lexical, ordenadas por freqüência.

Para testar o funcionamento deste módulo, a nova versão do corpus de aplicação é submetida ao autômato. Assim, pode-se avaliar quão bem sucedida foi esta fase, através da taxa de acerto medida.

Como resultado, obteve-se o seguinte texto anotado:

Teve **a/P/D-F** mesma/ADJ-F sorte o/D/CL padre/NPR/N Gregório da/P+D-F/P+D Rocha ,/, natural/ADJ-G da/P+D-F/P+D capitania de/P Pernambuco/NPR ./. Tinha ,/, ao/P+D tempo/N em/P que/WPRO/C Deus/NPR **o/D/CL** chamou para/P si/PRO ,/, trinta anos/N-P de/P idade/N ./. Entrou/VB-D no/P+D ano/N de/P 1611 ,/, de/P 15/NUM anos/N-P ,/, na/P+D-F Companhia/NPR ,/, e/CONJ nela/P+PRO viveu/VB-D outros/ADJ-P/N-P 15/NUM com/P satisfação e/CONJ observância religiosa/ADJ-F ./. Sabia bem/ADV **a/P/D-F** língua da/P+D-F/P+D terra ;/. e/CONJ **melhor/ADJ-R-G/ADV-R a/P/D-F** exercitava nas/P+D-F-P aldeias/N-P ,/, cultivando os/D-P/CL índios/N-P ./. Era enfermo de/P ordinário ,/, e/CONJ nas/P+D-F-P enfermidades **mui/ADV/Q** animoso e/CONJ paciente ;/. em/P especial/ADJ-G na/P+D-F última/ADJ-F ,/, que/WPRO/C foi/SR-D **mui/ADV/Q** trabalhosa ,/, e/CONJ de/P grandíssimas dores entre as/D-F-P quais/WPRO-P ,/, com/P muita/Q-F devoção/N e/CONJ consolação da/P+D-F/P+D sua/PRO$-F alma/N ,/, recebeu todos/Q-P os/D-P/CL Sacramentos e/CONJ faleceu/VB-D a/P/D-F 9 de/P Maio de/P 1625/NUM ./.

Pode-se notar que alguns itens lexicais não foram anotados; estes são desconhecidos, ou seja, não aparecem no corpus de treinamento. Pode-se também observar que estão associadas diversas etiquetas a algumas palavras, significando isto que, no corpus de treinamento, estas palavras pertencem a mais de uma categoria morfológica; nestes casos, as etiquetas aparecem em ordem decrescente de sua freqüência de aparecimento no corpus de treinamento, ou seja, a primeira é a mais freqüente.

Para avaliar o desempenho na anotação morfológica deste módulo temos de fazer uma suposição que, aliás, é muito razoável: nos casos em que mais de uma etiqueta é fornecida, considera-se a primeira, que é a mais freqüente, como sendo a anotação mais provável e, portanto, a escolhida para aquela palavra.

Desta forma, dos 128 itens lexicais (palavras e pontuações) encontrados no corpus de aplicação, 31 destes (24,22%) não foram anotados (isto porque não apareciam no corpus de treinamento) e 97 foram anotados (75,78%); dos anotados, 90 foram anotados corretamente (70,31% do total) e 7 erroneamente (5,47% do total). Estes foram marcados com negrito acima e constituem exemplos de ambigüidade morfológica. As etiquetas corretas, juntamente com as outras opções de etiquetas, podem ser vistas na Tabela 5.

Item Lexical	Etiqueta atribuída	Opções secundárias de etiqueta	Etiqueta correta
a	P	D-F	D-F
o	D	CL	CL
melhor	ADJ-R-G	ADV-R	ADV-R
a	P	D-F	CL
mui	ADV	Q	Q

Tabela 5 – Anotações incorretas na saída do primeiro módulo do etiquetador

Estes resultados foram muito positivos, visto que o módulo acertou 92,78% dos itens lexicais que etiquetou (acertou 90 num total de 97). Realmente, este módulo não conseguiu acertar mais por causa do pequeno tamanho do corpus de treinamento, ou seja, 24,22% dos itens lexicais eram desconhecidos.

As incorreções cometidas por este módulo e registradas na Tabela 5 podem ser facilmente explicadas. Pode-se notar que a etiqueta correta (CL) para a quarta palavra (“a”) nem aparecia entre as duas etiquetas sugeridas (P e D-F), o que mostra que faltaram informações para o módulo, que deveriam ter vindo do corpus de treinamento. Ou seja, era necessário um corpus maior, que contivesse a palavra “a” etiquetada como CL.

É importante observar que nas outras quatro incorreções, a etiqueta correta coincidia com a segunda sugestão de anotação morfológica. Isto mostra que alguma

forma de refinamento é necessário; o terceiro módulo do etiquetador dará atenção a problemas como este.

4.1.1.2 Segundo módulo: etiqueta para palavras desconhecidas, com base em sufixos

A segunda etapa da implementação do etiquetador morfológico, a qual trabalha com palavras desconhecidas, refere-se ao explanado na seção 3.2.2 (Figura 24).

O autômato inicial é treinado com o uso de uma versão pré-processada do corpus de treinamento, a qual é formada somente pelos sufixos das palavras e etiquetas correspondentes. Este autômato cresce, armazenando todos estes sufixos, relacionando-os às possíveis etiquetas morfológicas.

Para testar o funcionamento deste módulo, a saída do módulo anterior alimenta este autômato, depois de passar por um pré-processamento (Figura 33).

Vale comentar que os pré e pós-processamentos sugeridos tornam o processo mais simples de ser entendido, já que permite, por exemplo, a visualização de uma versão do corpus de treinamento que consiste apenas de sufixos e etiquetas (pode-se ter uma idéia disto pela observação de trechos do corpus sendo processados na Figura 33). Seria possível uma implementação que dispensasse estes passos intermediários, contudo, esta seria mais complexa; uma vantagem advinda desta mudança seria uma maior velocidade de processamento, porém, isto não alteraria a taxa de acerto, já que a heurística de trabalho seria a mesma.

A saída deste módulo, depois de pós-processada (Figura 33), pode ser observada a seguir:

> **Teve/ET-D a/P/D-F** mesma/ADJ-F **sorte/ADV/N/NUM** o/D/CL padre/NPR/N **Gregório/N** da/P+D-F/P+D **Rocha/TR-D/VB-D** ,/, natural/ADJ-G da/P+D-F/P+D **capitania/VB-D/N/NPR** de/P Pernambuco/NPR ./. Tinha/TR-D/VB-D ,/, ao/P+D tempo/N em/P que/WPRO/C Deus/NPR **o/D/CL** chamou/VB-D para/P si/PRO ,/, **trinta/D-F/P+D-F** anos/N-P de/P idade/N ./. Entrou/VB-D no/P+D ano/N de/P **1611/N** ,/, de/P 15/NUM anos/N-P ,/, na/P+D-F Companhia/NPR ,/, e/CONJ nela/P+PRO viveu/VB-D outros/ADJ-P/N-P 15/NUM com/P satisfação/N e/CONJ **observância/VB-D/N/NPR** religiosa/ADJ-F ./. Sabia/VB-D/N/NPR bem/ADV **a/P/D-F** língua/N da/P+D-F/P+D **terra/P/VB-RA** ;/. e/CONJ **melhor/ADJ-R-G/ADV-R a/P/D-F** exercitava/VB-D/ET-D nas/P+D-F-P aldeias/N-P ,/, cultivando/VB-G/SR-G/CONJS os/D-P/CL índios/N-P ./. **Era/P/VB-RA enfermo/CONJS** de/P **ordinário/NPR/N** ,/, e/CONJ nas/P+D-F-P

enfermidades/N-P/ADV/P+D-P/D-P **mui/ADV/Q** animoso/ADJ e/CONJ **paciente/ADV/N/NUM** ;/. em/P especial/ADJ-G na/P+D-F última/ADJ-F ,/, que/WPRO/C foi/SR-D **mui/ADV/Q** trabalhosa/ADJ-F ,/, e/CONJ de/P **grandíssimas/PRO$-F-P/N-P/NUM-F** dores/N-P/ADV/P+D-P/D-P **entre/ADJ-G/N/NPR** as/D-F-P quais/WPRO-P ,/, com/P muita/Q-F devoção/N e/CONJ consolação/N da/P+D-F/P+D sua/PRO$-F alma/N ,/, recebeu/VB-D todos/Q-P os/D-P/CL **Sacramentos/Q-P/VB-AN-P/N-P** e/CONJ faleceu/VB-D a/P/D-F **9/N** de/P Maio/NPR/N de/P 1625/NUM ./.

Nota-se que agora todos os itens lexicais estão anotados. Justamente aqueles que são desconhecidos foram os que receberam anotações desta vez. Da mesma maneira que no módulo anterior, as palavras desconhecidas puderam receber mais de uma etiqueta, significando que, no corpus de treinamento, aos sufixos correspondentes a estas palavras estão associadas diversas categorias morfológicas; nestes casos, as etiquetas aparecem em ordem decrescente de sua freqüência de aparecimento no corpus.

Far-se-á, novamente, a suposição de que, nos casos em que mais de uma etiqueta é fornecida, considera-se a primeira, que é a mais freqüente, como sendo a anotação mais adequada para aquela palavra.

Desta forma, dos 128 itens lexicais (palavras e pontuações) encontrados no corpus de aplicação, 104 foram anotados corretamente (81,25%) e 24 erroneamente (18,75%). Estas anotações erradas foram marcados com negrito acima. As etiquetas corretas podem ser vistas na Tabela 6, juntamente com as opções secundárias.

Sob esta seção, ainda cabe um outro experimento, que visa avaliar a heurística do pré-processamento do corpus de treinamento. Na seção 3.2.2 foi argumentado que o estágio denominado PODA, o qual eliminava do corpus todos os itens lexicais com três letras ou menos (além de deixar apenas as três últimas letras dos outros itens lexicais), contribuia para que o segundo módulo não aprendesse falsos sufixos.

Portanto, para verificar a veracidade desta afirmação, será feito um novo experimento, similar ao anterior, só que com uma modificação no estágio PODA: todos os itens lexicais com três letras ou menos são mantidos. Todo o restante do pré e pós-processamento é mantido.

Após o novo treinamento, a saída pré-processada do primeiro módulo alimenta este novo autômato; a saída deste, depois de pós-processada, pode ser observada a seguir:

Item Lexical	Etiqueta atribuída	Opções secundárias de etiqueta	Etiqueta correta
Teve	ET-D	–	TR-D
a	P	D-F	D-F
sorte	ADV	N/NUM	N
Gregório	N	–	NPR
Rocha	TR-D	VB-D	NPR
capitania	VB-D	N/NPR	N
o	D	CL	CL
trinta	D-F	P+D-F	NUM
1611	N	–	NUM
observância	VB-D	N/NPR	N
a	P	D-F	D-F
terra	P	VB-RA	N
melhor	ADJ-R-G	ADV-R	ADV-R
a	P	D-F	CL
Era	P	VB-RA	SR-D
enfermo	CONJS	–	ADJ
ordinário	NPR	N	N
mui	ADV	Q	Q
paciente	ADV	N/NUM	ADJ-G
mui	ADV	Q	Q
grandíssimas	PRO$-F-P	N-P/NUM-F	ADJ-S-F-P
entre	ADJ-G	N/NPR	P
Sacramentos	Q-P	VB-AN-P/N-P	NPR-P
9	N	–	NUM

Tabela 6 – Anotações erradas na saída do segundo módulo do etiquetador

Teve/ET-D a/P/D-F mesma/ADJ-F **sorte/ADV/N/NUM** o/D/CL padre/NPR/N **Gregório/N** da/P+D-F/P+D **Rocha/TR-D/VB-D** ,/, natural/ADJ-G da/P+D-F/P+D **capitania/VB-D/N** de/P Pernambuco/NPR ./. Tinha/TR-D/VB-D ,/, ao/P+D tempo/N em/P que/WPRO/C Deus/NPR **o/D/CL** **chamou/CONJ** para/P si/PRO ,/, **trinta/D-F/P+D-F** anos/N-P de/P idade/N ./. Entrou/VB-D no/P+D ano/N de/P 1611/NUM ,/, de/P 15/NUM anos/N-P ,/, na/P+D-F Companhia/NPR ,/, e/CONJ nela/P+PRO viveu/VB-D outros/ADJ-P/N-P 15/NUM com/P **satisfação/NEG/SENÃO/NPR** e/CONJ **observância/VB-D/N** religiosa/ADJ-

F ./. Sabia/VB-D/N bem/ADV **a/P/D-F** **língua/PRO$-F** da/P+D-F/P+D **terra/P/VB-RA** ;/. e/CONJ **melhor/ADJ-R-G/ADV-R** **a/P/D-F** exercitava/VB-D/ET-D nas/P+D-F-P aldeias/N-P ,/, **cultivando/P+D** os/D-P/CL índios/N-P ./. **Era/P/VB-RA** **enfermo/CONJS** de/P **ordinário/NPR/N** ,/, e/CONJ nas/P+D-F-P enfermidades/N-P/ADV/P+D-P/D-P **mui/ADV/Q** animoso/ADJ e/CONJ **paciente/ADV/N/NUM** ;/. em/P especial/ADJ-G na/P+D-F última/ADJ-F ,/, que/WPRO/C foi/SR-D **mui/ADV/Q** trabalhosa/ADJ-F ,/, e/CONJ de/P **grandíssimas/D-F-P** dores/N-P/ADV/P+D-P/D-P **entre/ADJ-G/N/NPR** as/D-F-P quais/WPRO-P ,/, com/P muita/Q-F devoção/N e/CONJ **consolação/NEG/SENÃO/NPR** da/P+D-F/P+D sua/PRO$-F alma/N ,/, recebeu/VB-D todos/Q-P os/D-P/CL **Sacramentos/D-P/CL** e/CONJ faleceu/VB-D a/P/D-F 9/NUM de/P Maio/NPR/N de/P 1625/NUM ./.

A convenção adotada para interpretar este resultado é a seguinte: palavras em negrito indicam incorreções na anotação; palavras dentro de uma borda, mas sem sombreamento interno, indicam acertos neste experimento que correspondem a anotações erradas no experimento anterior; e palavras dentro de uma borda, com sombreamento interno, indicam incorreções neste experimento, correspondentes a acertos no experimento anterior.

Pode-se observar que, neste último experimento, o módulo acertou 2 etiquetas que o experimento anterior não tinha acertado, mas errou 5 que este outro tinha acertado. Isto resulta em 101 itens lexicais (palavras e pontuações) anotados corretamente (78,91%) dos 128 encontrados no corpus de aplicação. Ou seja, houve uma piora no desempenho deste módulo.

Além disso, se for feita uma análise atenta ao tipo de erro cometido, será possível verificar que falsos sufixos foram aprendidos: por exemplo, a palavra "língua" foi etiquetada como PRO$-F (pronome possessivo feminino singular) porque as palavras "s<u>ua</u>" e "t<u>ua</u>" recebem esta etiqueta: o módulo inferiu o falso sufixo "ua", associando-o a PRO$-F.

De forma geral o desempenho deste módulo foi bom: ele foi responsável pela etiquetação de 31 palavras (são as palavras desconhecidas); destas 31 palavras, 14 delas foram anotadas corretamente (45, 16%) e 17, erroneamente (54,84%). Algumas das etiquetações errôneas possuem uma segunda possibilidade de anotação, a qual corresponde à etiqueta correta (Tabela 6); isto indica a necessidade de um refinamento posterior, com a atenção voltada ao contexto.

4.1.1.3 Terceiro módulo: refinamento contextual para tirar ambigüidades

A terceira etapa da implementação do etiquetador morfológico diz respeito à seção 3.2.3 e cuida de um refinamento no processamento dos módulos anteriores com base em informações contextuais (Figura 24).

O autômato inicial é treinado com o uso de uma versão do corpus de treinamento formado apenas por etiquetas. Este autômato cresce, armazenando todos os trigramas de etiquetas existentes nesta versão do corpus.

Para testar o funcionamento deste módulo, separam-se as etiquetas dos itens lexicais encontrados na saída do módulo anterior; as etiquetas (ou melhor, listas de etiquetas, visto que cada palavra pode estar associada a mais de uma etiqueta) são, então, fornecidas a este autômato treinado.

As saídas deste módulo devem ser novamente acopladas aos itens lexicais dos quais foram estraídas antes do processamento propriamente dito; abaixo, pode-se observar a saída resultante:

```
Teve/ET-D a/P mesma/ADJ-F sorte/ADV o/D padre/NPR Gregório/N da/P+D-F Rocha/TR-
D ,/, natural/ADJ-G da/P+D-F capitania/N de/P Pernambuco/NPR ./. Tinha/TR-D ,/,
ao/P+D tempo/N em/P que/WPRO Deus/NPR o/CL chamou/VB-D para/P si/PRO ,/,
trinta/D-F anos/N-P de/P idade/N ./. Entrou/VB-D no/P+D ano/N de/P 1611/N ,/,
de/P 15/NUM anos/N-P ,/, na/P+D-F Companhia/NPR ,/, e/CONJ nela/P+PRO viveu/VB-
D outros/ADJ-P 15/NUM com/P satisfação/N e/CONJ observância/N religiosa/ADJ-F
./. Sabia/VB-D bem/ADV a/P língua/N da/P+D-F terra/P ;/. e/CONJ melhor/ADJ-R-G
a/P exercitava/VB-D nas/P+D-F-P aldeias/N-P ,/, cultivando/CONJS os/D-P
índios/N-P ./. Era/P enfermo/CONJS de/P ordinário/N ,/, e/CONJ nas/P+D-F-P
enfermidades/N-P mui/ADV animoso/ADJ e/CONJ paciente/N ;/. em/P especial/ADJ-G
na/P+D-F última/ADJ-F ,/, que/WPRO foi/SR-D mui/ADV trabalhosa/ADJ-F ,/, e/CONJ
de/P grandíssimas/PRO$-F-P dores/N-P entre/ADJ-G as/D-F-P quais/WPRO-P ,/,
com/P muita/Q-F devoção/N e/CONJ consolação/N da/P+D-F sua/PRO$-F alma/N ,/,
recebeu/VB-D todos/Q-P os/D-P Sacramentos/Q-P e/CONJ faleceu/VB-D a/P 9/N de/P
Maio/N de/P 1625/NUM ./.
```

Nota-se que todos os itens lexicais estão anotados com apenas uma etiqueta; e justamente é este o trabalho do terceiro módulo, eleger uma dentre várias etiquetas possíveis, com base em um contexto próximo.

Para avaliar o desempenho do analisador morfológico como um todo, basta comparar esta saída com o corpus de aplicação. Dos 128 itens lexicais (palavras e pontuações) encontrados no corpus de aplicação, 106 foram anotados corretamente (82,81%) e 22 erroneamente (17,19%).

A Tabela 9 resume o desempenho do analisador morfológico, nos seus diversos módulos.

Taxa de acerto		
1° módulo	**2° módulo**	**3° módulo (final)**
70,31%	81,25%	82,81%

Tabela 7 – Resumo do desempenho do analisador morfológico no 1° experimento

Comparando-se a saída final (deste módulo) com a saída do módulo anterior, nota-se que, na maioria das vezes, o refinador contextual optou pela etiqueta mais provável (a primeira da lista); deste modo, a maioria dos supostos erros vislumbrados na seção anterior (4.1.1.2) acabaram sendo consumados.

Porém, há boas exceções na atuação do refinador contextual. As palavras marcadas com negrito acima constituem os seis casos em que o corretor não optou pela primeira etiqueta da lista, devido ao contexto. Em quatro destes seis casos, a decisão foi acertada; em dois deles (as palavras destacadas em negrito: **cultivando** e **Maio**), o corretor decidiu erroneamente (Tabela 8).

Item Lexical	Etiqueta atribuída	Opções de etiqueta	Etiqueta correta
capitania	N	VB-D/N/NPR	N
o	CL	D/CL	CL
observância	N	VB-D/N/NPR	N
cultivando	**CONJS**	**VB-G/SR-G/CONJS**	**VB-G**
ordinário	N	NPR/N	N
Maio	**N**	**NPR/N**	**NPR**

Tabela 8 – Seis casos em que o refinador contextual não decidiu pela pela etiqueta mais provável

Percebe-se que, na grande maioria dos casos, o refinador não conseguiu atuar; apenas foi operacional nestes seis casos citados.

Nos outros 122 itens lexicais, ou só há uma etiqueta atribuída ao mesmo, ou o autômato optou pela "Opção *Default*", que é a primeira etiqueta da lista, e, portanto, a mais provável, sem considerar o contexto. Este último caso só ocorre quando as

informações inferidas durante a fase de treinamento são insuficientes para a decisão, ou quando realmente o contexto aponta para a primeira etiqueta da lista.

Em outras palavras, boa parcela destes erros podem ter origem em um treinamento deficiente, devido ao pequeno tamanho de corpus de treinamento utillizado nestes experimento.

4.1.2 Segundo experimento

O segundo experimento realizado já é mais abrangente e confiável, em aspecto prático. Os três módulos foram treinados com o uso de um texto de António das Chagas (1631-1682), que faz parte do corpus Tycho Brahe, e que é composto de 57.425 palavras, divididas da mesma forma que no primeiro experimento [TBCHP-98]:

- corpus de treinamento: 51.017 itens lexicais (palavras e pontuações).
- corpus de aplicação: 6.408 itens lexicais.

Procurou-se dividir o corpus total na proporção de cerca de 90% para o corpus de treinamento e cerca de 10% para o de aplicação.

A Tabela 9 resume o desempenho do etiquetador, em seus diversos módulos.

Taxa de acerto		
1° módulo	**2° módulo**	**3° módulo (final)**
85,75%	88,70%	91,01%

Tabela 9 – Resumo do desempenho do analisador morfológico no 2° experimento

O desempenho global do etiquetador pode ser considerado bom, pelos dados relativos à taxa de acerto conseguida. Contudo, há espaço ainda para melhoras, mas problemas práticos limitam o prosseguimento dos experimentos.

4.2 Resultados Obtidos

Os resultados do primeiro conjunto de experimentos indicou um desempenho baixo, no que tange à taxa de acertos; contudo, vale ressaltar que o tamanho do corpus

de treinamento usado foi muito pequeno, praticamente insignificante: 1.684 palavras anotadas.

Mesmo assim, obteve-se uma taxa de acerto de 81,25%, comparável ao trabalho de C. Alves, que atingiu a taxa de 78,28%, com um corpus de treinamento de 5.000 palavras, e ao trabalho de A. Villavicencio, que alcançou 84,5%, com um corpus de treinamento de 14.000 palavras [ALVES-99 e VILLAVICENCIO-95].

Como já era esperado, o aumento do corpus de treinamento propiciou um considerável aumento nesta taxa de acerto.

	Tamanho do corpus de treinamento	**Taxa de acerto no 1° módulo**	**Taxa de acerto no 2° módulo**	**Taxa de acerto total**
1° experimento	1.684	70,31%	81,25%	82,81%
2° experimento	51.017	85,75%	88,70%	91,01%

Tabela 10 – Resumo do desempenho do analisador morfológico no 2° experimento

Por exemplo, E. Brill começou a fazer experimentos que produziram resultados práticos com um corpus de 45.000 palavras; W. Daelemans e outros pesquisadores argumentam que o método baseado em exemplos memorizados começa a produzir resultados satisfatórios a partir de um corpus com 300.000 palavras [BRILL-93 e DAELEMANS-96a].

5. Conclusões

5.1 Avaliação Geral

As propostas feitas e suas correspondentes implementações descritas neste trabalho mostram que os AA permitem facilmente a modelagem de paradigmas e algoritmos de aprendizado automático.

5.2 Contribuições

Sem dúvida nenhuma, este trabalho constitue uma constatação significativa da adequação dos AA para a representação e manipulação de conhecimento da área de PLN (processamento de linguagens naturais). Mostrou sua viabilidade especialmente para a modelagem de algoritmos de aprendizado automático. Também deve ser ressaltado que o etiquetador para a língua portuguesa gerado é a primeira aplicação prática de larga escala baseada nos AA, mostrando que estes são dispositivos simples, elegantes, eficientes e treináveis.

Sob o olhar da lingüística computacional ou do PLN, foi construída uma ferramenta que propricia a análise morfológica de textos livres, com taxa de acerto comparável às dos paradigmas que representam o estado-da-arte na área, e com algumas vantagens, como:

- A complexidade computacional do treinamento e da aplicação dos três módulos que formam o etiquetador morfológico é independente do número de etiquetas e

linear com respeito à cadeia de entrada. Isto é uma grande vantagem deste método proposto, em relação ao de E. Brill (cuja fase de treinamento tem dependência polinomial com relação à quantidade de etiquetas) e em relação ao de C. Alves (que adaptou o etiquetador de E. Brill para a língua portuguesa), pois não necessita de um módulo adicional, com um conjunto de regras escritas manualmente, para dar ao etiquetador condições de manipular uma grande quantidade de etiquetas [ALVES-99].

- A possibilidade de explicar ou justificar uma decisão tomada, com base na maior proximidade de um determinado exemplo memorizado.
- Como o autômato adaptativo presente no etiquetador comporta-se de modo praticamente igual a um autômato finito, depois da fase de treinamento (já que as ações adaptativas nesta fase não são tão usadas), o desempenho da implementação pode estar muito perto do melhor possível, que seria o de um autômato finito [ROCHE-97]. Uma exceção a esta afirmação é o módulo responsável pelo refinamento contextual, o qual, mesmo na fase de aplicação, conta com o uso de funções adaptativas na montagem e desmontagem de estruturas auxiliares de transições. Contudo, se o objetivo for a obtenção de alta velocidade de etiquetagem e se houver um bom gerenciamento de memória por parte da aplicação e/ou do sistema operacional, de modo que não haja limitação quanto ao número de transições, a heurística do refinamento contextual poderia ser otimizada de modo que não sejam removidas as estruturas criadas para a resolução de ambigüidade. Havendo outro trigrama idêntico em outra parte do texto, esta estrutura não precisaria ser novamente montada, sendo imediatamente aproveitada.

5.3 Trabalho Futuro

Pode-se dizer que, quanto aos formalismos adaptativos (entre os quais se enquadram os autômatos adaptativos E), o grande desafio hoje é a criação de métodos sistemáticos de projeto, visto serem de concepção muito recente. Atualmente, as aplicações estão sendo ainda projetadas muito artesanalmente.

Sob a ótica do PLN, almeja-se investigar a possibilidade de usar heurísticas similares às descritas neste trabalho para a tarefa da anotação sintática. Este seria o

próximo passo em direção à construção de diversos sistemas úteis, tais como interpretadores de textos livres, tradutores automáticos, etc.

Para que se possa viabilizar aplicações realistas, dever-se-ia construir uma espécie de compilador para os autômatos adaptativos E, visto que, na atual implementação, a velocidade de aplicação deste autômato sobre o texto a etiquetar é limitada pelo uso do interpetador construído, o qual é executado por uma linguagem interpretada (Prolog).

No que tange a desempenho, deve-se ainda discutir muito uma implementação eficiente da ação adaptativa elementar **Inspeção**, visto que sua realização em máquinas seqüenciais e com memória de acesso aleatório se mostra excessivamente onerosa. Em uma máquina paralela, com memória associativa, esta ação adaptativa poderia ser implementada muito facilmente e com ótimo desempenho.

Uma dissertação, inserida na área dos autômatos e linguagem formais, propôs uma linguagem, bem como um ambiente integrado de programação, baseados em autômatos adaptativos; a implementação destes proveu uma realização bastante eficiente desta ação adaptativa elementar, contudo, com apenas um parâmetro livre, devido à complexidade da solução geral [PEREIRA-99]. A solução adotada para o interpretador de autômatos adaptativos E aproveitou uma das características mais poderosas da linguagem Prolog: a **unificação**. Desta forma, o código do interpretador ficou bastante simples, mas não se pode esperar grande desempenho desta versão interpretada.

6. Referências

ALLEN, J. **Natural language understanding**. Menlo Park, Benjamin/Cummings, 1987.

ALVES, C.; FINGER, M. 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, 21-22 September 1999.

ANTWORTH, E. Introduction to Two-level Phonology, p.4-18. In **Notes on Linguistics N. 53**. Summer Institute of Linguistics, 1991 (http://www.sil.org/pckimmo/two-level_phon.html).

BENNETT, W.; SLOCUM, J. The LRC machine translation system. **Computational Linguistics**, v.11, n.2-3, p.111-21, 1985.

BICK, E. Automatic parsing of portuguese. **Anais do II Encontro para o Processamento Computacional de Português Escrito e Falado**, p.101-7, Curitiba, 21-2 de outubro de 1996.

BIEWER, A.; FÉNEYROL, C.; RITZKE, J.; STEGENTRITT, E. ASCOF - a modular multilevel system for French-German translation **Computational Linguistics**, v.11, n.2-3, p.137-54, 1985.

BRILL, E. **A corpus-based approach to language learning**. Thesis (PhD) - Department of Computer and Information Science of the University of Pennsylvania, Philadelphia, 1993, 154 p.

BROWN, P.F.; COCKE, J.; DELLA PIETRA, S.A.; DELLA PIETRA, V.J.; JELINEK, F.; LAFFERTY, J.D.; MERCER, R.L.; ROOSSIN, P.S. A statistical approach to machine translation. **Computational Linguistics**, v. 16, n.2, p.79-85, June 1990.

CHARNIAK, E. **Statistical language learning**. MIT Press, 1993.

CHIN, E. **Tradução por computador: dicionário e componentes de análise e transferência**. Tese de doutoramento – Departamento de Lingüística da Faculdade de Filosofia, Letras e Ciências Humanas da Universidade de São Paulo, São Paulo, 1996, 330 p.

DAELEMANS, W.; ZAVREL, J.; BERCK, P.; GILLIS, S. MBT: A memory-based part of speech tagger-generator. In **Proceedings WVLC**, 1996. Copenhagen.

DAELEMANS, W.; BOSCH, A.; WEIJTERS, T. IGTree: Using Trees for Compression and Classification in Lazy Learning Algorithms. **Artificial Intelligence Review**, special issue on Lazy Learning. D. Aha (ed.), 1996.

DARPA/ITO Research Areas. Human Language Systems Program Applying Statistical Methods to Machine Translation. **The International Business Machines Corporation**. http://www.ito.darpa.mil/Summaries95/7794--IBM.html, 21 Aug. 95.

ISABELLE, P.; BOURBEAU, L. TAUM-AVIATION: its technical features and some experimental results. **Computational Linguistics**, v.11, n.1, p.18-27, 1985.

JOSÉ NETO, J. Adaptative automata for context-dependent languages. **ACM SIGPLAN Notices**, v.29, n.9, p.115-24, Sept. 1994.

JOSÉ NETO, J. **Introdução à compilação**. Rio de Janeiro, LTC - Livros Técnicos e Científicos Editora S.A., 1987.

KARTTUNEN, L. Constructing Lexical Transducers. In **The Proceedings of the 15th International Conference on Computational Linguistics**. Coling 94, I, p.406-11, Aug. 5-9, 1994. Kyoto, Japão.

KINOSHITA, J. **Aspectos de implementação de uma ferramenta de auxílio à tradução inglês-português**. Tese de Doutoramento – Departamento de

Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo, São Paulo, 1997, 163 p.

KOSKENNIEMI, K. Representations and finite-state components in natural language, p.99-116. In ROCHE, E.; SCHABES, Y. (Eds.) – **Finite-state language processing**. MIT Press, 1997.

MARCUS, M.; SANTORINI, B.; MARCINKIEWICZ, M. Building a large annotaded corpus of English: the Penn Treebank. **Computational Linguistics**, v. 19, n.2, p.313-30, 1993.

MITCHELL, T. **Machine Learning**. McGraw-Hill, 1997.

PEREIRA, J. **Ambiente integrado de desenvolvimento de reconhecedores sintáticos, baseado em autômatos adaptativos**. Dissertação de Mestrado – Departamento de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo, São Paulo, Maio de 1999.

PPCME Penn-Helsinki Parsed Corpus of Middle English. **Departamento de Lingüística da Universidade da Pensilvânia**, EUA, http://www.ling.upenn.edu/mideng, 1998.

REYLE, U.; ROHRER, C. Introduction, p.1-17. In REYLE, U.; ROHRER, C. (Eds.) - **Natural language parsing and linguistic theories**. Kluwer Academic Publishers, 1988.

ROCHE, E.; SCHABES, Y. Deterministic part-of-speech tagging with finite-state transducers, p.205-39. In ROCHE, E.; SCHABES, Y. (Eds.) – **Finite-state language processing**. MIT Press, 1997.

RUSSEL, S.; NORVIG, P. **Artificial Inteligence: a modern approach**. New Jersey, Prentice-Hall, 1995.

SCOTT, B. The five layers of ambiguity. **Byte: the small systems journal**, v.18, n.1, p.174-5, Mcgraw-Hill Publications, Jan. 1993.

SLOCUM, J. A survey of machine translation: its history, current status, and future prospects. **Computacional Linguistics**, v.11, n.1, p.1-17, Jan./Mar. 1985.

SUERETH, R. **Developing natural language interfaces: processing human conversations**. McGraw-Hill, 1997.

TBCHP Tycho Brahe Parsed Corpus of Historical Portuguese. **Instituto de Estudos da Linguagem**, UNICAMP, SP, http://www.ime usp.br/~tycho/corpus, 1998.

VASCONCELLOS, M. Machine translation. **Byte: the small systems journal**, v.18, n.1, p.153-64, Mcgraw-Hill Publications, Jan. 1993.

VAUQUOIS, B.; BOITET, C. Automated translation at Grenoble University. **Computational Linguistics**, v.11, n.1, p.28-36, 1985.

VILLAVICENCIO, A.; MARQUES, N.; LOPES, G.; VILLAVICENCIO, F. Part-of-Speech Tagging for Portuguese Texts Introduction, p.323-32. In WAINER, J.; CARVALHO, A. (Eds.) – Lecture Notes in Artificial Intelligence 991 – Advances in Artificial Intelligence – 12th Brazilian Symposium on Artificial Intelligence, SBIA'95, Campinas, Brazil, Oct. 10-12, 1995 – Springer-Verlag.

Apêndice 1

A1.1 Interpretação de alguns termos específicos do Processamento de Linguagens Naturais utilizados nesta dissertação

- **AMBIGÜIDADES**

Pode-se classificar este problema em cinco níveis de abrangência [SCOTT-93]:

1. ambigüidade **sintática-lexical** (este nível de ambigüidade aparece quando o sistema busca uma palavra no seu dicionário e encontra duas ou mais categorias morfo-sintáticas possíveis para ela: por exemplo, "o", em português, pode um artigo ou um pronome, e é muito comum, na língua inglesa, palavras que tenham a função de substantivo e de verbo),
2. **sintática-sentencial** (este tipo de ambigüidade refere-se, por exemplo, à indeterminação de a qual palavra se subordina um sintagma preposicional; em "João viu o homem com o telescópio", não se sabe se "com o telescópio" liga-se ao verbo, significando que João usou o telescópio para ver o homem, ou ao objeto, significando que João viu o homem carregando o telescópio),
3. **semântica-lexical** (refere-se ao fenômeno da polissemia, ou seja, o de que as palavras poderem ter mais de uma significação; por exemplo, o verbo estimar pode significar apreciar ou avaliar, entre outros sentidos),

4. **semântica-sentencial** (aparece em situações nas quais o significado de uma palavra depende do contexto da sentença; em [SCOTT-93] encotra-se o seguinte exemplo, em inglês: "Check the newspaper for errors. Check the newspaper for dates"; na primeira sentença, "check" significa examinar, e na segunda, consultar) e

5. **extra-sentencial** (advinda de fenômenos como elipses e anáforas; por exemplo, a quem ou a que se refere determinado pronome?).

- **MÉTODO DE TRANSFERÊNCIA**

Neste método a tradução é realizada em três etapas:

1. **análise**, na qual uma sentença na entrada do sistema é transformada numa árvore sintática;

2. **transferência**, na qual regras lingüísticas e léxicas são aplicadas a esta árvore, para que esta obedeça à sintaxe da língua-alvo;

3. **geração**, que constrói, a partir da última árvore, uma sentença na língua-alvo.

A base de conhecimento necessária é guardada em gramáticas, léxicos e regras de transferência. Como este método armazena uma forma de representação intermediária do significado das sentenças, a qualidade da tradução produzida por ele é muito melhor em comparação com o método direto, que traduz por substituir palavras da língua-fonte por equivalentes na língua-alvo (a ciência da tradução procura traduzir sentidos, e não palavras, entre as línguas). Contudo, este método é mais sintático que semântico, fazendo com que a qualidade da tradução não seja tão boa em algumas situações. Quanto mais sintático for o sistema tradutor, mais resquícios da estrutura da língua-fonte permanecem na tradução, fazendo com que esta soe de maneira estranha para falantes nativos da língua-alvo, mesmo sendo perfeitamente inteligível.

- **CORPUS LINGÜÍSTICO**

Corpus é um "*conjunto de documentos, dados e informações sobre determinada matéria*" (Novo Dicionário da Língua Portuguesa, 2ª edição, revista e

aumentada, Aurélio Buarque de Holanda Ferreira). Quando se constrói um corpus de textos para uso em processamento de linguagens naturais, procura-se fazê-lo de modo que seja formado por textos que representem bem o domínio da aplicação em questão.

- **TEORIA DA COMUNICAÇÃO APLICADA À MODELAGEM DO PROCESSO DE TRADUÇÃO**

A tradução pode ser modelada da seguinte maneira [BROWN-90]: o chamado *modelo da tradução* encarrega-se de sugerir palavras da língua-alvo que poderiam ter produzido as palavras que se observam na sentença-fonte, e o *modelo da língua-alvo*, de sugerir uma ordem na qual ordenar as palavras da sentença-alvo. Este "sugerir" engloba o cálculo da palavra ou da ordem com maior probabilidade.

Antes de se poder traduzir, há a necessidade de calcular todos os parâmetos (tipicamente, dezenas de milhões de números em ponto flutuante) de ambos os modelos, através de um processo automático de treinamento que colhe informações de um corpus. Para o cálculo dos parâmetros do *modelo da língua-alvo*, basta o treinamento estatístico sobre um corpus monolíngüe (na língua-alvo). Já para a determinação dos parâmetros do *modelo da tradução*, há a necessidade de treinamento com base em um corpus bilíngüe, com sentenças na língua-fonte alinhadas com sentenças na língua-alvo.

No primeiro experimento em tradução, um texto em francês foi traduzido para inglês: 48% das sentenças foram consideradas traduções razoáveis. Esta pesquisa foi continuada por um grupo da IBM, criando o sistema tradutor *Candide* [MARCUS-93, DARPA/ITO-95].

- **TEXTOS LIVRES**

Textos livres referem-se a textos reais, escritos por seres humanos sem nenhuma restrição, em oposição a sentenças criadas para serem exemplos simples e didáticos.

- **FORMALISMO DE UNIFICAÇÃO**

São muitas as variantes das gramáticas de unificação, as quais têm sido desenvolvidas especialmente desde o começo da década de 80. Entre as mais conhecidas estão PATR, PATR-II, LFG (*Lexical Functional Grammar*), GPSG (*Generalized Phrase Structure Grammar*) e HPSG (*Head-driven Phrase Structure Grammar*).

A idéia básica por detrás destas gramáticas é que as características (ou traços, como preferem os lingüistas), responsáveis pela representação da subcategorização, são melhor representadas por estruturas hierárquicas, não apenas por subcategorias atômicas [REYLE-88].

- **ALGORITMO DE BUSCA HEURÍSTICA DO TIPO "SUBIDA AO MONTE PELO MAIOR ACLIVE"**

É um conhecido algoritmo de busca heurística aplicado a problemas que possuem um conjunto de soluções muito grande, e deseja-se encontrar a **solução ótima** (a melhor solução possível); é obrigatório a existência de uma **função de avaliação**, a qual indica numericamente um grau de qualidade para a solução atual.

Este algoritmo pressupõe que se parta de uma solução qualquer (mesmo que esta esteja distante da solução ótima), que se faça pequenas modificações nesta solução, sucessivamente, objetivando a maximização da função de avaliação e a respectiva melhora da solução atual.

Diferentemente de outros algoritmos de busca, ele não mantém registro de estados ou soluções anteriores.

Seu nome faz alusão a uma representação gráfica (Figura 58), na qual o conjunto de estados ou soluções é acomodado em uma superfície plana e a função de avaliação alocada em um eixo perpendicular a este plano. Deste modo, o problema seria resumido em caminhar-se a partir de um ponto inicial qualquer com o objetivo de encontrar o ponto mais alto de toda uma região (que representa todo o espaço das possíveis soluções), com o menor número de passos possível [RUSSEL-95].

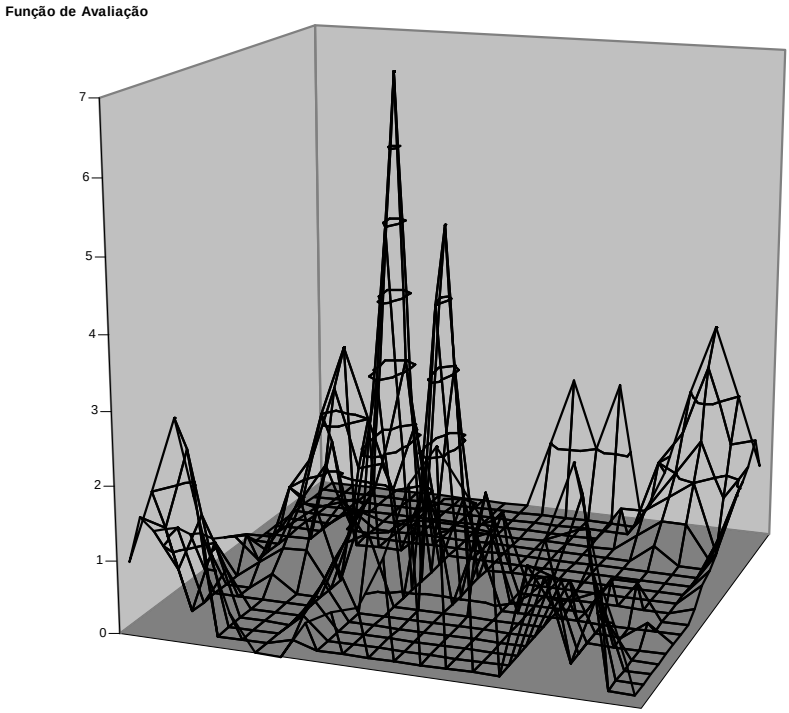


Figura 58 – Representação gráfica do algoritmo de busca do tipo "subida ao monte pelo maior aclive"

Apêndice 2

A2.1 Listagens dos módulos do etiquetador

A2.1.1 Módulo para etiquetar palavras conhecidas

```
/* Lista de transições do automato original */
transicao(1, 2, simbolo, 0, fpre1, nada, 0).
transicao(1, 1, separador, 0, nada, nada, 0).

/* Lista das funcoes adaptativas */

nada(X,Y,Z):-  true.

/* Responsável pelo crescimento da árvore do léxico apenas */
fpre1(X,Y,Z):-
       gerador(Novo1),
       gerador(Novo2),
       eliminacao(X,     Y,     simbolo,   0, fpre1, nada, 0),
       insercao(X,     Novo1, Z,         0, nada,  nada, 0),
       insercao(X,     Novo2, simbolo,   0, fpre1, nada, 0),
       insercao(Novo1, Y,     simbolo,   0, fpre1, nada, 0),
/* Se agora vier uma etiqueta: */
       insercao(Novo1, 1,     barra,     0, fpre2, nada, 0),
/* Se a palavra não tiver etiqueta: */
       insercao(Novo1, 1,     separador, 0, fpre5, nada, 0).

/* Responsável pela aquisição da 1a. etiqueta de uma palavra */
fpre2(X,Y,Z):-
       gerador(Novo1),
       gerador(Novo2),
       eliminacao(X,     Y,     barra,     0, fpre2, nada, 0),
       insercao(X,     Novo1, Z,         0, nada,  nada, 0),
       insercao(Novo1, Novo2, etiqueta,  0, fpre3, nada, 0),
       insercao(Novo2, 1,     separador, 0, nada,  nada, 0).
```

```
/* Esta função , um caso particular de fpre4 */
fpre3(X,Y,Z):-
       eliminacao(X,     Y,     etiqueta,  0, fpre3, nada,  0),
       insercao(X,     Y,     etiqueta,  0, fpre4, nada,  0),
       insercao(X,     Y,     Z        , 0, nada,  fpos1, 0).

/* Adquire uma etiqueta e prepara o autômato para receber uma outra etiqueta */
fpre4(X,Y,Z):-
       gerador(Novo1),
       eliminacao(X,     Y,     etiqueta,  0, fpre4, nada,  0),
       insercao(X,     Novo1, vazio,     0, nada,  nada,  0),
       insercao(Novo1, Y,     etiqueta,  0, fpre4, nada,  0),
       insercao(Novo1, Y,     Z,         0, nada,  fpos1, 0).

fpre5(X,Y,Z):-
/* Decisão */
      ( /*1a. poss.: não há exemplo de etiq. p/ esta palavra */
        (not(existe_transicao(X, A, [47|_], 0, nada, nada))
        );
        /* 2a. poss.: vamos sugerir todas as etiq. possíveis */
        (existe_transicao(X, A, [47|_], 0, nada, nada),
         /* C é a etiqueta que tem maior freqüência */
         inspecao(A, B, C, 0, nada, fpos1, _),
         eliminacao(X, Y, separador, 0, fpre5, nada, 0),
         insercao(X, Y, vazio, [47,C], nada, fpos2, 0),
         fx(X, Y, A)
        )
      ).

fx(X,Y,A):-
      atualize_banco_de_dados,
/* Decisão */
      (/*1a. poss.: não há mais ex. de etiq. p/ esta palavra */
        (not(existe_transicao(A, B, vazio , 0, nada, nada))
        );
       /* 2a. poss.: vamos sugerir as outras etiq. possiveis */
        (existe_transicao(A, B, vazio , 0, nada, nada),
         /* C é a etiqueta */
         inspecao(B, _, C, 0, nada, fpos1, _),
         inspecao(X, Y, vazio, Z, nada, fpos2, 0),
         eliminacao(X, Y, vazio, Z, nada, fpos2, 0),
         insercao(X, Y, vazio, [Z,47,C], nada,  fpos2, 0),
         fx(X, Y, B)   /* Recursivamente */
        )
      ).

fpos1(X,Y,Z):-
/* Apenas para atualizar o contador: */
       eliminacao(X, Y, Z, 0, nada, fpos1, N),
       soma_um(N, N2),
/* Ramo do autômato anterior a este: */
       inspecao(A, X, B, 0, nada, nada, 0),
  /* Decisão */
     ( /* 1a. poss.: esta é a etiqueta com maior freqüência */
```

```
        (barra(B),    /* B é uma barra? */
         insercao(X, Y, Z, 0, nada, fpos1, N2)
        );
        /* 2a. poss.: esta NÃO é a etiqueta com maior freq. */
        (B == vazio,
/* Etiqueta que tinha freqüência maior ou igual que esta: */
         inspecao(A, Y, C, 0, nada, fpos1, N3),

        /* Decisão */
         (/* 1a: a etiq. ant. ainda tem freq. maior que esta */
          (N3 >= N2,
/* Nada de especial é feito */
           insercao(X, Y, Z, 0, nada, fpos1, N2)
          );
          /* 2a: a etiq. anterior agora tem freqüência menor */
          (N3 < N2,
/* Troca de posições entre esta etiqueta e a anterior: */
           /* retira a anterior: */
           eliminacao(A, Y, C, 0, nada, fpos1, N3),
           /* insere esta no lugar da anterior: */
           insercao(A, Y, Z, 0, nada, fpos1, N3),
/* A princípio, poder-se-ia esperar que acima fosse usado N2. Porém, com a
chamada recursiva a seguir, este contador será incrementado posteriormente, na
última chamada */
           /* insere a anterior no lugar desta: */
           insercao(X, Y, C, 0, nada, fpos1, N3),
/* Propagação deste incremento, com a finalidade de manter a lista de etiquetas
ordenada: */
           atualize_banco_de_dados,
           fpos1(A, Y, Z)  /* Recursivamente */
          )
         )
        )
      ).

fpos2(X,Y,Z):-
      eliminacao(X,     Y,     vazio    , W, nada,  fpos2, 0),
      insercao(  X,     Y,     separador, 0, fpre5, nada,  0).
```

A2.1.2 Módulo para etiquetar palavras desconhecidas

```
/* Lista de transições do automato original */
transicao(1, 1, separador, 0, nada,  nada, 0).
transicao(1, 2, simbolo,   0, fpre1, nada, 0).

/* Lista das funcoes adaptativas */
nada(X,Y,Z):-  true.

/*Responsável por aprender a última letra de uma série de letras do sufixo */
fpre1(X,Y,Z):-
      gerador(Novo1),
```

```
      gerador(Novo2),
      eliminacao(X,     Y,     simbolo,  0, fpre1, nada,  0),
      insercao(  X,     Y,     Z,        0, nada,  fpos2, 0),
        /* fpos2 serve apenas para adicionar o contador */
      insercao(  Y,     Novo1, simbolo,  0, fpre2, nada,  0),
      insercao(  Y,     1,     barra,    0, fpre3, nada,  0),
      insercao(  X,     Novo2, simbolo,  0, fpre1, nada,  0).

/* Responsável por aprender a penúltima e as outras de uma série de letras do
sufixo */
fpre2(X,Y,Z):-
      gerador(Novo1),
      eliminacao(X,     Y,     simbolo,   0, fpre2, nada,  0),
      insercao(  X,     Y,     Z,         0, nada,  fpos1, 0),
      insercao(  X,     3,     simbolo,   0, fpre4, nada,  0),
/* fpre4 é variante de fpre2. O estado 3 será usado na fase de aplicação */
      insercao(  Y,     1,     barra,     0, fpre3, nada,  0),
      insercao(  Y,     Novo1, simbolo,   0, fpre2, nada,  0).

/* Esta função é um caso particular da fpre2. Responsável por aprender uma
outra letra, nas posições da segunda, terceira, etc letra */
fpre4(X,Y,Z):-
      gerador(Novo1),
      gerador(Novo2),
      gerador(Novo3),
      eliminacao(X,     Y,     simbolo,   0, fpre4, nada,  0),
      insercao(  X,     Novo1, vazio,     0, nada,  nada,  0),
      insercao(  Novo1, Novo2, Z,         0, nada,  fpos1, 0),
      insercao(  Novo1, Y    , simbolo,   0, fpre4, nada,  0),
      insercao(  Novo2, 1,     barra,     0, fpre3, nada,  0),
      insercao(  Novo2, Novo3, simbolo,   0, fpre2, nada,  0).

/*Responsável pela aquisição da 1a. etiqueta de um sufixo*/
fpre3(X,Y,Z):-
      gerador(Novo1),
      gerador(Novo2),
      eliminacao(X,     Y,     barra,     0, fpre3, nada, 0),
      insercao(  X,     Novo1, Z,         0, nada,  nada, 0),
      insercao(  Novo1, Novo2, etiqueta,  0, fpre5, nada, 0),
      insercao(  Novo2, 1,     separador, 0, nada,  nada, 0).

/* Esta função é um caso particular de fpre6 */
fpre5(X,Y,Z):-
      eliminacao(X,     Y,     etiqueta,  0, fpre5, nada,  0),
      insercao(  X,     Y,     etiqueta,  0, fpre6, nada,  0),
      insercao(  X,     Y,     Z        , 0, nada,  fpos1, 0).

/* Adquire uma etiqueta e prepara o autômato para receber uma outra etiqueta */
fpre6(X,Y,Z):-
      gerador(Novo1),
      eliminacao(X,     Y,     etiqueta,  0, fpre6, nada,  0),
      insercao(  X,     Novo1, vazio,     0, nada,  nada,  0),
      insercao(  Novo1, Y,     etiqueta,  0, fpre6, nada,  0),
      insercao(  Novo1, Y,     Z,         0, nada,  fpos1, 0).
```

```
/* Coloca as etiquetas associadas aos sufixos em ordem decrescente de freqüência */
fpos1(X,Y,Z):-
      eliminacao(X, Y, Z2, W, nada, fpos1, N), /* Apenas para atualizar o contador */
      soma_um(N, N2),
      inspecao(A, X, B, _, _, _, _), /* Ramo anterior a este do autômato */
  /* Decisão */
      (/* 1a. poss.: esta é a etiqueta com maior freq. */
        (B \== vazio,    /* B nao é vazio */
         insercao(X, Y, Z2, W, nada, fpos1, N2)
        );
       /* 2a. poss.: esta NÃO é a etiqueta com maior freq. */
        (B == vazio,
         inspecao(A, K, C, W, nada, fpos1, N3), /* Etiqueta que tinha freqüência maior ou igual que esta */
         /* Decisão */
          (/* 1a. poss.: a etiqueta anterior ainda tem freqüência maior que esta */
            (N3 >= N2,
             insercao(X, Y, Z2, W, nada, fpos1, N2)
             );
           /* 2a. poss.: a etiqueta anterior agora tem freqüência menor */
            (N3 < N2,
             /* Devo fazer uma troca de posições entre esta etiqueta e a anterior */
             eliminacao(A, K, C,  W, nada, fpos1, N3), /* retira a anterior */
             insercao(A, Y, Z2, W, nada, fpos1, N3), /* insere esta no lugar da anterior */
              /* A princípio poderia-se esperar que acima fosse usado N2. Porém, com a chamada recursiva a seguir, este contador será incrementado posteriormente, na última chamada */
             insercao(X, K, C, W, nada, fpos1, N3), /* insere a anterior no lugar desta */
              /* Propagação deste incremento, com a finalidade de manter a lista de etiquetas ordenada */
             atualize_banco_de_dados,
             fpos1(A, Y, Z)  /* Recursivamente */
            )
          )
        )
      ).

/* Apenas para atualizar o contador das transições correspondentes às últimas letras dos sufixos. */
fpos2(X,Y,Z):-
      eliminacao(X, Y, Z, A, B, C, N), /* Apenas para atualizar o contador */
      soma_um(N, N2),
      insercao(X, Y, Z, A, B, C, N2).

/*  Funções que serão usadas somente na fase de aplicação */
```

```
/* Usada na fase de aplicação, com transição por símbolo. Apenas coloca entre
as transições 6 e 7 todas as etiquetas do sufixo mais longo encontrado. */
fpreAcheMaisProvaveis(X, Y, Z):-
/* Decisão */
      (/* 1a. poss.: Voltando, se possível, uma transição por uma letra */
        (existe_transicao(J, X, K, 0, nada, fpos1),
         C = J
        );
       /* 2a. possibilidade: Não consegui voltar  */
        (not(existe_transicao(J, X, K, 0, nada, fpos1)),
         C = X
        )
      ),
      fAchePrimeiroVazio(C, A),
      fAchePrimeiroExemplo(A, B),
      eliminacao(6, 7, vazio, [47,"N"], nada, nada, 0), /* Entre 6 e 7 ficam as
tags mais prováveis */
      fCopieTags(B, 6, Fim),
      insercao(Fim, 7, vazio, 0, nada, nada, 0).

fAchePrimeiroVazio(X, A):-
  /* Decisão */
      (/* 1a. poss.: Voltando, se possível, uma transição em vazio */
        (existe_transicao(J, X, vazio, 0, nada, nada),
         fAchePrimeiroVazio(J, A)
        );
       /* 2a. possibilidade: Cheguei onde queria        */
        (not(existe_transicao(J, X, vazio, 0, nada, nada)),
         A = X
        )
      ).

fAchePrimeiroExemplo(X, A):-
  /* Decisão */
      (/* 1a. poss.: Há um exemplo, ao menos, deste sufixo */
        (existe_transicao(X, B, bloqueio, 0, nada, nada),
         A = B
        );
       /* 2a. possibilidade: Não há um exemplo deste sufixo */
        (not(existe_transicao(X, B, bloqueio, 0, nada, nada)),
         inspecao(X, C, D, 0, nada, fpos1, _),
         fAchePrimeiroExemplo(C, A)
        )
      ).

fCopieTags(X, Y, Fim):-
      inspecao(X, A, B, 0, nada, fpos1, _),
      gerador(Novo1),
      insercao(Y, Novo1, vazio, [47, B], nada, nada, 0),
  /* Decisão */
      ( /* 1a. possibilidade: Há outro exemplo */
        (existe_transicao(X, C, vazio, 0, nada, nada),
         fCopieTags(C, Novo1, Fim)
        );
```

```
        /* 2a. possibilidade: Não há outro exemplo */
        (not(existe_transicao(X, C, vazio, 0, nada, nada)),
         Fim = Novo1
        )
      ).

/* Apenas coloca as trans. entre 6 e 7 no caminho para que sejam executadas. */
fpreEscrever(X, Y, Z):-
/* Sei que X vale 3 e Y vale 1 */
      eliminacao(X, Y, separador, 0, fpreEscrever, nada,        0),
      insercao(  X, 6, vazio,     0, nada,         nada,        0),
      insercao(  7, Y, vazio,     0, nada,         fposAjeitar, 0).

/* "Arruma a casa" novamente: apaga tudo entre as transições 6 e 7; idem para
as transições inseridas em fpreEscrever. Restaura a transição default entre 6 e
7 e também entre 3 e 1.*/
fposAjeitar(X, Y, Z):-
      eliminacao(3, 6, vazio,   0,        nada,         nada,        0),
      eliminacao(7, 1, vazio,   0,        nada,         fposAjeitar, 0),
      fApagueTudo(6, 7),
      insercao(  6, 7, vazio,   [47,"N"], nada,         nada,        0),
/* N é a Tag Default */
      insercao(3, 1, separador, 0,        fpreEscrever, nada,        0).

/* "Arruma a casa" novamente: apaga tudo entre os estados 6 e 7. Restaura a
transição default entre 6 e 7. */
fposAjeitar2(X, Y, Z):-
      fApagueTudo(6, 7),
      insercao(6, 7, vazio,     [47,"N"], nada,          nada, 0). /* Esta é a
Tag Default */

fApagueTudo(X, Y):-
      eliminacao(X, A, _, _, _, _, _),
  /* Decisão */
      ( /* 1a. possibilidade: existe mais a apagar     */
        (A \== Y,
         fApagueTudo(A, Y)
        );
        /* 2a. possibilidade: não existe mais a apagar     */
        (A == Y
        )
      ).

/* Usada na fase de aplicação, com transição por separador. */
fpreAcheMaisProvaveis2(X, Y, Z):-
      eliminacao(X, Y, separador, 0, fpreAcheMaisProvaveis2, nada,        0),
      fpreAcheMaisProvaveis(X, Y, Z),
      insercao(  X, Y, vazio,     0, nada,                   fposVoltar,  0).

/* Recoloca a transição através de separador entre X e Y. */
fposVoltar(X, Y, Z):-
      eliminacao(X, Y, vazio,     0, nada,                   fposVoltar, 0),
      insercao(  X, Y, separador, 0, fpreAcheMaisProvaveis2, nada,       0).
```

A2.1.3 Módulo para fazer refinamentos contextuais

```
/* Lista de transições do automato original */
transicao(1, 2, etiqueta, 0, fpre1, nada, 0).
transicao(2, 3, etiqueta, 0, fpre2, nada, 0).

/* Lista das funcoes adaptativas */

nada(X,Y,Z):-  true.

/*Responsável por aprender a primeira etiqueta de uma série de três*/
fpre1(X,Y,Z):-
      gerador(Novo1),
      gerador(Novo2),
      eliminacao(X,     Y,     etiqueta,  0, fpre1, nada,  0),
      insercao(  X,     Y,     Z,         Z, nada,  fpos3, 0),
  /* Certamente estou no estado 1, mas vamos chamá-lo de X. fpos3 serve apenas
para adicionar o contador      */
      insercao(  X,     Novo1, etiqueta,  0, fpre1, nada,  0),
      insercao(  Novo1, Novo2, etiqueta,  0, fpre2, nada,  0).

/* Responsável por aprender a segunda etiqueta de uma série de três */
fpre2(X,Y,Z):-
      gerador(Novo1),
      gerador(Novo2),
      eliminacao(X,     Y,     etiqueta,  0, fpre2, nada,  0),
      insercao(  X,     Novo1, etiqueta,  0, fpre4, nada,  0),      /* fpre4 é
variante de fpre2 */
      insercao(  X,     Novo2, conj_tag,  0, fpre6, nada,  0),     /* fpre6 faz
parte do módulo de aplicação */
      insercao(  X,     Y,     Z,         0, nada,  fpos1, 0),

  /* Decisão */
      ( /* 1a. poss.: não existe transição saindo do estado 1, com estimulo Z e
escrevendo Z no stream de saída         */
        (not(existe_transicao(1, A, Z, Z, nada, fpos3)),
         gerador(Novo3),
         gerador(Novo4),
         insercao(1,     Novo3, Z,         Z, nada,  fpos3, 0),
         insercao(Novo3, Novo4, etiqueta,  0, fpre2,  nada, 0),
         insercao(Y,     Novo3, vazio,     Z, fpre3,  nada, 0)
        );
        /* 2a. poss.: existe transição saindo do estado 1, com estímulo Z e
escrevendo Z no stream de saída         */
        (existe_transicao(1, A, Z, Z, nada, fpos3),
         insercao(Y,     A,     vazio,     Z, fpre3,  nada, 0)
        )
      ).

/*Responsável por aprender a terceira etiqueta de uma série de três*/
fpre3(X,Y,Z):-
      gerador(Novo1),
      eliminacao(X,     Y,  vazio,      A, fpre3, nada,  0),
```

```
      insercao(X,     Y,  lap(Z),     A, nada,  fpos1, 0), /* fpos1 serve para
manter em ordem decrescente de freqüência as transições */
      insercao(X, Novo1,  etiqueta,   0, fpre5, nada,  0),
      insercao(X,     Y,  conj_tag,   A, fpre7, nada,  0).

/* Esta função é um caso particular da fpre2. Responsável por aprender uma
outra etiqueta, na segunda posição, de uma série de três */
fpre4(X,Y,Z):-
      gerador(Novo1),
      gerador(Novo2),
      eliminacao(X,     Y,     etiqueta,  0, fpre4, nada,  0),
      eliminacao(X,     K,     conj_tag,  0, fpre6, nada,  0),
      insercao(  X,     Novo1, vazio,     0, nada,  nada,  0),
      insercao(  Novo1, Novo2, Z,         0, nada,  fpos1, 0),
      insercao(  Novo1, Y    , etiqueta,  0, fpre4, nada,  0),
      insercao(   Novo1, K,      conj_tag,  0, fpre6, nada,  0), /* Módulo de
Aplicação */

  /* Decisão */
      ( /* 1a. poss.: não existe transição saindo do estado 1, com estímulo Z e
escrevendo Z no stream de saída         */
        (not(existe_transicao(1, A, Z, Z, nada, fpos3)),
         gerador(Novo3),
         gerador(Novo4),
         insercao(1,     Novo3, Z,         Z,      nada,  fpos3, 0),
         insercao(Novo3, Novo4, etiqueta,  0, fpre2, nada, 0),
         insercao(Novo2, Novo3, vazio,     Z, fpre3,  nada, 0)
        );
        /* 2a. poss.: existe transição saindo do estado 1, com estímulo Z e
escrevendo Z no stream de saída         */
        (existe_transicao(1, A, Z, Z, nada, fpos3),
         insercao(Novo2, A,     vazio,     Z, fpre3,  nada, 0)
        )
      ).

/* Esta função é um caso particular da fpre3. Responsável por aprender uma
outra etiqueta, na terceira posição, de uma série de três */
fpre5(X,Y,Z):-
      gerador(Novo1),
      eliminacao(X,     Y,  etiqueta,  0, fpre5,  nada,  0),
      eliminacao(X,     K,  conj_tag,  A, fpre7,  nada,  0),
      insercao(  X,     Y,     vazio,  0,  nada,  nada,  0),
      insercao(   Y,      K,     lap(Z),   A,   nada, fpos1,   0), /* fpos1 serve
para manter em ordem decrescente de freqüência as transições */
      insercao(  Y, Novo1,  etiqueta,  0, fpre5,  nada,  0),
      insercao(Y,     K,  conj_tag,  A, fpre7,  nada,  0).

/* Esta função é usada no processo de Aplicação do conhecimento adquirido. Ela
pesquisa a opção mais provável para tirar a ambigüidade de um conjunto de
etiquetas possíveis, através do contexto (etiqueta precedente e posterior). */
fpre6(X,Y,Z):-
      gerador(Novo1),
      gerador(Novo2),
      eliminacao(X,     Y,     conj_tag,  0, fpre6, nada,  0),
```

```
      eliminacao(X,     K,     etiqueta,  0, fpre4, nada,  0),
      insercao(  X,     Novo1, vazio,     0, nada,  nada,  0),
      insercao(Novo1, Novo2, Z,         0, nada,  fpos1, 0),
      insercao(Novo1, Y,       conj_tag,   0, fpre6, nada,   0), /* Módulo de
Aplicação */
      insercao(Novo1, K,     etiqueta,  0, fpre4, nada,  0),
      fAchePrimeiro(X, A), /* Procura o estado - A - em que se iniciam os
exemplos de segundo "nível" referentes à etiqueta de primeiro "nível" entrada
imediatamente antes */
      fMonteTransicoes(A, X, Novo2, Z). /* Constrói toda a estrutura de
Aplicação */

fAchePrimeiro(X, A):-
  /* Decisão */
      (   /* 1a. poss.: Voltando, se possível, uma transição em vazio */
        (existe_transicao(J,   X,    vazio,       0, nada,  nada),
         fAchePrimeiro(J, A)
        );
          /* 2a. possibilidade: Cheguei onde queria        */
        (not(existe_transicao(J,   X,    vazio,       0, nada,  nada)),
         A = X
        )
      ).

fMonteTransicoes(Primeiro, Ultimo, Destino, Ambigua):-
      inspecao (Primeiro, A,    B,   0, nada,  fpos1,  _),
/* Decisão */
      (   /* 1a. possibilidade:         */
        (pertence(B, Ambigua),
         fCopieTransicoes(A, Destino, OndeParou),
        );
          /* 2a. possibilidade:         */
        (not(pertence(B, Ambigua)),
         OndeParou = Destino,
        )
      ),
  /* Outra Decisão */
      (   /* 1a. possibilidade: Não acabei de montar as transições */
        (Primeiro \== Ultimo,
         inspecao(Primeiro, Segundo, vazio, 0, nada, nada, 0),
         fMonteTransicoes(Segundo, Ultimo, OndeParou, Ambigua),
        );
          /* 2a. possibilidade: Acabei de montar as transições */
        (Primeiro == Ultimo,
/* Agora inserirei a opção default como sendo aquela que é a mais freqüente  */
         fOpcaoDefault(OndeParou, Ambigua),
        )
      ), !.

fCopieTransicoes(DeOnde, ParaOnde, OndeParou):-
      gerador(Novo1),
      inspecao(DeOnde, A, lap(B), C, D, E, F),
      insercao(ParaOnde, A, lap(B), C, D, fposRemoveTudo, F),
      insercao(ParaOnde, Novo1, vazio, 0, nada, nada, 0),
```

```
  /* Decisão */
      (/* 1a. p.:existe trans. em vazio; isto é sinal que há mais que copiar */
        (existe_transicao(DeOnde, NovoDeOnde, vazio, 0, nada, nada),
         fCopieTransicoes(NovoDeOnde, Novo1, OndeParou)
        );
/* 2a. p.: não existe trans. em vazio; isto é sinal que não há mais o que
copiar */
        (not(existe_transicao(DeOnde, NovoDeOnde, vazio, 0, nada, nada)),
         OndeParou = Novo1
        )
      ), !.

pertence([B], [HAmb|TAmb]):-
/* Decisão */
      ( /* 1a. poss.: B é equivalente à primeira etiqueta da lista  */
        (equivalente(B, HAmb)
        );
        /* 2a. poss.: B é equivalente à alguma etiqueta do resto da lista */
        (pertence([B], TAmb)
        )
      ), !.

/* Agora inserirei a opção default como sendo aquela que é a mais freqüente */
/* [HAmb] é a etiqueta mais provável da lista de etiquetas */
fOpcaoDefault(OndeParou, [HAmb|TAmb]):-
/* Decisão */
      ( /* 1a. poss.: não existe transição saindo do estado 1, com estímulo
[HAmb] e escrevendo [HAmb] no stream de saída */
        (not(existe_transicao(1, A, [HAmb], [HAmb], nada, fpos3)),
         gerador(Novo1),
         gerador(Novo2),
         insercao(1,        Novo1, [HAmb],    [HAmb], nada,  fpos3,         0),
         insercao(Novo1,    Novo2, etiqueta,  0,      fpre2, nada,          0),
         insercao(OndeParou,Novo1, vazio,     [HAmb], nada,  fposRemoveTudo,0)
        );
        /* 2a. poss.: existe transição saindo do estado 1, com estímulo [HAmb]
e escrevendo [HAmb] no stream de saída */
        (existe_transicao(1, A, [HAmb], [HAmb], nada, fpos3),
         insercao(OndeParou,A,     vazio,     [HAmb], nada,  fposRemoveTudo, 0)
        )
      ).

fposRemoveTudo(X, Y, Z):-
      fAchePrimeiro(X, A),
      eliminacao(B, A, _,        0, _,     _,    _),
      eliminacao(B, C, etiqueta, 0, fpre4, nada, 0),
      eliminacao(B, D, conj_tag, 0, fpre6, nada, 0),
      eliminacao(E, B, vazio,    0, nada,  nada, 0),
      insercao(  E, C, etiqueta, 0, fpre4, nada, 0),
      insercao(  E, D, conj_tag, 0, fpre6, nada, 0),
      fApague(A).

fApague(A):-
      eliminacao(A, B, _, _, _, fposRemoveTudo, _),
```

```
  /* Decisão */
      ( /* 1a. possibilidade: existe mais a apagar     */
        (existe_transicao(A, C, vazio, 0, nada, nada),
         eliminacao(A, C, vazio, 0, nada, nada, 0),
         fApague(C)
        );
        /* 2a. possibilidade: nao existe mais a apagar     */
        (not(existe_transicao(A, C, vazio, 0, nada, nada))
        )
      ).

fpre7(X,Y,Z):-
      eliminacao(X,     Y,  conj_tag,  A, fpre7, nada,  0),
      insercao(  X,     Y,    lap(Z),  A,  nada, fpos2, 0).

/* Coloca as etiquetas (do segundo e terceiro "níveis" ou posições) em ordem
decrescente de freqüência        */
fpos1(X,Y,Z):-
      eliminacao(X,  Y,  Z2,  W,  nada,  fpos1,  N), /* Apenas para atualizar o
contador */
      soma_um(N, N2),
      inspecao(A, X, B, _, _, _, _), /* Ramo anterior a este do autômato */
  /* Decisão */
      ( /* 1a. possibilidade: esta é a etiqueta com maior freqüência */
        (B \== vazio,    /* B não é vazio */
         insercao(X, Y, Z2, W, nada, fpos1, N2)
        );
        /* 2a. possibilidade: esta NÃO é a etiqueta com maior freqüência */
        (B == vazio,
         inspecao(A,  K,  C,  W,  nada,  fpos1,  N3),  /*  Etiqueta  que  tinha
freqüência maior ou igual que esta */
        /* Decisão */
         ( /* 1a. poss.: a etiqueta anterior ainda tem freq. maior que esta */
           (N3 >= N2,
             insercao(X,  Y,  Z2,  W,  nada,  fpos1,  N2)  /*  Não  faço  nada  de
especial */
           );
           /* 2a. poss.: a etiqueta anterior agora tem freqüência menor */
           (N3 < N2,
      /* Devo fazer uma troca de posições entre esta etiqueta e a anterior */
             eliminacao(A, K, C,  W, nada, fpos1, N3), /* retira a anterior */
             insercao(A, Y, Z2, W, nada, fpos1, N3), /* insere esta no lugar da
anterior */
 /* A princípio poderia-se esperar que acima fosse usado N2. Porém, com a
chamada recursiva a seguir, este contador será incrementado posteriormente, na
última chamada */
             insercao(X, K, C, W, nada, fpos1, N3), /* insere a anterior no
lugar desta */
 /*  Propagação  deste  incremento,  com  a  finalidade  de  manter  a  lista  de
etiquetas ordenada */
             atualize_banco_de_dados,
             fpos1(A, Y, Z)  /* Recursivamente */
           )
         )
```

```
        )
      ).

fpos2(X,Y,Z):-
      eliminacao(X,     Y,    lap(Z),  A, nada,  fpos2, 0),
      insercao(  X,     Y,  conj_tag,  A, fpre7, nada,  0).

/* Apenas para atualizar o contador das transições correspondentes às etiquetas
de primeiro "nível".
fpos3(X,Y,Z):-
      eliminacao(X, Y, Z, A, B, C, N), /* Apenas para atualizar o contador */
      soma_um(N, N2),
      insercao(X, Y, Z, A, B, C, N2).
```