INSTITUTO MILITAR DE ENGENHARIA

CÍCERO NOGUEIRA DOS SANTOS

APRENDIZADO DE MÁQUINA NA IDENTIFICAÇÃO DE SINTAGMAS NOMINAIS: O CASO DO PORTUGUÊS BRASILEIRO

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Ciências em Sistemas e Computação.

Orientadora: Prof^a. Cláudia Maria Garcia M. de Oliveira - Ph.D

Rio de Janeiro
2005

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80-Praia Vermelha
Rio de Janeiro-RJ CEP 22290-270

S237a Santos, Cícero Nogueira dos
Aprendizado de máquina na identificação de sintagmas nominais: o caso do português brasileiro / Cícero Nogueira dos Santos. - Rio de Janeiro : Instituto Militar de Engenharia, 2005.
104 p.: il, tab.

Dissertação (mestrado) - Instituto Militar de Engenharia- Rio de Janeiro, 2005

1. Inteligência Artificial - aprendizado de máquina. 2. Aprendizado baseado em transformações. 3. Identificação de Sintagmas Nominais. I. Santos, Cícero Nogueira dos II. Instituto Militar de Engenharia. III. Título.

CDD 006.31

INSTITUTO MILITAR DE ENGENHARIA

CÍCERO NOGUEIRA DOS SANTOS

APRENDIZADO DE MÁQUINA NA IDENTIFICAÇÃO DE SINTAGMAS NOMINAIS: O CASO DO PORTUGUÊS BRASILEIRO

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Ciências em Sistemas e Computação.

Orientadora: Prof^a. Cláudia Maria Garcia M. de Oliveira - Ph.D.

Aprovada em 24 de fevereiro de 2005 pela seguinte Banca Examinadora:

Prof^a. Cláudia Maria Garcia M. de Oliveira - Ph.D do IME - Presidente

---

Prof^a. Cláudia Marcela Justel - D.Sc. do IME

Prof. Ruy Luiz Milidiú - Ph.D da PUC-RIO

Prof^a. Violeta de San Tiago D. B. Quental - D.Sc. da PUC-RIO

Rio de Janeiro
2005

A meu pai, Sr. Joaquim (in memorian).
A minha mãe, Damiana, e meus irmãos que sempre me apoiaram e me incentivaram.
A Meire, minha namorada, pelo apoio, carinho, amor e paciência que tem me dedicado.

AGRADECIMENTOS

À Prof[a] Claudia Oliveira pela orientação inteligente, incentivo e amizade.

À Prof[a] Claudia Justel pelos conselhos e ensinamentos transmitidos nas várias disciplinas em que fui seu aluno.

Aos professores Ruy Milidiú e Violeta Quental, ambos da PUC-Rio, por aceitarem o convite de participar da banca desse trabalho.

Ao Instituto Militar de Engenharia por disponibilizar um curso de qualidade e com bons professores.

À CAPES pela concessão da bolsa de estudos.

Ao meu amigo Roberto Jefferson que me incentivou a fazer a seleção de mestrado no IME e me deu grande apoio quando da minha chegada ao Rio de Janeiro.

Aos meus amigos do CLIC, em especial às companheiras Claudia Oliveira, Cândida, Maria Claudia e Milena, que me ajudaram bastante numa tentativa de correção do corpus de treino.

Aos amigos que fiz dentro e fora do IME e que de alguma forma fizeram com que esses dois anos fossem menos estressantes: Maria Ana, Ernesto, Wagner, Márcio (Tche 1), Evandro (Tche 2), Sarah, Carlos, Eduarda, Isolina, Lauro, dentre outros.

Aos meus companheiros(as) do Colégio Paraíso, em Juazeiro do Norte, por "aguentarem as pontas" quando não pude dar a devida atenção à manutenção do software Saber. Em especial agradeço à minha amiga Lucilene pelo constante incentivo.

"We can only see a short distance ahead, but we can see plenty there that needs to be done."

"Nós vemos apenas uma curta distância à frente, mas podemos ver nela muito que precisa ser feito."

Alan Turing

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SÍMBOLOS E ABREVIATURAS

SÍMBOLOS

C	conjunto de classificações possíveis
e	expressão condicional
r	uma regra
e_r	expressão condicional da regra r
$B(r)$	conjunto de itens do corpus para o qual a regra r se aplica de forma negativa r
$G(r)$	conjunto de itens do corpus para o qual a regra r se aplica de forma positiva r
$bad(r)$	|B(r)|, número de elementos em B(r)
$good(r)$	|G(r)|, número de elementos em G(r)
S	espaço de amostras (ítens do corpus)
s	um item do corpus
$C[s]$	classificação atual do item s
$T[s]$	classificação correta do item s

SIGLAS

AM	Aprendizado de Máquina
HMM	*Hidden Markov Models* (Modelos Ocultos de Markov)
IME	Instituto Militar de Engenharia
IA	Inteligência Artificial
MBL	*Memory-Based Learning* (Aprendizado Baseado em Memória)
PLN	Processamento de Linguagem Natural
PUC-Rio	Pontifícia Universidade Católica do Rio de Janeiro
RI	Recuperação de Informações
SN	Sintagma Nominal

SRIT	Sistema de Recuperação de Informações Textuais
TA	Termo Atômico
TBL	*Transformation-Based Learning* (Aprendizado Baseado em Transformações)
UFRJ	Universidade Federal do Rio de Janeiro
WWW	*World Wide Web*

RESUMO

Nos últimos anos, tem-se observado um retorno aos métodos empíricos de Processamento de Linguagem Natural, em que a aquisição do conhecimento pela máquina é majoritariamente realizada com base nos dados. Desde a última década, várias técnicas de Aprendizado de Máquina (AM) têm sido utilizadas para a identificação automática de sintagmas nominais (SNs). A identificação de SNs em textos tem aplicações em diversos problemas como: recuperação e extração de informações, análise sintática, resolução de co-referência, identificação de relações semânticas, entre outros. A maior parte dos trabalhos publicados na área têm o inglês como língua alvo; trabalhos sobre o uso de técnicas de AM para a identificação de SNs do português brasileiro não foram encontrados.

Nesse trabalho, a identificação automática de SNs do português brasileiro é tratada como uma tarefa de classificação a ser automaticamente aprendida com o uso da técnica de AM chamada Aprendizado Baseado em Transformações (do inglês *Transformation Based Learning* – TBL). Nesta abordagem, o aprendizado é guiado por um corpus de treino que contém exemplos corretamente classificados. A própria classificação dos exemplos foi derivada automaticamente a partir de um corpus preexistente. O conhecimento lingüístico gerado por essa técnica consiste de uma lista ordenada de regras de transformação, que pode ser utilizada para a classificação de novos textos.

Para a redução dos erros de classificação de preposições foi proposto um novo tipo de molde de regras, cuja a unidade básica, aqui chamada de Termo Atômico, possui uma janela de contexto de tamanho variável e um teste que precede a captura dos valores que compõem as regras.

Nesse trabalho também é mostrado que o uso de uma classificação inicial mais precisa para o português e o uso de um lematizador de verbos contribuem para a redução do tempo de treinamento, e, ainda, trazem alguns benefícios para a eficácia das regras aprendidas. Na identificação de SNs do português brasileiro com o uso da ferramenta TBL desenvolvida foi obtido, no melhor caso, precisão de 86,6% e abrangência de 85,9%.

ABSTRACT

In recent years, empirical methods of Natural Language Processing have been back in focus, methods in which the linguistic data is the major source of knowledge acquisition. During the last decade, Machine Learning (ML) has been applied to the identification of noun phrases (NPs). NP identification has a wide range of applications, such as information retrieval and extraction, syntactic parsing, co-reference resolution and identification of semantic relations. Most of the work that has been published in the field has taken English as the target language; reports on the application of ML techniques the identification of Brazilian Portuguese NPs have not been found.

In this work, the automatic identification of Brazilian Portuguese NPs is treated as a classification task that should be automatically learned using the ML technique denominated Transformation Based Learning – TBL. In this framework, the learning process is guided by a training corpus containing examples that have been correctly classified. The classification itself has been automatically derived from an existing corpus. The linguistic knowledge generated by this technique consists of an ordered list of transformation rules that can be used to classify new texts.

In order to reduce the number of classification errors related to prepositions, a new type of rule template has been proposed, in which the basic unit, the Atomic Term, has a variable size context window and a condition to be tested before the capture of the values that will make up the rules.

The baseline classification method is discussed and improved for the Portuguese case, and also, to cover language-specific verb inflexions, a stemmer is used in the reduction of training time and the improvement in learned rules. In the identification of Brazilian Portuguese NPs with the TBL tool, the best result obtained was 86,6% for precision and 85,9% for recall.

1 INTRODUÇÃO

1.1 CONTEXTO E MOTIVAÇÃO

Desde a invenção dos computadores, o homem vem buscando formas de simplificar cada vez mais a interação homem-máquina. Nesse sentido, esforços têm sido realizados na tarefa de processar a linguagem natural, com o intuito de que no futuro o homem possa se comunicar com o computador utilizando uma linguagem natural (e.g, português, inglês, espanhol, etc). Dessa maneira, elimina-se a necessidade de o usuário ter que se adequar a formas mais complicadas de interação, como o aprendizado de uma linguagem artificial, o que geralmente demanda bastante tempo e dedicação (como é o caso das linguagens de consulta a banco de dados).

Dentro desse contexto, o Processamento de Linguagem Natural (PLN) apresenta-se como um ramo da Inteligência Artificial (IA) que tem por objetivo a interpretação e a geração de linguagem natural (BARROS, 1997).

O alto nível de complexidade normalmente encontrado nas abordagens de PLN pode ser percebido pelo caráter multidisciplinar deste ramo de pesquisa, que envolve estudos nas áreas de Ciência da Computação, Lingüística e Ciências Cognitivas. Segundo BARROS (1997), a pesquisa em PLN divide-se em duas subáreas de trabalho: interpretação e geração.

A interpretação de linguagem natural baseia-se em métodos que procuram "compreender" enunciados em alguma Linguagem Natural, buscando traduzi-los para uma representação que possa ser utilizada pelo computador. No âmbito da geração de linguagem natural, o contrário ocorre, ou seja, o computador traduz uma representação interna para a sua expressão em uma língua natural.

A interpretação completa de um enunciado requer vários processos, dentre os quais: (1) a *análise fonética*, que se refere ao reconhecimento das sílabas ou palavras a partir dos caracteres – na escrita – ou dos sons – na fala – que formam o enunciado; (2) a *análise morfológica*, que identifica os elementos significativos básicos que formam as palavras; (3) a *análise sintática*, que identifica como as palavras se combinam para determinar a estrutura das frases; (4) a *análise semântica*, que identifica o significado das palavras e como elas se combinam para formar o significado das frases; (5) e a *análise do discurso*, que tem como uma de suas tarefas a identificação da influência de uma ou mais frases na

interpretação das frases subseqüentes.

Dentro desse panorama de PLN situa-se a identificação de Sintagmas Nominais (SNs). Tal tarefa tem aplicações em diversos problemas como: recuperação e extração de informações, análise sintática, resolução de co-referência, identificação de relações semânticas, etc. A identificação automática de SNs diz respeito ao uso de programas computacionais que, dada uma sentença, possam identificar quais seqüencias de palavras constituem SNs.

A construção de tecnologia de processamento de língua é freqüentemente prejudicada pela dificuldade de aquisição pela máquina de conhecimento lingüístico. Alguns sistemas de identificação automática de SNs, como (VOUTILAINEN, 1993; MIORELLI, 2001), utilizam gramáticas criadas manualmente que descrevem o comportamento sintático do SN e realizam a análise das sentenças. A produção artesanal desse conhecimento para que seja armazenado na máquina é muito custosa e, na maioria dos casos, não é compartilhável entre diferentes aplicações e diferentes línguas.

Em contrapartida, nos últimos anos, tem-se observado um retorno aos métodos empíricos de processamento de linguagem, em que a aquisição do conhecimento pela máquina é majoritariamente realizada com base nos dados. O volume de dados disponível eletronicamente e o aumento vertiginoso do poder computacional das máquinas vêm possibilitando esse direcionamento. Desde a ultima década, várias técnicas de Aprendizado de Máquina (AM) têm sido utilizadas para a identificação de SNs. O aprendizado a partir de grandes volumes de textos tem sido utilizado para possibilitar a automatização do processamento de língua em larga escala.

Normalmente, para a utilização de uma técnica de AM, é necessária a existência de um corpus[1] de treino contendo exemplos corretamente identificados do problema que se deseja resolver. De acordo com os experimentos realizados por NGAI (2000), com relação à identificação de Sintagmas Nominais de textos em inglês, é mais vantajoso utilizar recursos humanos para fazer anotação[2] de corpus e utilizar uma técnica de AM para "aprender" a identificar SNs a partir desse corpus, do que utilizar recursos humanos para criar manualmente regras de transformações para uma gramática de identificação. Dentre algumas vantagens listadas em NGAI (2000), podemos citar:

- **Aquisição Distribuída de Conhecimento:** com a utilização de AM fica mais fá-

[1]O termo corpus (corpora no plural) é utilizado para designar um repositório de dados lingüísticos. Na maioria das vezes é um conjunto de textos contendo informações adicionais de interesse para um determinado problema lingüístico.

[2]Estamos designando de *anotação de corpus* o ato de acrescentar, aos *tokens* (palavras e sinais de pontuação) do corpus, etiquetas que identificam alguma informação linguística de interesse.

cil de se combinar esforços de múltiplas pessoas. Corpora de treino criados por pessoas diferentes podem ser combinados facilmente para formarem um corpus maior. Em contraste, é muito difícil, ou quase impraticável, a combinação de listas de regras criadas manualmente por pessoas diferentes.

- **Robustez:** baseado em observações empíricas, a performance de sistemas que utilizam regras codificadas manualmente tende a apresentar uma maior variação, enquanto que os resultados de sistemas treinados com corpora anotados são mais uniformes.

- **Independente dos Mecanismos de Inferência:** uma vez construído um corpus de treino, o aprendizado pode ser realizado por diversas técnicas, sendo que a performance dos resultados vai depender principalmente da técnica usada. Subseqüentes melhoras nos algoritmos de treinamento podem trazer melhorias nos resultados, sem a necessidade de alterações no corpus. Ao contrário, a performance obtida por um conjunto de regras codificado manualmente é definitiva, a não ser que haja um revisão humana das regras.

Nesse trabalho, o problema da identificação de SNs em textos em português é abordado com o uso do método conhecido como Aprendizado Baseado em Transformações (do inglês *Transformation Based Learning* – TBL).

1.2 OBJETIVOS DA DISSERTAÇÃO

O principal objetivo desse trabalho é o desenvolvimento de uma ferramenta para a identificação automática de Sintagmas Nominais (SNs) de textos em português brasileiro, com o uso de uma técnica de Aprendizado de Máquina (AM). De um ponto de vista mais abrangente, a pesquisa também obteve outros resultados que são importantes para a área de Aprendizado de Máquina e para a pesquisa em identificação de SNs do português.

A ferramenta para identificação de SNs aqui proposta possui características e objetivos diferentes dos objetivos da principal ferramenta atualmente disponível que identifica SNs de textos em português, o analisador sintático PALAVRAS (BICK, 2000). A tarefa específica da ferramenta apresentada nesse trabalho é a identificação de SNs, buscando executar essa tarefa de forma rápida, e, assim, permitindo o seu uso como uma "fase inicial" de outras tarefas de PLN. O PALAVRAS tem como propósito fazer a análise sintática completa de uma sentença, que é um processamento bem mais custoso e que gera informações muitas vezes desnecessárias para certas tarefas, como é o caso da extração

de termos de indexação para sistemas de recuperação de informações. Outra característica de distinção é que nesse trabalho, o conhecimento utilizado pela ferramenta para identificar SNs é gerado com o uso de uma técnica de AM, enquanto que o PALAVRAS usa conhecimento lingüístico gerado manualmente.

Nesse trabalho também é proposto um novo tipo de molde de regra para o TBL, cuja unidade básica, aqui chamada de Termo Atômico (TA), possui uma janela de contexto de tamanho variável e um teste que precede a captura dos valores que compõem as regras.

Outro desafio proposto nesse trabalho é o desenvolvimento de um corpus de treino contendo textos em português brasileiro com os SNs identificados.

1.3 ORGANIZAÇÃO DA DISSERTAÇÃO

Após essa introdução, o capítulo 2 apresenta os conceitos básicos sobre o SN do português, dando ênfase à sua estrutura sintática. O capítulo 2 também discute a importância da identificação automática de SNs e algumas das suas aplicações. Nesse capítulo também é especificada a codificação utilizada para identificar os SNs nas sentenças e o tipo de SN que será utilizado.

O capítulo 3 apresenta alguns conceitos sobre Aprendizado de Máquina, suas aplicações em processamento de linguagem natural e mostra ainda uma breve descrição das técnicas de AM mais utilizadas para a identificação de SNs em textos em inglês.

O capítulo 4 descreve a técnica de AM conhecida por Aprendizado Baseado em Transformações (Transformation-Based Learning - TBL), mostrando ainda as suas vantagens e desvantagens em relação a outras técnicas de AM. Esse capítulo também descreve o algoritmo FastTBL.

No capítulo 5 é apresentada uma proposta de um novo Termo Atômico (TA) que aumenta o poder de especificação dos moldes de regras do TBL. Esse capítulo também descreve a ferramenta TBL desenvolvida como uma das tarefas dessa pesquisa.

O capítulo 6 mostra um estudo de caso da aplicação da ferramenta TBL proposta para a identificação de SNs em textos em português brasileiro. São mostradas as decisões a obtenção dos dados para treino e teste, as configurações usadas para os experimentos e os resultados comparativos.

O capítulo 7 apresenta as conclusões do trabalho e traz algumas sugestões para trabalhos futuros.

No APÊNDICE 1 é mostrado o conjunto de moldes de regras que obteve os melhores resultados na identificação de SNs do português. Nesse conjunto existem seis moldes

que contêm TAs com restrição. No APÊNDICE 2 são listadas as 96 primeiras regras aprendidas com o uso dos moldes mostrados no APÊNDICE 1 juntamente com um corpus de treino de 200 mil itens. No APÊNDICE 3 é mostrado um fragmento do corpus de teste com os SNs identificados pela aplicação das regras aprendidas. O APÊNDICE 4 mostra os moldes de regras que foram adicionados ao conjunto de moldes de RAMSHAW (1995) para formar um conjunto estendido, que foi utilizado nos experimentos da seção 6.5. Os APÊNDICES 5 e 6 contêm informações a cerca do treinamento e aplicação de regras aprendidas, respectivamente, com o uso da ferramenta TBL desenvolvida.

O ANEXO 1 mostra o conjunto de etiquetas morfossintáticas utilizadas nos corpora de treino e teste. No ANEXO 2 são listados os 100 moldes de regras propostos por RAMSHAW (1995).

2 IDENTIFICAÇÃO DE SINTAGMAS NOMINAIS

Nesse capítulo são apresentadas as definições e a estrutura do Sintagma Nominal (SN) do Português. É mostrada também a importância e algumas das aplicações da identificação automática de SNs. O tipo de SN que será utilizado nesse trabalho também é apresentado, bem como a codificação utilizada para identificar os SNs.

2.1 CONCEITOS E ESTRUTURA DO SN DO PORTUGUÊS

Essa seção traz uma revisão sobre o Sintagma Nominal do Português usando uma abordagem sintática, tendo como principal referência a gramática de Mário A. Perini (PERINI, 2003) e a descrição sobre SN encontrada em PINILLA (2004). Nos exemplos mostrados, os SNs terão seus limites identificados por colchetes, quando necessário.

2.1.1 A CLASSE DOS SINTAGMAS E O SN

As seqüências gramaticais de uma língua são chamadas de *sentenças*. A combinação das palavras para formarem as sentenças não é aleatória; é necessário obedecer a determinados princípios da língua. A *sintaxe* é a parte da gramática que estuda como esses elementos se combinam para formarem as sentenças. O estudo da estrutura (forma) da sentença recebe o nome tradicional de *análise sintática*.

Segundo PERINI (2003), as sentenças são formadas por *constituintes*, muitas vezes uns dentro dos outros. Por constituinte, Perini entende "certos grupos de unidades que fazem parte de seqüências maiores, mas que mostram certo grau de coesão entre eles". Por exemplo, na frase:

(2.1) A casa de Lulu é azul e branca.

Os falantes "sentem" que "a casa de Lulu" forma uma unidade, o que não é verificado em "Lulu é azul" (PERINI, 2003). Dizemos então, para a frase (2.1), que "a casa de Lulu" é um constituinte, e que "Lulu é azul" não é um constituinte.

De acordo com a idéia de que os constituintes muitas vezes contêm outros constituintes, pode-se analisar a sentença (2.1) como contendo, entre outros, os seguintes constituintes: "a casa de Lulu é azul e branca", "a casa de Lulu", "casa de Lulu", "azul e branca", "é azul e branca", etc.

Segundo Perini, é importante ter uma boa noção da estruturação das sentenças em constituintes, porque toda a análise se baseia nela. Por exemplo, os constituintes costumam receber uma "função" na análise tradicional: "a casa de Lulu" é o *sujeito*, e "azul e branca" é o *predicativo do sujeito*. Já a seqüência "Lulu é azul" não recebe nenhuma função, visto que não é um constituinte.

Perini diz que é preciso levar em conta, ao se fazer a análise de uma sentença, o fato de que as sentenças se estruturam de maneira hierárquica, isto é, contêm constituintes que, por sua vez, contêm outros constituintes. Por exemplo, seja a seguinte sentença:

(2.2) Meus vizinhos arranjaram um cachorro horrivelmente barulhento

Pode-se fazer um "primeiro corte", definindo os *grandes constituintes* (ou *sintagmas*) da oração, da seguinte forma (PERINI, 2003):

(2.3) [Meus vizinhos] – [arranjaram] – [um cachorro horrivelmente barulhento]

Podemos dizer que esses constituintes são os sintagmas da sentença, ou ainda, como diz Perini, são os "constituintes imediatos da oração". Cada um desses sintagmas desempenha uma função especial na sentença. Na sentença (2.3) essas funções se denominam, na ordem: "sujeito", "predicado" e "objeto direto".

De acordo com KOCH (1985), o *sintagma* consiste num conjunto de elementos que constituem uma unidade significativa dentro da sentença e que mantêm entre si relações de dependência e de ordem. Organizam-se em torno de um elemento fundamental, denominado núcleo, que pode, por si só, constituir o sintagma. Segundo LOBATO (1986) os sintagmas são formados por constituintes, que são um conjunto de palavras de uma sentença que a divide, normalmente, em duas subpartes básicas: sintagma nominal (SN) e sintagma verbal (SV). Quando o núcleo for um verbo, tem-se um SV e quando o núcleo for um nome têm-se um SN. Funcionando como modificador de um SN ou de um SV, temos o sintagma preposicionado (SP), que combina preposições e substantivos.

Segundo Perini, o SN pode ser definido de maneira muito simples: "é o sintagma que pode ser sujeito de alguma oração". Por exemplo, na seguinte oração:

(2.4) [Esse professor] é [um neurótico]

temos que "esse professor" é um SN porque é sujeito da oração (2.4); e "um neurótico" é também um SN, porque, mesmo que não seja sujeito em (2.4) pode ser sujeito em outra oração, como em (2.5).

(2.5) [Um neurótico] rabiscou [meus livros]

2.1.2 CONSTITUINTES DO SN

As palavras que constituem as classes abertas da língua portuguesa, mas que se opõe a verbo, são chamados de *nomes*. As classes abertas possuem mecanismos de produção de novos itens, sendo que a derivação e composição são os principais mecanismos. Esses processos fazem com que, mesmo contabilizando todas as palavras da língua já proferidas em um instante, ainda assim, o conjunto dos itens potenciais não têm limite. Fazem parte dessa classe os substantivos, adjetivos e verbos. As classes fechadas são aquelas cuja forma se consagrou na língua e onde as mudanças são quase inexistentes. São elas: artigo, pronome, numeral, advérbio, preposição, conjunção e interjeição.

Tipicamente, o núcleo do sintagma nominal é um nome, próprio ou comum; mas pode ser também um pronome substantivo (pessoal, demonstrativo, indefinido, interrogativo, possessivo ou relativo). O SN também pode conter, além do núcleo, determinantes e/ou modificadores. Determinantes são os elementos que especificam outro numa expressão lingüística – no caso dos SNs eles antecedem e especificam o núcleo. No SN os elementos determinantes são representados pelos artigos, pronomes e numerais (PINILLA, 2004). Modificadores são os elementos que estabelecem relação de modificação dentro de um sintagma. No SN o núcleo geralmente é modificado por adjetivos, por sintagmas preposicionados (SPs) ou por orações adjetivas; em certos casos, o modificador pode ser um advérbio ou locução adverbial. O modificador, no caso do português, geralmente vem posposto ao núcleo. Por exemplo, vejamos novamente a sentença (2.2):

(2.2) [Meus vizinhos] arranjaram [um cachorro horrivelmente barulhento]

Nessa sentença "meus" e "um" são determinantes, que especificam os núcleos "vizinhos" e "cachorro", respectivamente. E "horrivelmente barulhento" é um modificador, que está modificando o núcleo "cachorro".

Os nomes são freqüentemente usados para dois tipos de função: identificar seres e objetos (substantivos) e caracterizar, especificar e especializar seres e objetos (adjetivos). Segundo MIORELLI (2001) o substantivo é, por natureza, o núcleo do SN e o adjetivo, do ponto de vista funcional, tem a posição de modificador do núcleo do SN.

O SN apresenta diversas possibilidade de estruturação, ou seja, o SN não é fixo. Ele pode ser formado ora por um único nome (N), ora por determinante (DET) + núcleo (N) + modificador (MOD); ora por N + MOD; ora por DET+ MOD + N + MOD. Enfim, não há grande rigidez na sua formação, conforme pode ser observado nos exemplos de SNs da TAB. 2.1 que foram obtidos de PINILLA (2004), e cujos núcleos encontram-se em negrito.

TAB 2.1: Exemplo de SNs com diferentes estruturações

os **aguaceiros** de verão as **folhas** com lustro o **ar** limpíssimo muitas **novidades** em cada canto	DET + N + MOD
chuva grossa	N + MOD
uma certa **crença**	DET + DET + N
a **terra** e a **areia** assentadas	DET + N + DET + N + MOD
grande **movimentação** de bichos	MOD + N + MOD
uma certa **alegria** despropositada	DET + DET + N + MOD

No caso do SN "a **terra** e a **areia** assentadas" não existe um SN com dois núcleos, mas sim, dois SNs ("a terra"; "a areia") coordenados pela conjunção "e", formando por sua vez um terceiro SN juntamente com o modificador "assentadas".

De acordo com PINILLA (2004), "os pronomes pessoais têm, como uma de suas características básicas, a possibilidade de constituírem, sozinhos, um SN". Por exemplo, na sentença (2.6), o pronome pessoal "eu" constitui, sozinho, um SN.

(2.6) [Eu] fiquei cansado

Também é possível que outros pronomes possam funcionar como núcleos de SN ou constituírem, sozinhos, um SN. Por exemplo:

(2.7) [Isto] é meu

(2.8) [Tudo] foi resolvido

Segundo PINILLA (2004), "qualquer vocábulo de outra classe, que ocupe a função de núcleo do sintagma nominal, será entendido como um substantivo". As sentenças (2.9) e (2.10) exemplificam esse caso, onde o vocábulo de outra classe aparece em negrito.

(2.9) "[**Viver**] é muito perigoso..." (Guimarães Rosa)

(2.10) Recebi [um sonoro **não**] como [resposta ao meu pedido].

Os SNs, como dito anteriormente, podem ser modificados por adjetivos, sintagmas preposicionados e por orações subordinadas adjetivas. Esse processo de inclusão de modificadores é chamado de ampliação do SN (PINILLA, 2004).

Os SPs são formados pela combinação de uma preposição e um SN, e são utilizados como modificadores de SNs e SVs. A preposição que inicia o SP estabelece uma relação

entre o SN (ou SV) que a precede e o SN que a sucede. Essa ligação entre termos é um processo de subordinação denominado regência, onde o primeiro elemento – chamado antecedente – é o termo que rege, que impõe um regime e o segundo elemento, por sua vez – chamado conseqüente – é o termo regido, aquele que cumpre o regime estabelecido pelo antecedente. A sentença (2.11) mostra exemplos de SNs contendo SPs.

(2.11) [O técnico **da seleção brasileira**] não ouve [os apelos **da torcida**].

Muitos adjetivos podem ser substituídos por SPs mantendo o mesmo significado, como mostrado na TAB. 2.2.

TAB 2.2: Exemplos de sintagmas preposicionados

substantivo	adjetivo	locução adjetiva
proteína	animal	dos animais
bichos	interessantíssimos	de muito interesse
esqueleto	cartilaginoso	de cartilagem
pesquisadores	preocupados	com preocupação

As orações subordinadas adjetivas são formadas pela união de duas outras sentenças usando-se o pronome relativo *que*. Por exemplo, a união das duas sentenças "*o rapaz é meu amigo*" + "*o rapaz subiu no ônibus*" formam a oração subordinada adjetiva "*que subiu no ônibus*" na sentença (2.12).

(2.12) [o rapaz **que subiu no ônibus**] é [meu amigo].

PERINI (2003) denomina de *sintagma complexo* o sintagma que contém oração subordinada. Nesse trabalho será tratada somente a identificação de *SNs simples*, ou seja, de SNs que não possuem orações subordinadas adjetivas. Dessa forma, na identificação dos SNs da sentença (2.12), teríamos três SNs num mesmo nível hierárquico(ver 2.13), ao invés de apenas dois, como normalmente seria analisado.

(2.13) [o rapaz] que subiu em [o ônibus] é [meu amigo].

2.2 IDENTIFICAÇÃO/EXTRAÇÃO AUTOMÁTICA DE SNS E SUAS APLICAÇÕES

Como citado antes, a identificação automática de SNs diz respeito ao uso de programas computacionais que, dada uma sentença, possam identificar quais seqüências de palavras constituem SNs.

A identificação automática de SNs tem várias utilidades no campo de PLN, bem como em outras áreas como Recuperação de Informações (RI) e Extração de Informações (EI). Em PLN alguns dos usos seriam, por exemplo, para a resolução de co-referência; para a identificação de relações semânticas como hiperonímia/hiponímia; e ainda para funcionar como uma etapa inicial num processo de análise sintática. A principal aplicação da identificação (e extração) de SNs em RI é a criação de termos de indexação.

Nas subseções seguintes serão apresentadas, de forma um pouco mais detalhada, algumas das aplicações da identificação automática de SNs na RI, na resolução de co-referência e na identificação de relações de hiponímia-hiperonímia.

2.2.1 RECUPERAÇÃO DE INFORMAÇÕES

Os pesquisadores da área de RI vêm mostrando crescente interesse na identificação automática de SNs. A RI lida com a representação, o armazenamento, a organização e o acesso a itens de informações. O propósito é disponibilizar para o usuário, através dessa representação e organização dos itens, o acesso à informação desejada, de forma fácil e precisa (BAEZA-YATES, 1999).

Com o surgimento da Internet, mais especificamente da Word Wide Web (WWW), houve um enorme crescimento da produção e consumo de textos devido ao acesso livre e custo muito baixo. E são os Sistemas de Recuperação de Informações Textuais (SRIT) os principais fornecedores dos meios de recuperação e manipulação dessa grande massa de documentos. Por isso a pesquisa em RI tem se intensificado muito nas duas últimas décadas, com o intuito de desenvolver métodos cada vez mais eficazes de indexação e busca. Muitas dessas novas abordagens são baseadas em PLN.

Os SRITs geralmente funcionam a partir da construção de índices de termos presentes nos documentos. Tais termos devem representar bem o conteúdo do texto, visto que o SRIT realiza as buscas dos usuários tomando como base esse índice. Segundo OLIVEIRA (2003) as expressões de maior poder discriminatório são, em geral, aquelas de sentido substantivo que podem realizar funções temáticas, como sujeito e objeto, e certas funções semânticas, como agente e instrumento, ou certas funções retóricas, como tópico ou tema. As expressões desse tipo são em grande parte Sintagmas Nominais (SNs), por isso é grande o interesse na extração automática de SNs para a construção desses índices de termos.

Para o inglês, existem diversos trabalhos sobre identificação ou extração automática de SNs com o propósito de criar termos de indexação. Dentre esses trabalhos podemos destacar o de VOUTILAINEN (1993), que descreve uma ferramenta de extração de SNs

chamada de *Nptool*. Tal ferramenta utiliza um conjunto de tarefas comuns a um sistema clássico de PLN, ou seja, possui um léxico, um processamento morfológico e um processo de análise sintática (possui regras sintáticas para toda a gramática da língua inglesa). O principal objetivo dessa ferramenta é auxiliar a tarefa de reconhecer palavras representativas de um texto, para a função de índice. Podemos destacar ainda os trabalho de ZHAI (1997) e de EVANS (1996). Neste último, estatísticas de corpus e heurísticas lingüísticas são utilizadas para a extração de SNs.

Para o português existe o trabalho de MIORELLI (2001) que propõe um método, o ED-CER, para a extração de SNs em sentenças em português. Esse método constitui-se de duas fases: na primeira fase são selecionados candidatos a SNs, e, na segunda fase, um analisador sintático verifica quais candidatos são realmente SNs, tomando como base uma gramática do SN construída a partir da abordagem de PERINI (2003). Existe ainda o trabalho de KURAMOTO (1995), que descreve um SRIT que utiliza SNs como indexadores, mas a extração dos SNs, nesse sistema, é realizada de forma manual.

2.2.2 RESOLUÇÃO DE CO-REFERÊNCIA

Segundo SOON (2001), *resolução de co-referência* é o processo de determinar quando duas expressões, em linguagem natural, referem-se a uma mesma entidade do mundo. Num sistema que realiza processamento de co-referência, o objetivo é identificar as seqüências de expressões em um texto que se referem a uma mesma entidade. De acordo com VIEIRA (2000), a resolução de co-referência é importante para diversas aplicações de PLN como, por exemplo, geração automática de resumos, tradução automática e extração de informações, e ainda, na recuperação de informações. O seguinte trecho, retirado de VIEIRA (2000), mostra um exemplo de expressões co-referentes (em negrito):

> (...) ligou **a subestação de energia elétrica do Complexo Automotivo da General Motors**. "A avançada tecnologia utilizada na construção d**a subestação** otimizou equipamentos, possibilitando a expansão das atividades e menos perdas de energia".

A identificação de SNs é um fator importante para a resolução de co-referência, visto que as entidade referenciadas em um texto são SNs. Dentre alguns trabalhos que usam identificação de SNs na resolução de co-referência para o inglês, podemos citar (SOON, 2001; BEAN, 1999; CARDIE, 1999). Para o português existe o trabalho de VIEIRA (2000), que apresenta a construção de um corpus composto por uma lista de sintagmas nominais extraídos semi-automaticamente a partir de artigos do jornal Correio do Povo.

Tal corpus serve de base para um sistema de anotação automática de co-referência textual para a língua portuguesa.

2.2.3 IDENTIFICAÇÃO DE RELAÇÕES DE HIPERONÍMIA/HIPONÍMIA

De acordo com PINILLA (2004), na relação de hiperonímia temos o caso em que a primeira expressão mantém com a segunda uma relação de todo-parte ou classe-elemento. Na relação de hiponímia existe o caso inverso: a primeira expressão mantém com a segunda uma relação de parte-todo ou elemento-classe. Dessa forma, um *hiperônimo* é um vocábulo mais geral e de sentido mais abrangente que outro, enquanto um *hipônimo* é um vocábulo mais específico e mais limitado que outro. Por exemplo, "animal" é um hiperônimo de "rato" e "rato" é um hipônimo de "animal".

Para construir um léxico computacional estruturado, FREITAS (2004) propõe um método para a extração automática de relações semânticas em textos do português. Dentre as relações de interesse está a de hiperonímia/hiponímia. O método proposto para a extração desse tipo de relação envolve, numa primeira etapa, a identificação de SNs, e numa segunda etapa, a identificação de expressões do tipo:

(a) "*tipos de* SN1: SN (,SN...,SN) e/ou (SN)"

(b) "SN1 *como* SN (,SN...,SN) e/ou (SN)"

onde a relação semântica é interpretada como: os SNs que aparecem após ":" ou após "como" são hipônimos do SN1. Por exemplo:

(2.14) Nas restingas coexistem vários **tipos de [vegetação]: [árvores altas], [campos] e [plantas]** situadas em locais alagados

(2.15) **[Barreiras geográficas] como [montanhas], [desertos] e [rios caudalosos]** impedem o deslocamento dos animais para regiões distantes

Em (2.14) os SNs "árvores altas", "campos" e "plantas" são hipônimos de "vegetação", e, em (2.15), os SNs "montanhas", "desertos" e "rios caudalosos" são hipônimos de "barreiras geográficas".

2.3 SNS DO PORTUGUÊS VS. SNS BÁSICOS DO INGLÊS

As línguas portuguesa e inglesa não são totalmente dissimilares no nível sintático. No geral, a ordem das palavras, com relação às classes gramaticais, é similar. Uma gramática

de estrutura frasal básica, como a definida na FIG. 2.1[3], abrange as duas línguas da regra 1 até a regra 3. A regra 4, que define a estrutura do SN, mostra que a posição do adjetivo é preferencialmente pré-nominal no inglês e pós-nominal no português.

Regras Sintáticas Inglês	Regras Lexicais Inglês	Regras Sintáticas Português	Regras Lexicais Português
1. S → SN SV	Det → *the*	1. S → SN SV	Det → *o*
2. SV → V SN	V → *repaired*	2. SV → V SN	V → *consertou*
3. SN → Det N	N → *engineer*	3. SN → Det N	N → *engenheiro*
4. SN → Det Adj N	Adj → *old*	4. SN → Det N Adj	Adj → *velho*

FIG 2.1: Gramática de estrutura frasal básica

Outra característica distinta do SN do inglês é que um substantivo pode ocupar a posição de um predicado adjetivo, como em "*door mat*", mas em português o mesmo tipo de SN precisa de uma preposição, como em "*tapete da porta*". A sequência *PREP (Det) N* é considerada como uma locução adjetiva por alguns gramáticos.

Na utilização de aprendizado de máquina para a identificação de SNs do inglês tem-se usado a noção de *SN básico*, que é definido por RAMSHAW (1995) como um SN não recursivo que inclui determinantes e pré-modificadores, mas não inclui pós-modificadores como sintagmas preposicionados e orações subordinadas. Em Português essa noção provê um conjunto de SNs muito pobre. Vejamos o seguinte exemplo de SN básico em inglês e a sua tradução para o português.

$_{SN}$[the first Government drug manufacturing plant]$_{SN}$
$_{SN}$[a primeira fábrica de $_{SN}$[produção de $_{SN}$[remédios do $_{SN}$[governo]]]]

Observa-se que a tradução tem necessariamente quatro SNs básicos aninhados, sendo que essa é uma construção bastante comum em português. Dessa forma, para se obter um SN mais "informativo" na lingua portuguesa, é necessário considerar SNs recursivos. Por isso, resolvemos identificar SNs que, em oposição aos SNs básicos, contenham pós-modificadores como adjetivos e sintagmas preposicionados, mas não incluindo pós-modificadores que contenham orações subordinadas. Logo, nesse modelo, o exemplo anterior em português representaria um único SN, como mostrado a seguir:

$_{SN}$[a primeira fábrica de produção de remédios do governo]$_{SN}$

[3]S= Sentença; SN= Sintagma Nominal; SV= Sintagma Verbal; V= Verbo; N= Nome; Det= Determinante; Adj= Adjetivo

Como resultado, o problema de identificar SNs do português torna-se mais complexo do que o da identificação de SNs básicos, visto que inclui o problema de ligação do sintagma preposicionado.

2.4 CODIFICAÇÃO UTILIZADA PARA IDENTIFICAR OS SNS NAS SENTENÇAS

A identificação de SNs é tratada como um problema de classificação, onde o objetivo é associar a cada item do corpus uma etiqueta adicional que o classifique como pertencente ou não a um SN. Nesse trabalho, é usado o conjunto de etiquetas {I,O,B} proposto por RAMSHAW (1995), onde as palavras etiquetadas com I (*In*) pertencem a um SN, as marcadas com O (*Out*) estão fora de um SN, e a etiqueta B (*Begin*) é utilizada para marcar a palavra mais à esquerda de um SN que se inicia logo após um outro SN. Chamaremos essas etiquetas de *etiquetas SN*.

Nesse trabalho considera-se que o texto que terá seus SNs identificados já esteja etiquetado morfossintaticamente, ou seja, cada palavra já deve possuir uma etiqueta que identifica a sua classe de palavras.

A seguinte sentença, que tem os SNs identificados entre colchetes:

> [O terrorismo] espalhou [medo] em [o mundo inteiro].

seria codificada como:

> O/ART/I terrorismo/N/I espalhou/V/O medo/N/I em/PREP/O o/ART/I mundo/N/I inteiro/ADJ/O ./O

3 APRENDIZADO DE MÁQUINA

Esse capítulo traz uma pequena revisão sobre Aprendizado de Máquina, dando ênfase nas suas aplicações em PLN.

3.1 CONCEITOS

Segundo MITCHELL (1997), a pesquisa em *Aprendizado de Máquina* (AM) lida com a questão de como construir programas de computadores que possam "aprender" com a experiência, ou seja, cujo desempenho em determinada tarefa melhora com a experiência. AM é uma subárea de pesquisa de muita importância na Inteligência Artificial (IA), e engloba os estudos de métodos computacionais para a automação da aquisição do conhecimento e para a estruturação e acesso do conhecimento já existente. MITCHELL (1997) afirma ainda que um entendimento detalhado dos algoritmos de AM pode levar também a um melhor entendimento da capacidade (e incapacidade) do aprendizado humano.

AM é multidisciplinar e trabalha com idéias de diversas áreas, incluindo inteligência artificial, probabilidade e estatística, complexidade computacional, teoria da informação, psicologia, neurociências e filosofia.

De acordo com MITCHELL (1997), algoritmos de AM têm provado serem de grande valor prático para uma variedade de domínios de aplicações. Eles são especialmente úteis em: (a) problemas de Mineração de Dados (Data Mining), onde grandes banco de dados são analisados automaticamente, na busca de regularidades implícitas que possam ser úteis; (b) em domínios ainda pouco entendidos onde os humanos poderiam não ter o conhecimento necessário para desenvolver algoritmos efetivos (por exemplo, no reconhecimento facial a partir de imagens); (c) em domínios onde o programa necessita adaptar-se dinamicamente a mudanças (por exemplo, um sistema que se adapta às preferências de leitura de um indivíduo); e (d) em domínios em que o custo da aquisição ou codificação manual do conhecimento é muito custosa, como é o caso de PLN.

3.2 APRENDIZADO DE MÁQUINA E PROCESSAMENTO DE LINGUAGEM NATURAL

Nos últimos 10 anos houve uma série de mudanças na área de Processamento de Linguagem Natural (PLN) em relação à construção de gramáticas e bases de conhecimentos (lexicais, semânticos, etc). Tais recursos, que antes eram construídos manualmente, passaram a ser parcialmente ou completamente adquiridos com o uso de métodos de aprendizado estatísticos e simbólicos, treinados com grandes corpora de linguagem natural anotados ou não. Tais técnicas também são conhecidas, em PLN, como *métodos empíricos* ou *baseados em corpus*.

Segundo MÀRQUEZ (2000), dentre as principais razões da atual popularidade dos métodos baseados em corpus é possível destacar: (a) o crescente surgimento de grandes corpora digitais de diferentes níveis de anotações, linguagens, etc; (b) a constante melhora de performance dos hardwares e softwares; (c) o sucesso inicial obtido pelos processos estatísticos em problema básicos de PLN; e (d) o aparecimento e desenvolvimento de um grande número de aplicações de PLN baseadas em texto e com requerimentos específicos, para os quais os métodos convencionais baseados em conhecimento lingüístico parecem não ser muito apropriados.

A maioria das tarefas de PLN em que se tem utilizado AM são as que podem ser tratadas como problemas de classificação. Na maioria das aplicações, o objetivo é associar a cada palavra de um texto uma etiqueta que indica a sua classificação. O aprendizado geralmente é induzido por um corpus de treino que contém exemplos corretamente classificados. O conhecimento gerado pelo aprendizado quase sempre é representado por um conjunto de probabilidades contextuais ou por um conjunto de regras de transformações, que podem ser utilizados para a classificação de novos textos.

Dentre algumas tarefas de PLN que já foram abordadas com o uso de técnicas de AM podemos citar:

- etiquetagem morfossintática – *POS tagging* (BRILL, 1995; RATNAPARKHI, 1998): consiste da atribuição, aos itens lexicais de um texto, de etiquetas que identificam sua classe de palavras, gênero, número e outras categorias gramaticais;

- identificação de sintagmas nominais – *noun phrase chunking* (RAMSHAW, 1995; CARDIE, 1998; SANG, 2000a): como descrito neste trabalho;

- análise sintática parcial – *shallow parsing* (KOELING, 2000; MEGYESI, 2002; MOLINA, 2002; OSBORNE, 2000; RAMSHAW, 1995; SANG, 2000b, 2002): que con-

siste em dividir as sentenças de um texto em constituintes não recursivos chamados de *chunks*. Nessa divisão, cada palavra só deve pertencer a um único constituinte e as palavras sintaticamente relacionadas devem pertencer ao mesmo *chunk*;

- desambigüação do significado de palavras – *word sense disambiguation* (FLORIAN, 2001): essa tarefa consiste na associação do significado apropriado para uma dada palavra, em um contexto onde este significado é distinguível dentre outros potencialmente possíveis para aquela palavra;

- identificação dos limites de sentenças – *sentence boundary disambiguation* (STAMATATOS, 1999): essa tarefa consiste em segmentar um texto em sentenças a partir da identificação dos limites dessas sentenças;

- reconhecimento de entidades mencionadas – *named entity recognition* (UCHIMOTO, 2000; ZHOU, 2002): entidades mencionadas são sintagmas que contêm os nomes de pessoas, organizações, locais e etc. Dado um texto, o reconhecimento de entidades mencionadas consiste em identificar esses sintagmas no texto.

As técnicas de AM que têm sido mais utilizadas para a identificação de SNs do inglês são: Aprendizado Baseado em Transformações (*Transformation Based Learning – TBL*), Modelos Ocultos de Markov (*Hidden Markov Models*), Modelos de Entropia Máxima (*Maximum Entropy Models*) e Aprendizado Baseado em Memória (*Memory Based Learning*).

Para efetuar a identificação de SNs do Português Brasileiro escolhemos a técnica TBL. Essa escolha foi motivada principalmente pelos seguintes fatores: esse método tem conseguido bom desempenho para a identificação de SNs do inglês; o método vem sendo explorado por muitos pesquisadores da área, o que propicia o intercâmbio de resultados; e ainda, as vantagens que o TBL possui em relação aos métodos estatísticos, e que são mostradas no capítulo 4.

Para o entendimento dos métodos de aprendizado de máquina é fundamental a compreensão do conceito de *feature*, aqui traduzido como *traço*. Durante todo o trabalho, o termo *traço* denotará uma unidade observável de um item do corpus. Um item do corpus pode ser constituído por um ou mais traços e o tipo de tarefa é que define como esse item é representado e quais traços ele possui. Por exemplo, no caso da etiquetagem morfossintática (a tarefa de associar a cada palavra uma etiqueta que corresponde à sua

classe de palavras) cada item é composto por dois traços: a unidade léxica[4] (traço *word*) e a sua etiqueta morfossintática (traço *tpos*), como exemplificado a seguir:

O/ART rato/N comeu/V o/ART queijo/N.

Na tarefa de identificação de SNs, os itens são compostos por três traços: a unidade léxica, a sua etiqueta morfossintática e a etiqueta SN (traço *tsn*) que identifica se a palavra pertence ou não a um sintagma nominal:

O/ART/I rato/N/I comeu/V/O o/ART/I queijo/N/I.

A eficácia dos métodos de AM para a identificação de SNs geralmente é verificada com a utilização de um corpus de testes, cujos SNs são conhecidos. Tal eficácia é medida em termos de precisão – percentual de SNs identificados que estavam corretos – e abrangência – percentual de SNs existentes no corpus de teste e que foram identificados corretamente.

O método TBL será apresentado detalhadamente nos capítulos 4 e 5. As seguintes subseções apresentam, de forma breve, as outras técnicas de AM citadas anteriormente.

3.2.1 MODELOS DE ENTROPIA MÁXIMA

De acordo com RATNAPARKHI (1997), muitos problemas de PLN podem ser reformulados como problemas de classificação estatística, onde a tarefa consiste em estimar a probabilidade de uma "classe" a ocorrer com um "contexto" b, ou $p(a, b)$. Em PLN esses contextos geralmente incluem palavras, etiquetas de classe, etc. e são dependentes do problema. Grandes corpora de texto normalmente contêm alguma informação sobre a co-ocorrência de a's e b's, mas nunca o suficiente para especificar completamente $p(a, b)$ para todos os possíveis pares (a, b), uma vez que as palavras em b tipicamente são esparsas. O problema então seria encontrar um método que, usando a evidência esparsa sobre a's e b's, possa estimar um modelo de probabilidade $p(a, b)$ viável.

Modelos de Entropia Máxima são modelos exponenciais que implementam a intuição de que, se não existe nenhuma evidência que favoreça uma alternativa de solução em relação a uma outra, então ambas as alternativas devem ser consideradas com probabilidades iguais (KOELING, 2000). De forma mais objetiva, o princípio da entropia máxima afirma que a distribuição $p(a, b)$ mais correta é aquela que maximiza a entropia, ou incerteza, e que respeita determinadas restrições que representam as evidências (os

[4]Segundo BIDERMAN (1999) *unidades léxicas* "tratam-se das unidades que coincidem com uma seqüência gráfica indecomponível".

fatos) conhecidas para aquele experimento. Ou seja, se A denota o conjunto de possíveis classes e B denota o conjunto de possíveis contextos, a melhor distribuição p, consistente com as "informações parciais" existentes, é a que maximiza a entropia

$$H(p) = -\sum_{x \in S} p(x) \log p(x)$$

onde $x = (a, b)$, $a \in A$, $b \in B$ e $S = A \times B$ (RATNAPARKHI, 1997).

Para a estimação de parâmetros, geralmente é usado o algoritmo *Improved Iterative Scaling (IIS)* SKUT (1998), no qual é assumido que p tem a forma:

$$p(a|b) = \frac{1}{Z(b)} . e^{\sum_i \lambda_i f_i(a,b)}$$

onde $f_i(a, b)$ é uma função de valor binário e que verifica, em (a,b), uma característica de interesse para o problema. λ_i é um parâmetro que indica o quanto f_i é importante para o modelo e $Z(b)$ é um fator de normalização.

Com relação à identificação de SNs com Entropia Máxima, encontramos os trabalhos de SKUT (1998) e (KOELING, 2000) que usam essa técnica para a tarefa de análise sintática parcial, onde uma das estruturas identificadas nessa análise são os sintagmas nominais básicos[5]. Em SKUT (1998) é reportada uma precisão de 87% na identificação de SNs básicos, e em (KOELING, 2000) a precisão e a abrangência reportadas são de aproximadamente de 93%.

3.2.2 MODELOS OCULTOS DE MARKOV

Segundo BERGER (2001), um *processo estocástico* é uma máquina de estados que gera uma sequência de valores de saída $o = \{o_1, o_2, o_3 ... o_n\}$, por exemplo, pixels de uma imagem, ganhadores de corrida de cavalos, palavras em um texto, etc. Um processo estocástico é chamado de Markoviano (*Modelo de Markov*) se o estado da máquina no tempo $t + 1$ e no tempo $t - 1$ forem independentes, dado o estado no tempo t:

$$p(o_{t+1} \mid o_{t-1} o_t) = p(o_{t+1} \mid o_t) \quad \text{e} \quad p(o_{t-1} \mid o_t o_{t+1}) = p(o_{t-1} \mid o_t)$$

Em outras palavras, num Modelo de Markov, o estado anterior e o futuro são independentes, dado o presente estado.

Uma *Cadeia de Markov* é um método gráfico para representar essa propriedade de independência estatística desse modelo, e nada mais é do que um autômato finito com

[5]RAMSHAW (1995) define SN básico como um SN não recursivo que inclui apenas o núcleo e seus determinantes e pré-modificadores.

probabilidades associadas aos elementos de transição. A FIG. 3.1, mostra um exemplo de uma Cadeia de Markov. Nesse modelo, os estados são representados por círculos, onde no centro encontra-se o nome do estado. O estado inicial é indicado por uma seta de chegada. As transições possíveis são representadas por setas conectando os estados. Tais setas são rotuladas com a probabilidade de esta transição ser percorrida, dado que se está no estado de onde sai a seta. Transições com probabilidade zero não são mostradas no diagrama. A soma das probabilidades das setas que partem de um estado é 1.

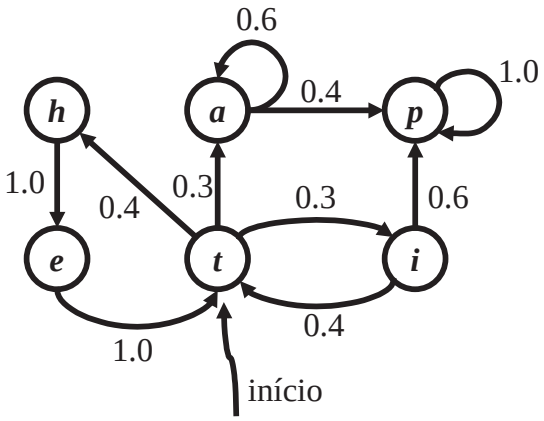

FIG 3.1: Um Modelo de Markov (MANNING, 1999)

Um modelo de Markov com n estados é caracterizado por n^2 probabilidades de transições $p(i, j)$ – a probabilidade de que o modelo se moverá para o estado j, estando no estado i. Dada uma sequência de estados observados (corpus de treino), pode-se estimar a probabilidade para cada entrada da matriz $n \times n$ pelo seguinte cálculo: $p(i, j)$ é dado pelo número de vezes em que o estado j seguiu o estado i, dividido pelo número de vezes que o estado i apareceu antes de qualquer estado.

Modelos Ocultos de Markov (*Hidden Markov Models* – HMMs) são uma generalização dos modelos de Markov. Nos modelos de Markov convencionais o estado e a saída observados no tempo i representam o mesmo elemento; num Modelo Oculto de Markov o estado e a saída estão separados. Mais especificamente, num HMM o autômato gera probabilisticamente um símbolo a cada estado; somente o símbolo, e não a identidade do estado, é visível (BERGER, 2001). Daí o nome "oculto".

Um HMM é descrito formalmente como uma quádrupla $< e^1; E; P; A >$ na qual E é o conjunto de estados $(e^1, e^2, ..., e^m)$, $e^1 \in E$ é o estado inicial do modelo, P é um conjunto de símbolos de saída $(p^1, p^2, ..., p^n)$, e A é um conjunto de arcos ou transições $(a^1, a^2, ..., a^{\in})$. Assim, uma transição que consome uma palavra p^k mudando de estado,

é escrita como $e^i \longrightarrow^{p^k} e^j$ (BONFANTE, 2003).

HMMs são bastante utilizados para o reconhecimento de voz. Outro uso comum de HMMs é a tarefa de etiquetagem (anotação) de corpus, que envolve associar categorias gramaticais ou semânticas às palavras de um texto.

MOLINA (2002) utiliza HMM para a tarefa de análise sintática parcial do inglês, tratando a tarefa como um problema de etiquetagem, onde o objetivo é associar a cada palavra uma etiqueta que indica o seu tipo de constituinte não recursivo. Do ponto de vista estatístico, o problema de etiquetagem pode ser resolvido como um problema de maximização, como é mostrado a seguir.

Seja $\mathcal{O}$ um conjunto de etiquetas de saída e $\mathcal{I}$ o vocabulário de entrada da aplicação. Dado uma sentença de entrada $I = i_1, ..., i_T$, onde $i_j \in \mathcal{I} \ : \ \forall_j$, o processo consiste em encontrar a sequência de estados que maximiza a probabilidade do modelo. Isto é, a sequência de etiquetas de saída $O = o_1, ..., o_T$, onde $o_j \in \mathcal{O} \ : \ \forall_j$. Este processo pode ser formalizado da seguinte forma:

$$\begin{aligned} \hat{O} &= \arg\max_{O} P(O|I) \\ &= \arg\max_{O} \left(\frac{P(O) \ . \ P(I|O)}{P(I)} \right); O \in \mathcal{O}^T \end{aligned} \tag{3.1}$$

Devido ao fato de que esse processo de maximização é independente da sequência de entrada, e assumindo uma modelagem markoviana de segunda ordem, o problema é reduzido à resolução da seguinte equação:

$$\arg\max_{O_1 ... O_T} \left(\prod_{j:1...T} P(o_j|o_{j-1}, o_{j-2}).P(i_j|o_j) \right) \tag{3.2}$$

Os parâmetros da EQ. 3.2 podem ser representados por um HMM de segunda ordem cujos estados correspondem a um par de etiquetas. Probabilidades contextuais, $P(o_j|o_{j-1}, o_{j-2})$, representam as probabilidades das transições entre os estados e $P(i_j|o_j)$ representam as probabilidades das saídas.

Dentro desse contexto de análise sintática parcial do inglês com HMM, MOLINA (2002) consegue identificar SNs básicos com 92,3% de precisão e 92,68% de abrangência.

3.2.3 APRENDIZADO BASEADO EM MEMÓRIA

O Aprendizado Baseado em Memória (*Memory-Based Learning* – MBL) é uma forma de aprendizado supervisionado e induzido a partir de exemplos. Segundo VEENSTRA (1998) um exemplo (ou caso) é representado por um vetor contendo valores de traços

e uma etiqueta que identifica a sua classificação. Durante o treinamento, um conjunto de exemplos (o corpus de treino) é fornecido, de forma incremental, para o algoritmo de treinamento e os exemplos são adicionados à memória (a base de casos). Na classificação de novos casos, para cada caso X a ser classificado é computada a sua distância em relação a cada exemplo Y existente na memória. A etiqueta de classe do exemplo mais próximo, ou seja, que tem maior similaridade com o caso X, é usada para classificar X. Esta abordagem é baseada na premissa de que o raciocínio é baseado no reuso direto da experiência adquirida, ao invés da aplicação do conhecimento abstraído da experiência, como regras ou árvores de decisões.

Em IA, o conceito de MBL tem aparecido em diversas disciplinas (de visão computacional a robótica) usando nomes tais como: baseado em exemplos, baseado em casos, baseado em similaridades, baseado em instâncias, etc (VEENSTRA, 1998).

A métrica de similaridade (ou distância) mais adotada para casos constituídos por traços simbólicos (traços não numéricos como unidades lexicais, etiquetas, etc.) é a métrica *Overlap*, dada na EQ. 3.3:

$$\Delta(X, Y) = \sum_{i=1}^{n} w_i \ \delta(x_i, y_i) \tag{3.3}$$

onde $\Delta(X, Y)$ é a distância entre os casos X e Y, representados por vetores de n traços; w_i é um peso associado ao traço i; e δ é a distância entre os traços x_i e y_i, onde:

$$\delta(x_i, y_i) = \begin{cases} 0 & \text{se } \ x_i = y_i \\ 1 & \text{caso contrário} \end{cases}$$

Em VEENSTRA (1998), é apresentada uma aplicação de MBL para a identificação de SNs básicos do inglês. Os corpora para treino e teste são os mesmos utilizados por RAMSHAW (1995) e resultados reportados são de 89% de precisão e 94.3% de abrangência. VEENSTRA (2000) obteve 90.69% de precisão e 92.86% de abrangência na identificação de SNs básicos do inglês, utilizando MBL na tarefa de análise sintática parcial.

4 APRENDIZADO BASEADO EM TRANSFORMAÇÕES

Nesse capítulo é feita a apresentação do método de aprendizado chamado Aprendizado Baseado em Transformações. É mostrado inicialmente como funciona essa abordagem de aprendizado, e em seguida são mostradas algumas vantagens e desvantagens do uso desse método em relação às abordagens estocásticas.

4.1 O ALGORITMO TBL

Aprendizado Baseado em Transformações (TBL) é um dos algoritmos de aprendizado de máquina baseados em regras mais bem sucedidos. Foi introduzido por Eric Brill em (BRILL, 1992), e tem sido utilizado para diversas tarefas importantes de PLN tais como: etiquetagem morfossintática (BRILL, 1995); identificação de sintagmas nominais do inglês (RAMSHAW, 1995); análise sintática - *parsing* (BRILL, 1996); desambiguação da ligação do sintagma preposicionado - *prepositional phrase attachment* (BRILL, 1994); etiquetagem de atos de diálogo - *dialog act tagging* (SAMUEL, 1998b); correção ortográfica - *spelling correction* (MANGU, 1997); análise sintática parcial - *shallow parsing* (RAMSHAW, 1995; FLORIAN, 2000; MEGYESI, 2002); desambigüação do significado das palavras - *word sense disambiguation* (FLORIAN, 2001) e identificação dos limites das sentenças - *sentence boundary disambiguation* (STAMATATOS, 1999).

A idéia central do algoritmo TBL é gerar uma lista ordenada de regras que melhoram progressivamente uma classificação inicial atribuída aos itens do corpus de treino. O TBL é considerado como um algoritmo guloso, visto que, a cada iteração, a regra escolhida para ingressar na lista de regras aprendidas é aquela que provocar maior redução de erros na classificação atual dos itens do corpus de treino.

As seguintes definições e notações serão utilizadas no decorrer desse trabalho.

- S denota o *espaço de amostras* (itens do corpus).
- C denota o *conjunto de classificações possíveis*. Por exemplo, na primeira tarefa apresentada no item anterior, C seria o conjunto de etiquetas morfossintáticas que podem ser atribuídas às palavras (N, ART, PREP, V, VAUX, etc.) e na segunda tarefa, C seria o conjunto de etiquetas {I,B,O}.

- $C[s]$ denota a classificação associada ao item do corpus s, e $T[s]$ denota a classificação correta do item s;

- e indica uma *expressão condicional* envolvendo traços e valores válidos para esses traços, e que pode ser testada num determinado item de S;

- Uma *regra* r é definida como um par (expressão condicional, classificação), (e, $ftr = c$), onde ftr é um traço, c é a nova classificação a ser atribuída a ftr e $c \in C$. Por convenção, esse par será tratado apenas por (e,c) e c será chamado de *conclusão da regra*;

- O conjunto de todas as regras é representado por R;

- Se $r = (e, c)$, e_r denotará e e c_r denotará c;

- Uma regra $r = (e_r, c_r)$ aplica-se a um determinado item do corpus s se $e_r(s) =$ *verdadeiro* e $c_r \neq C[s]$; Denotaremos de s_r o item s no qual a regra r foi aplicada.

Para a utilização do algoritmo TBL são necessários os seguintes itens de entrada:

- um corpus de treino contendo a classificação correta para algum traço lingüístico que deseja-se aprender a classificar;

- uma classificador básico (*Base-Line System*), utilizado para atribuir uma classificação inicial aos itens do corpus, geralmente baseada em freqüências verificadas no corpus de treino;

- um conjunto de *moldes de regras (templates)*. Esses *moldes* determinam os tipos de expressões condicionais das regras geradas, indicando combinações de traços, na vizinhança de um item, que possam determinar a classificação desse item. Os moldes são os elementos que provocam maior impacto no comportamento do aprendizado com TBL, visto que eles devem exprimir exatamente as informações contextuais importantes para o problema em questão. Segundo CORSTON-OLIVER (2003), as abordagens atuais de TBL dependem crucialmente da pré-seleção de todos e somente os moldes de regras que são relevantes para o problema. A falha na escolha desses moldes pode levar ao insucesso nos resultados.

- Uma função objetivo f para o aprendizado. Diferente de outros algoritmos de aprendizado, a função objetivo para TBL irá diretamente otimizar a função de

avaliação. O tipo de função objetivo mais utilizada em TBL é a seguinte, que representa o ganho de performance resultante da aplicação da regra:

$$f(r) = good(r) - bad(r)$$

onde:

$$good(r) = |\{s|C[s] \neq T[s] \wedge C[s_r] = T[s]\}|$$
$$bad(r) = |\{s|C[s] = T[s] \wedge C[s_r] \neq T[s]\}|$$

O valor de $f(r)$ será denotado por *pontuação* (*score*) de r.

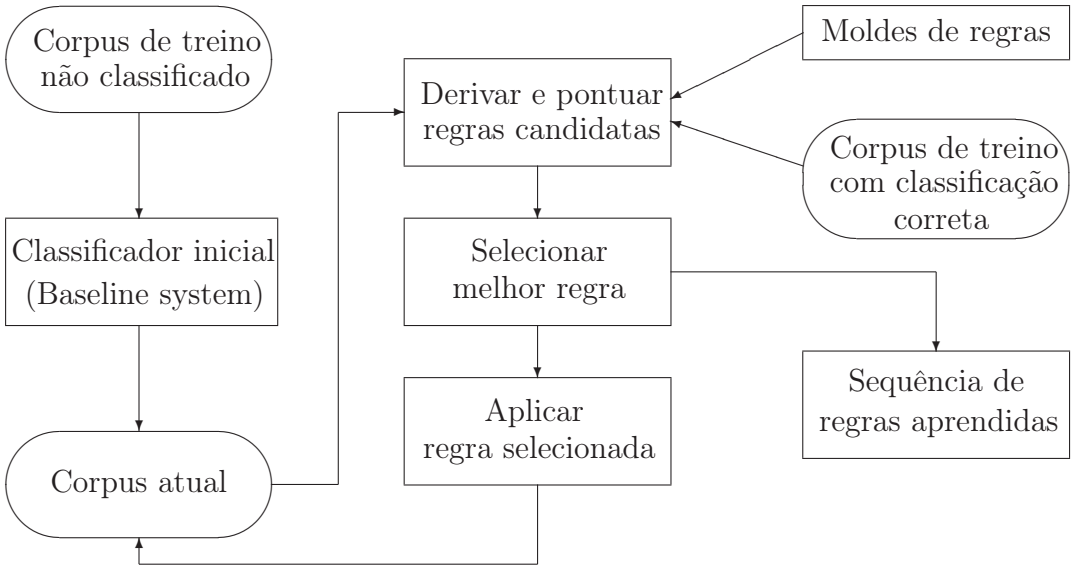

FIG 4.1: Aprendizado Baseado em Transformações

A FIG. 4.1 mostra o processo de aprendizagem utilizado pelo método TBL. O aprendizado inicia-se com a atribuição de uma classificação inicial aos itens do corpus de treino com a utilização do classificador inicial (*base-line system*). Em seguida, a classificação resultante é comparada com a classificação correta e em cada ponto em que houver erro, todas as regras que o corrigem serão geradas a partir da instanciação dos moldes de regras com o contexto do item atualmente analisado. Normalmente uma regra r irá corrigir alguns erros ($good(r)$), mas também poderá provocar outros erros pela alteração de itens que estavam classificados corretamente ($bad(r)$). Dessa forma, após computados os valores de *good* e *bad* para todas as regras candidatas, a regra que tiver maior pontuação será selecionada e colocada na lista de regras aprendidas. A regra selecionada é então

aplicada ao corpus, e o processo de geração de regras será reiniciado enquanto for possível gerar regras com pontuação acima de um limite especificado.

Na classificação de novos textos com uso dessa técnica necessitamos apenas submeter o texto ao classificador inicial, e logo em seguida aplicar a lista de regras na seqüência em que foram aprendidas.

A FIG. 4.2 mostra um exemplo de aprendizado usando TBL. Nesse exemplo, assume-se que existem apenas quatro regras possíveis (R1, R2, R3 e R4) e que deseja-se aprender todas as regras que melhorem a classificação do corpus de treino, ou seja, regras com pontuação maior que zero. O corpus de treino não classificado recebe uma classificação inicial, e o resultado é um corpus contendo 5.100 erros de classificação. Logo em seguida as regras possíveis são geradas e têm suas pontuações computadas. Nesse caso, nota-se que a regra R2 é a que tem a melhor pontuação, desse modo, ela é colocada como a primeira regra aprendida. R2 é aplicada ao corpus e o aprendizado continua. Verifica-se agora que a regra com a maior pontuação é a R3, então R3 é escolhida para entrar na lista de regras aprendidas, e em seguida é aplicada ao corpus. Após a aplicação da regra R3 verifica-se que não existem mais regras com pontuação acima de zero e o processo de aprendizado termina.

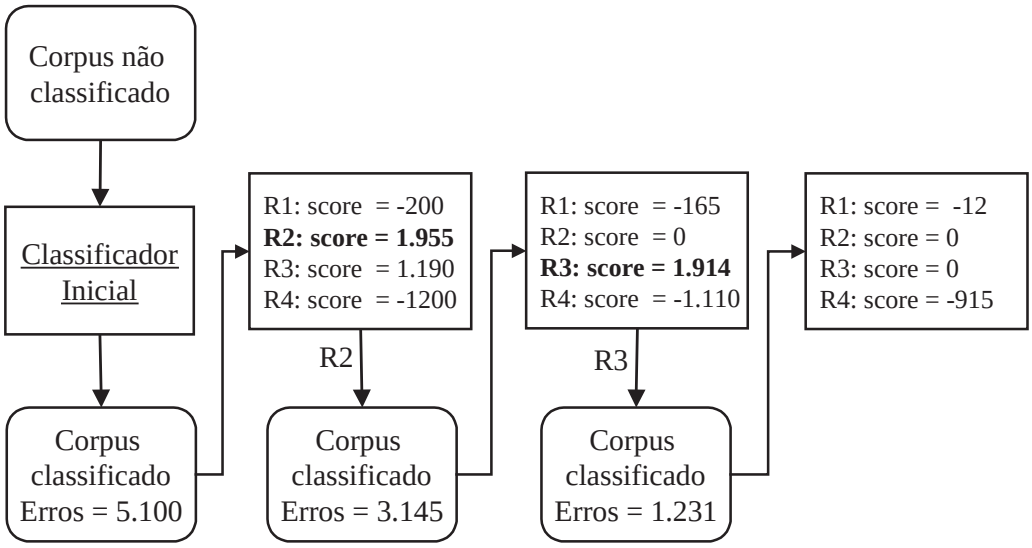

FIG 4.2: Exemplo de aprendizado com TBL (adaptado de AIRES (2000))

Na identificação de SNs básicos do inglês com TBL, RAMSHAW (1995) obteve 92,3% de abrangência e 91,8% de precisão, utilizando um corpus de treino contendo 200 mil itens e um corpus de teste com 50 mil itens. Em MEGYESI (2002) são reportados resultados de 99,39% de precisão e 99,12% de abrangência na identificação de SNs básicos em textos

em lingua sueca com o uso de TBL, sendo que a identificação é feita como parte da tarefa de análise sintática parcial. MEGYESI (2002) usou um corpus de treino contendo 200 mil itens e um corpus de teste contendo aproximadamente 100 mil itens.

4.2 REGRAS E MOLDES DE REGRAS

As regras contextuais geradas pelo método TBL seguem o formato:

$$\langle t_1 \rangle = val_1 \;\; \langle t_2 \rangle = val_2 \;\; \langle t_3 \rangle = val_3 \;\; ... \;\; \langle t_n \rangle = val_n \;\; \rightarrow \;\; \langle ftr \rangle = val$$

No lado esquerdo da seta existe uma expressão condicional formada pela conjunção de pares $\langle t_i \rangle = val_i$, onde t_i é um *Termo Atômico* (TA) e val_i é um valor válido para ele. Do lado direito da seta é indicada a associação do valor val ao traço ftr. Uma regra aplica-se a um determinado item do corpus (item alvo), se nesse item $ftr \neq val$ e a expressão condicional for verdadeira. A expressão condicional é verificada substituindo-se os termos $\langle t_i \rangle$ por valores de traços de itens presentes na vizinhança do item alvo. Se uma regra aplica-se a um item, então a associação de valor especificada do lado direito pode ser realizada nesse item.

Como foi mencionado na seção anterior, o método TBL gera regras com base em moldes que determinam os tipos de expressões condicionais possíveis. Nos pares (TA,val) de uma expressão condicional, o TA determina o item e o traço que, durante o processo de aprendizado, terá seu valor capturado em val para compor a regra. Dessa forma, um molde é simplesmente uma sequência de TAs:

$$\langle t_1 \rangle \;\; \langle t_2 \rangle \;\; \langle t_3 \rangle \;\; ... \;\; \langle t_n \rangle$$

Nas aplicações de TBL encontradas na literatura, os TAs geralmente possuem um dos formatos[6] (a) e (b) mostrados a seguir:

(a) ***ftr_índice***: captura o traço *ftr* de um item que se encontra deslocado, para a esquerda ou para a direita, *índice* posições em relação ao item alvo. Exemplos de TAs para tal padrão seriam: $word\_0$, que identifica o traço *word* do item alvo; $tpos\_-1$ e $tpos\_2$ que capturam, respectivamente, o traço *tpos* do item uma posição anterior e duas posições posteriores ao item alvo.

(b) ***ftr [índice_inicial; índice_final]***: captura o traço *ftr* num intervalo de itens posicionados entre *índice_inicial* e *índice_final*, em relação ao item alvo. Um

[6]Isso no caso de regras contextuais, visto que no caso de etiquetagem morfossintática também são utilizados moldes que verificam por exemplo o sufixo/prefixo de um traço.

exemplo de TA para tal padrão seria $word[1;3]$, que captura uma determinada unidade léxica nos itens das posições +1, +2 e +3 em relação ao item alvo.

O seguinte molde de regras é formado por esses padrões de TAs:

$$tsn\_-1 \;\; tsn\_0 \;\; tpos\_0 \;\; word[1;2]$$

Se usarmos esse molde para gerar regras que corrijam o erro de classificação da preposição *em* na sentença (4.1) – que está marcada como I e deveria estar como O:

(4.1) A/ART/I menina/N/I deixou/V/O a/ART/I boneca/N/I **em/PREP/I** a/ART/I cama/N/I

serão geradas as seguintes regras:

$$tsn\_-1 = I \;\; tsn\_0 = I \;\; tpos\_0 = PREP \;\; word[1;2] = a \;\; \Rightarrow \;\; tsn = O$$

$$tsn\_-1 = I \;\; tsn\_0 = I \;\; tpos\_0 = PREP \;\; word[1;2] = cama \;\; \Rightarrow \;\; tsn = O$$

cuja primeira regra deve ser lida como, "**Se** $tsn\_-1 = I$ **e** $tsn\_0 = I$ **e** $tpos\_0 = PREP$ **e** $word[1;2] = a$ **Então** $tsn\_0 = O$" (**Se** o traço tsn do item anterior tiver valor I **e** os traços tsn e $tpos$ do item atual (alvo) forem iguais a I e $PREP$, respectivamente, **e** o traço $word$ de um dos dois próximos itens for *"a"* **Então** mudar o valor do traço tsn para O no item alvo).

4.3 VANTAGENS E DESVANTAGENS DO MÉTODO TBL EM RELAÇÃO ÀS ABORDAGENS ESTOCÁSTICAS

A característica mais atrativa do TBL é que as regras aprendidas são interpretáveis pelos humanos. Além disso, a lista de regras tende a ser bem mais econômica que a saída de etiquetadores estocásticos. Segundo BRILL (1995), 200 regras TBL aprendidas num corpus de 64.000 palavras produzem uma precisão de etiquetagem morfossintática comparável à de um conjunto de 10.000 probabilidades contextuais emitidas por um etiquetador estocástico. Outra vantagem é que a aplicação das regras aprendidas pode ser executada de forma bem mais rápida do que a classificação com etiquetadores estocásticos.

As desvantagens do TBL estão mais relacionadas à questão do tempo de treinamento. O algoritmo de treinamento pode ter tempo de aprendizado inviável para certas tarefas, mas isso pode ser melhorado por esquemas de indexação, por amostragem do conjunto de regras possíveis, ou por assumir independência entre as regras (SAMUEL, 1998a; NGAI, 2001; HEPPLE, 2000). Similarmente, a performance da aplicação das regras aprendidas

também pode ser melhorado, como mostrado nos trabalhos de ROCHE (1995) e SATTA (1996).

4.4 O ALGORITMO FASTTBL

O algoritmo FastTBL é uma variante do TBL. Foi proposto por NGAI (2001) e provê uma grande redução do tempo de treinamento, mantendo a mesma eficácia dos resultados, em relação ao TBL original.

A principal diferença entre o algoritmo FastTBL e o algoritmo TBL original é que, neste último, como vimos na seção 4.1, a cada iteração todas as regras que corrigem algum erro no corpus são geradas e têm seus valores de *good* e *bad* calculados. Já no FastTBL, todas as regras que corrigem pelo menos um erro são geradas na primeira iteração e são armazenadas juntamente com os seus valores de *good* e *bad*. A cada iteração, quando a melhor regra é escolhida e aplicada ao corpus, apenas as regras que são afetadas pelas alterações provocadas no corpus são geradas/localizadas e têm os seus valores de *good* e *bad* atualizados. A vantagem desse algoritmo é que apenas os itens do corpus que estão na vizinhança de um item que foi alterado pela aplicação da melhor regra precisam ser reexaminados, e apenas as regras que se aplicam a eles precisam ter suas pontuações recomputadas. Para identificar essas regras que devem ser atualizadas a cada iteração, usa-se o conjunto de moldes para regerar as regras que se aplicam na vizinhança de cada item que foi alterado pela aplicação da melhor regra do momento. Segundo os testes de NGAI (2001), o FastTBL é de 13 a 139 vezes mais rápido que o TBL original. A FIG. 4.3 mostra o processo de aprendizado do FastTBL.

Para mostrar uma visão mais detalhada do FastTBL as seguintes notações e considerações são necessárias (vide seção 4.1 para rever outras notações):

- $G(r) = \{s \in S | e_r(s) = verdadeiro \text{ e } C[s] \neq c_r \text{ e } c_r = T[s]\}$ - O conjunto de itens do corpus para o qual a regra r se aplica de forma positiva (corrigindo a classificação). Logo, $good(r) = |G(r)|$.

- $B(r) = \{s \in S | e_r(s) = verdadeiro \text{ e } C[s] \neq c_r \text{ e } C[s] = T[s]\}$ - O conjunto de itens do corpus para o qual a regra r se aplica de forma negativa (alterando de uma classificação correta para uma errada). Logo, $bad(r) = |B(r)|$.

Dada uma nova regra aprendida b que será aplicada a S, o objetivo é identificar as regras r para as quais pelo menos um dos conjuntos $G(r)$ e $B(r)$ são modificados pela

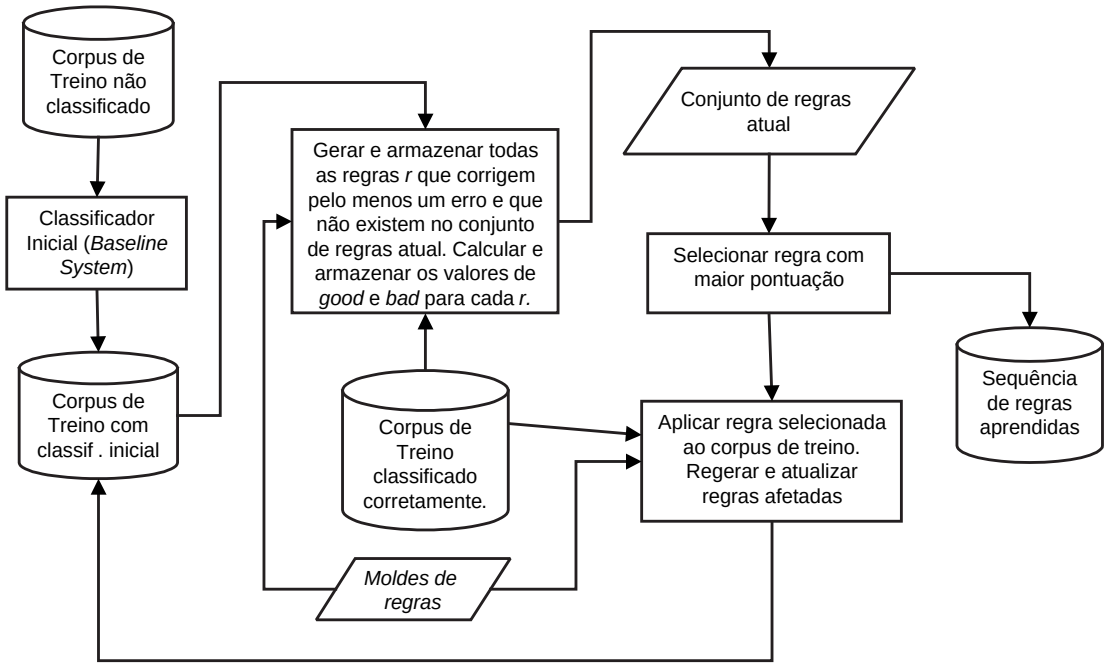

FIG 4.3: Fast Transformation-Based Learning

aplicação da regra b. Obviamente, se ambos os conjuntos não forem modificados quando aplicamos b, então o valor da função objetivo da regra r continua o mesmo.

Seja s um item cuja melhor regra b se aplica (ou seja, $C[s_b] \neq C[s]$). Necessitamos identificar as regras r que são afetadas pela alteração $s \rightarrow s_b$. Seja r uma dessas regras. $f(r)$ precisa ser atualizado se e somente se existe pelo menos um item s' tal que:

$$s' \in G(r) \;\; e \;\; s'_b \notin G(r) \;\; ou \tag{4.1}$$

$$s' \in B(r) \;\; e \;\; s'_b \notin B(r) \;\; ou \tag{4.2}$$

$$s' \notin G(r) \;\; e \;\; s'_b \in G(r) \;\; ou \tag{4.3}$$

$$s' \notin B(r) \;\; e \;\; s'_b \in B(r) \tag{4.4}$$

As condições anteriores correspondem a atualizações específicas dos valores de $bad(r)$ ou $good(r)$.

Em várias tarefas de PLN os itens (palavras) do corpus (S) não são independentes. Por exemplo, na etiquetagem morfossintática, um item é dependente da classificação dos 2 itens antecedentes e posteriores (assume-se que existe uma ordem natural dos itens em S). Vamos denominar de *vizinhança* de um item s (denotado por $V(s)$) o conjunto de itens dos quais a classificação de s pode depender, ou seja, o conjunto de itens que formarão o contexto das regras geradas para corrigir um erro em s (por consistência,

$s \in V(s)$). Nas tarefas onde a classificação de um item é independente, então $V(s) = s$, ou seja, o contexto é formado apenas pelo próprio item.

O algoritmo FastTBL usa o conceito de vizinhança para identificar quais regras devem ser atualizadas quando da alteração de um item. Quando aplicamos uma regra b a um item s, as regras que podem ser afetadas são exatamente as regras que se aplicam a itens s' presentes em $V(s)$. Por isso, apenas os itens presentes em $V(s)$ precisam ser verificados. Qualquer outro item s' que não esteja no conjunto

$$\bigcup_{\{s|b\ modifica\ s\}} V(s)$$

pode ser ignorado, visto que $s' = s'_b$.

A FIG. 4.4 mostra um pseudo código para o algoritmo FastTBL, adaptado de (NGAI, 2001).

No algoritmo assume-se que durante todo o processo de aprendizado todas as regras que possuem o valor de $good > 0$ estão armazenadas e tiveram suas pontuações calculadas. Como dito anteriormente, no início do algoritmo todas as regras que corrigem pelo menos um erro são geradas. Logo em seguida, para calcularmos os valores de *bad* dessas regras, são geradas todas as expressões condicionais e que se aplicam a cada item s que esteja classificado corretamente ($C[s] = T[s]$) e, para cada regra r que tem e como expressão condicional e como conclusão um valor diferente da classificação atual (e correta) de s ($e_r = e$ e $c_r \neq C[s]$), o valor de *bad* é incrementado. A partir desse ponto, o algoritmo entra num laço onde, a cada iteração, a regra b com maior pontuação é escolhida e aplicada ao corpus. Durante a atualização do corpus, cada item alterado por b tem a sua vizinhança examinada para que as regras que se aplicam nessa vizinhança tenham os seus valores de *bad* ou *good* atualizados (caso seja necessário). Observe na FIG. 4.4 que os pontos (4.1), (4.2), (4.3) e (4.4) correspondem, respectivamente, às condições 4.1, 4.2, 4.3 e 4.4 mostradas anteriormente. Essas condições de atualização são tratadas de formas diferentes quando a classificação do item s' analisado foi alterado pela regra b ou não (teste $C[s'] = C[s'_b]$). Uma análise mais detalhada dessas condições de atualização pode ser encontrada em (NGAI, 2001). O laço iterativo, em que a melhor regra b é escolhida e aplicada, é repetido enquanto $f(b)$ for maior que um limite especificado (este limite deve ser maior ou igual a zero).

Para todos os itens s, atribuir uma classificação inicial $C[s]$;

Para todos os itens s que satisfazem $C[s] \neq T[s]$, gerar todas as regras r que corrigem a classificação de s, incrementando $good(r)$;

Para todos os itens s que satisfazem $C[s] = T[s]$:
gerar todas as expressões condicionais e tal que $e(s) = verdadeiro$;
incrementar $bad(r)$ para todas as regras r tal que $e_r = e$ e $c_r \neq C[s]$;

Selecionar a regra $b = \arg\max_{r \in R} f(r)$;
Enquanto ($f(b)$ > GANHO_INSIGNIFICANTE)

Para cada expressão condicional e, seja $R(e)$ o conjunto de regras cuja expressão condicional é e $(e_r = r)$;

Para cada item s, s' s.t. $C[s] \neq C[s_b]$ e $s' \in V(s)$:

Se $C[s'] = C[s'_b]$ Então
- **Para** cada expressão condicional e s.t. $e(s') = verdadeiro$
  - **Se** $C[s'] \neq T[s']$ **Então**

(4.1) **Se** $e(s'_b) = falso$ **Então** decrementar $good(r)$, onde $r = [e, T[s']]$ (a regra que tem e como expressão condicional e $T[s]$ como conclusão);

  - **Senão**

(4.2) **Se** $e(s'_b) = falso$ **Então** decrementar $bad(r)$ para todas as regras $r \in R(e)$ s.t. $c_r \neq C[s']$;

- **Para** cada expressão condicional e s.t. $e(s'_b) = verdadeiro$
  - **Se** $C[s'_b] \neq T[s']$ **Então**

(4.3) **Se** $e(s') = falso$ **Então** incrementar $good(r)$, onde $r = [e, T[s']]$;

  - **Senão**

(4.4) **Se** $e(s') = falso$ **Então** incrementar $bad(r)$ para todas as regras $r \in R(e)$ s.t. $c_r \neq C[s'_b)]$;

Senão
- **Para** cada expressão condicional e s.t. $e(s') = verdadeiro$
  - **Se** $C[s'] \neq T[s']$ **Então**

(4.1) **Se** $e(s'_b) = falso \ \vee \ C[s'_b] = c_r$ **Então** decrementar $good(r)$, onde $r = [e, T[s']]$;

  - **Senão**

(4.2) Decrementar $bad(r)$ para todas as as regras $r \in R(e)$ s.t. $c_r \neq C[s']$;

- **Para** cada expressão condicional e s.t. $e(s'_b) = verdadeiro$
  - **Se** $C[s'_b] \neq T[s']$ **Então**

(4.3) **Se** $e(s') = falso \ \vee \ C[s'] = c_r$ **Então** incrementar $good(r)$, onde $r = [e, T[s']]$;

  - **Senão**

(4.4) Incrementar $bad(r)$ para todas as regras $r \in R(e)$ s.t. $c_r \neq C[s'_b]$;

$b = \arg\max_{r \in R} f(r)$

FIG 4.4: Algoritmo FastTBL

5 FERRAMENTA TBL PROPOSTA

Nesse capítulo é apresentada a ferramenta TBL desenvolvida como uma das atividades desse trabalho. Tal ferramenta conta com uma novidade em relação às outras implementações do algoritmo TBL encontradas na literatura: os Termos Atômicos com restrição, que dão maior poder de especialização e mobilidade para as regras.

5.1 TERMO ATÔMICO COM RESTRIÇÃO: UMA NOVA ABORDAGEM DE MOLDE DE REGRAS PARA O TBL

Os tipos de TAs (a) e (b) mostrados na seção 4.2 são apropriados para tarefas em que o tamanho da janela de contexto – onde a informação que leva à classificação correta de um item deve ser encontrada – é bem delimitado e relativamente pequeno. Nas aplicações de TBL encontradas na literatura geralmente é utilizada uma janela de sete itens de tamanho, incluindo o item alvo, os três anteriores e os três posteriores. Esse tipo de molde não é viável para certos problemas de PLN onde a classificação depende de itens com distâncias variáveis entre si.

A identificação de Sintagmas Nominais que incluem preposições é um desses problemas. Essa tarefa é mais complexa do que a identificação de SNs básicos, visto que envolve o problema de ligação do sintagma preposicionado, que é a questão de distinguir quando a preposição está introduzindo um complemento de um verbo (e deve ser classificada como fora do SN anterior a ela), ou quando está introduzindo o complemento de um nome (e deve ser classificada como pertencente ao SN anterior). Esse é um caso onde uma janela de contexto com sete itens de tamanho é insuficiente.

A idéia mais proeminente que surge dessa descrição do problema de identificação de SNs é habilitar a verificação de uma possível dependência entre a preposição a ser classificada e o verbo que a precede. Ou seja, gerar regras específicas para observar uma preposição e o verbo que a antecede. Mas para se fazer isso, os seguintes obstáculos relacionados aos TAs (a) e (b) devem ser superados:

(1) Quando as expressões condicionais estão sendo geradas com o uso dos TAs (a) e (b), o valor do traço indicado é sempre capturado independentemente de qualquer pré-condição em relação ao correspondente item, exceto a distância em relação ao item alvo. Com os TAs (a) e (b) não seria possível capturar o valor do traço indicado

somente se o item atender a alguma condição. Por exemplo, não seria possível definir um TA que, durante a geração de uma regra, só capture a unidade léxica (traço *word*) de um item se ele atender ao requisito de ser uma preposição (traço $tpos = PREP$); e caso contrário, a regra não seja criada.

(2) Uma vez que a distância exata entre a preposição e o verbo que a precede não é conhecida, não é possível a utilização do TA do tipo (a) na tentativa de capturá-los numa mesma expressão condicional. Assumindo que o verbo precedente a uma preposição é encontrado numa janela de tamanho arbitrário, usar o TA do tipo (b) provocaria a geração de diversas regras desnecessárias.

O seguinte exemplo ilustra melhor esses casos. Supondo que se deseja obter uma regra que corrija a classificação de uma preposição (como dentro (I) ou fora (O) de um SN) e cuja expressão condicional é formada por TAs que testam exatamente o item lexical da preposição e do verbo que a antecede. Um molde que poderia gerar regras com esses requisitos, usando os TAs (a) e (b), seria o seguinte:

$$tsn\_0 \;\; word\_0 \;\; word[-2;-7]$$

onde tsn_0 captura o traço relativo à identificação de SNs do item alvo, $word\_0$ captura a unidade léxica do item alvo ("a preposição") e $word[-2;-7]$ captura a unidade léxica dos itens pertencentes ao intervalo dado (supondo-se que o verbo possa se encontrar no intervalo fechado [-2;-7]).

Seja a seguinte sentença, onde cada item contém os três traços já mencionados, seguindo o formato $word/tpos/tsn$:

(5.1) O/ART/I aluno/N/I esqueceu/V/O o/ART/I caderno/N/I de/PREP/I caligrafia/N/I amarelo/ADJ/O em/PREP/I casa/N/I

Para ilustrar o tipo de problema (1) basta lembrar que todo molde formado pelos TAs descritos é bastante geral e será aplicado para gerar regras em qualquer item que tenha um erro de classificação, e não apenas para preposições. Logo, o molde anterior também seria instanciado para gerar as seguintes regras que corrigem a classificação do termo *amarelo* da sentença (5.1), que está como O e deveria estar como I,

tsn_0 =O $word\_0$=amarelo $word[-2;-7]$=de $\Rightarrow$ tsn=I

tsn_0 =O $word\_0$=amarelo $word[-2;-7]$=caderno $\Rightarrow$ tsn=I

...

tsn_0 =O $word\_0$=amarelo $word[-2;-7]$=O $\Rightarrow$ tsn=I

Para ilustrar o tipo de problema (2) basta verificar que, mesmo quando o molde é aplicado para gerar regras que corrigem a classificação de uma preposição, são geradas diversas outras regras, além da que desejávamos (em negrito), como mostrado a seguir.

tsn_0 =I $word\_0$=em $word[-2;-7]$=caligrafia $\Rightarrow$ tsn=O

tsn_0 =I $word\_0$=em $word[-2;-7]$=de $\Rightarrow$ tsn=O

tsn_0 =I $word\_0$=em $word[-2;-7]$=caderno $\Rightarrow$ tsn=O

tsn_0 =I $word\_0$=em $word[-2;-7]$=o $\Rightarrow$ tsn=O

tsn_0=I word_0=em word[-2;-7]=esqueceu $\Rightarrow$ tsn=O

tsn_0 =I $word\_0$=em $word[-2;-7]$=aluno $\Rightarrow$ tsn=O

Essa criação de regras indesejáveis por esse tipo de molde provoca maior consumo de memória e tempo de execução durante o aprendizado. Quanto maior for o tamanho da janela de contexto usado pelo TA do tipo (b), mais regras desnecessárias serão geradas. E caso essa janela seja pequena, é provável que o verbo que antecede a preposição fique fora do intervalo verificado, e, desse modo, a regra desejada, relacionando a preposição e o verbo, não será criada.

Com a expectativa de contornar essas dificuldades, propomos um novo tipo de TA, o qual denominamos de TA com restrição, que possui uma janela de contexto com tamanho variável e um teste que precede a captura do valor de um traço. O uso desse tipo de TA assume que só se deve capturar o valor de um traço X de um item, se um outro traço Y, no mesmo item, atender a um determinado teste condicional. O teste consiste em verificar se o valor do traço Y é igual a um valor predefinido. O formato do TA proposto é o seguinte:

$$traçoX \ \ [ind\_inicial;ind\_final](traçoY{=}valY)$$

Um exemplo de TA que segue esse padrão é:

$$word[-2;-8](tpos = V)$$

que deve ser interpretado como "Capturar o traço *word* do item mais próximo ao item alvo, que esteja entre o intervalo fechado -2 e -8, e cujo traço *tpos* é igual a *V*".

Com esse tipo de TA podemos construir um molde que gera expressões condicionais que observem exatamente uma preposição e o verbo que a antecede. Tal molde teria a forma:

$$tsn\_0 \;\; word[0;0](tpos = PREP) \;\; word[-2;-10](tpos = V)$$

e quando aplicado para gerar regras na sentença (5.1) só geraria a seguinte regra:

$$tsn\_0 = \text{I} \;\; word[0;0](tpos = PREP) = \text{em} \;\; word[-2;-10](tpos = V) = \text{esqueceu} \;\; \Rightarrow \;\; tsn = \text{O}$$

que deve ser lida como: "**SE** no item alvo (índice 0) o traço *tsn*=I **e** os traços *tpos*=PREP **e** *word*=PREP, **e** o primeiro item no intervalo fechado [-2;-10] que atenda ao teste *tpos*=V também atender a *word*=esqueceu **ENTÃO** mudar o valor do traço *tsn* para *O* no item alvo".

O tipo de TA proposto aumenta consideravelmente o poder de especificação e abrangência dos moldes de regras. Como mostrado, é possível especificar regras que tenham aplicação restringida a itens específicos, e também gerar regras que, mesmo considerando um contexto muito grande, verificam apenas os elementos que são importantes para o problema que está sendo tratado. Um exemplo de aplicação de TAs com restrição na identificação de SNs do português é mostrado no capítulo 6.

5.2 IMPLEMENTAÇÃO DO TA COM RESTRIÇÃO

Para a implementação da ferramenta TBL proposta nesse trabalho foi escolhido o algoritmo *FastTBL*. Optamos por essa versão do algoritmo TBL porque é bem mais rápida que a versão original proposta por Eric Brill e mantém a mesma eficácia dos resultados.

Na implementação dos TAs com restrição, a principal mudança ocorreu no procedimento para a captura dos traços indicados pelos TAs de um molde de regra, tanto na geração quanto na aplicação das regras. Não houve alterações em relação aos procedimentos do algoritmo FastTBL mostrados na FIG. 4.4, visto que o algoritmo FastTBL não indica exatamente o modo como as regras devem ser geradas ou aplicadas.

Será designado de *captura de um traço* o ato de observar um item do corpus e obter o valor de um determinado traço desse item. Esse procedimento é necessário em dois momentos: durante a geração de cada regra (para a instanciação do molde na criação da expressão condicional da regra) e durante a aplicação de uma regra (para capturar os valores dos traços nos itens do contexto indicado pela regra, e com isso verificar se o contexto atende à expressão condicional).

Quando se utiliza um TA sem teste, como $word\_-2$, a captura do traço é realizada de forma direta, ou seja, é só acessar o item da vizinhança com deslocamento igual a -2 e capturar o valor de *word*, caso o item exista. No caso de um TA sem restrição do tipo

$word[-1,-3]$ é necessário percorrer todo o intervalo e gerar uma instância para cada item encontrado no deslocamento. Por exemplo, na sentença (5.1):

(5.1) O/ART/I aluno/N/I esqueceu/V/O o/ART/I caderno/N/I de/PREP/I caligrafia/N/I amarelo/ADJ/O em/PREP/I casa/N/I

se utilizarmos o TA $word\_ - 2$, tendo como alvo o item que contém a preposição *em*, esse TA seria instanciado assim: $word\_ - 2 = caligrafia$. E no caso de um TA do tipo $word[-1,-3]$, teríamos as seguintes instâncias: $word[-1,-3] = amarelo$, $word[-1,-3] = caligrafia$ e $word[-1,-3] = de$. Quando o deslocamento ultrapassa os limites da sentença utilizamos, por convenção, o valor "*ZZZ*" para indicar esse fato. Por exemplo, se quisermos capturar o valor de $word\_ - 2$ tendo como alvo o item que contém a palavra *aluno*, esse TA seria instanciado como $word\_ - 2 = ZZZ$, uma vez que esse item não possui vizinho com deslocamento -2.

Para capturar o traço indicado por um TA com restrição, como o que propomos,

traçoX[ind_inicial;ind_final](traçoY=valY),

não se pode apenas verificar o deslocamento e capturar diretamente o traço do item encontrado; é necessário verificar se o item atende ao teste, ou seja, olhar se o valor do traço *traçoY* é igual a *valY*.

A função *Captura_Traço*, mostrada na FIG. 5.1, implementa os procedimentos necessários para a captura do *traçoX* de um TA com teste. A função recebe como parâmetro um vetor w que representa uma sentença do corpus, um número i que indica o índice do item alvo na sentença, e o TA a ser utilizado. Esse algoritmo percorre os itens da sentença w que pertencem ao intervalo de deslocamento informado no TA, verificando se o *traçoY* do item visitado atende ao teste (*traçoY=valY*). O primeiro item que atender a esse teste terá o valor do *traçoX* capturado e retornado pela função. Durante o percurso, se um dos limites da sentença for alcançado, o valor "*ZZZ*" é retornado. Se o intervalo é completamente percorrido e nenhum item que atenda ao teste for encontrado, o valor *nulo* é retornado e o TA não será instanciado.

Para exemplificar o uso dessa função, sejam os seguintes parâmetros que durante a instanciação de uma regra sejam passados para função *Captura_Traço*: a sentença (5.1), o índice $i = 9$ (o item alvo será o que contém a preposição *em* - posição 9), e o TA *word[-2;-6](tpos=V)*. Seria obtido como retorno o valor do traço *word* ("*esqueceu*") do item de posição -6 em relação ao item alvo. E o TA seria instanciado como:

word[-2;-6](tpos=V)=esqueceu

Mas se for passado como alvo o item que contém a palavra *aluno* ($i = 2$), teríamos como retorno "ZZZ", e o TA seria instanciado assim:

word[-2;-6](tpos=V)=ZZZ

Uma outra possibilidade é ter como alvo o item que contém a preposição *em* ($i = 9$) e o TA *word[-2;-6](tpos=ADJ)* e nesse caso teríamos *nulo* como retorno, e o TA (e conseqüentemente a regra) não seria instanciado. Essa função pode ser utilizada para capturar os traços de TAs com restrição tanto na criação quanto na aplicação de um regra.

```
Função Captura_Traço(w: vetor, i: número inteiro, TermoAt:
Termo Atômico) {
    Seja w uma sentença representada por um vetor com n
    posições, cada posição contém um item da sentença;
    Seja i o índice em w do item alvo da análise;
    Seja traçoX o traço a ser capturado, e seja traçoY o
    traço a ser testado com o valor valY predefinido em
    TermoAt;
    Sejam ini e fin números inteiros representando o intervalo
    de deslocamento definido em TermoAt;
    Seja desloc um número inteiro representando o deslocamento
    em relação ao item alvo i.
    desloc = ini + i;
    Enquanto ( desloc > 0  e  desloc < n + 1  e
    Valor_Absoluto(ini) < Valor_Absoluto(fin) )
        Se o valor do traçoY == valY no item de w com
        posição desloc Então
            Retornar valor do traçoX do item de w que possui
            posição desloc;
        desloc = ±1;
        ini = ±1;
    Se ( Valor_Absoluto(ini) < Valor_Absoluto(fin) ) Então
        Retornar "ZZZ";
    Retornar nulo;
}
```

FIG 5.1: Algoritmo para capturar o traço no TA com restrição

5.3 ALGUNS DETALHES DA IMPLEMENTAÇÃO DA FERRAMENTA TBL PROPOSTA

Essa seção descreve alguns aspectos da implementação e utilização da ferramenta TBL que foi desenvolvida como uma das atividades desse trabalho. Esse programa, denominado de catTBL, está dividido em dois módulos: Treinamento e Aplicação de Regras. Para maiores detalhes sobre o uso da ferramenta (formato dos arquivos de entrada e algumas configurações) vide APÊNDICES 5 e 6.

Tanto a ferramenta de treinamento quanto a ferramenta de aplicação de regras foram desenvolvidas com a linguagem de programação Java^{TM}2. Optamos por utilizar a linguagem Java porque é multi-plataforma e por ser uma novidade em termos de implementação do algoritmo TBL, visto que até a data do desenvolvimento desse software, não tínhamos conhecimento da existência de ferramenta TBL desenvolvida com Java.

Como a linguagem de programação Java é considera mais lenta que outras linguagens como C/C++, foi tomado um cuidado especial na implementação (principalmente em termos de estrutura de dados) para que a ferramenta tivesse desempenho satisfatório.

É importante destacar que o catTBL é uma ferramenta de aprendizado de propósito geral e que pode ser utilizada para várias tarefas de PLN. No momento a ferramenta pode ser utilizada para aprender regras contextuais para problemas como etiquetagem morfossintática, identificação de SNs e análise sintática parcial. Com pequenas adaptações no código pode-se estender a aplicabilidade dessa ferramenta.

5.3.1 FERRAMENTA DE TREINAMENTO

A ferramenta de treinamento foi implementada com base no algoritmo FastTBL, e ainda dispõe da funcionalidade adicional do suporte aos TAs com restrição. A ferramenta tem desempenho totalmente compatível com outras ferramentas TBL, tanto em termos de performance, quanto de eficácia das regras produzidas.

Foi realizada uma comparação do catTBL com a ferramenta fnTBL versão 1.1(Linux), que também implementa o algoritmo FastTBL e foi desenvolvida por Radu Florian e Grace Ngai[7]. Os dois programas foram testados num mesmo computador e com os mesmos dados de entrada, ou seja, mesmo conjunto de moldes, corpus de treino, classificação inicial e critério de parada. O problema de classificação escolhido foi o da identificação

[7]O fnTBL encontra-se disponível para *download* pela internet, no endereço http://nlp.cs.jhu.edu/ rflorian/fntbl/tbl-toolkit/tbl-toolkit.html

de SNs básicos do inglês, usando o corpus de RAMSHAW (1995). As duas ferramentas obtiveram a mesma eficácia dos resultados em termos de precisão e abrangência, inclusive os conjuntos de regras aprendidas foram bastante semelhantes. Em relação ao tempo de execução o fnTBL foi melhor, terminando o treinamento em 10 minutos enquanto o catTBL demorou 12 minutos para concluir o treinamento. Este desempenho já era esperado, visto que o fnTBL foi desenvolvido em C++, e o catTBL foi desenvolvido em Java, que é uma linguagem interpretada e por isso torna-se bem mais lenta.

5.3.2 FERRAMENTA DE APLICAÇÃO DAS REGRAS

A aplicação das regras é a classificação de novos textos com o uso das regras aprendidas no treinamento. Tal aplicação consiste basicamente nos seguintes passos:

- iniciar aplicando uma classificação básica à cada palavra do texto utilizando o classificador inicial;
- aplicar a lista de regras aprendidas; sendo que a seqüência de aplicação deve seguir a seqüência em que as regras foram aprendidas.

A FIG. 5.2 ilustra o processo de aplicação das regras.

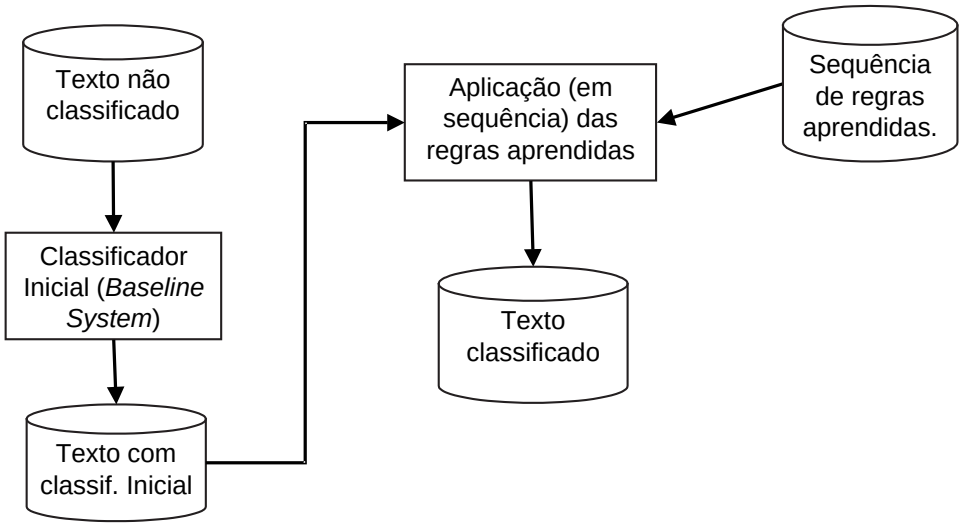

FIG 5.2: Aplicação das regras aprendidas

A ferramenta de aplicação de regras desenvolvida compartilha diversas partes do código da ferramenta de treinamento como a implementação das rotinas de manipulação do corpus, configuração dos traços e moldes de regras, etc. O algoritmo de aplicação de regras ainda não está funcionando em tempo ótimo, visto que, quando se aplica uma

regra, ela é testada em cada item do corpus. Mas esse procedimento pode ser realizado em tempo linear, conforme indica ROCHE (1995), codificando-se a lista de regras num transdutor de estados finitos (uma espécie de autômato finito onde cada transição é rotulada com um par de símbolos: o primeiro é o símbolo de entrada e o segundo o de saída).

6 IDENTIFICAÇÃO DE SINTAGMAS NOMINAIS DO PORTUGUÊS BRASILEIRO COM TBL

Nesse capítulo são relatadas as decisões relativas à identificação de SNs do português brasileiro usando TBL. São tratadas questões como o desenvolvimento dos corpora para treino e teste, classificação inicial e moldes de regras utilizados. O capítulo é encerrado com a apresentação dos resultados obtidos no experimentos.

Além de descrever os itens de entrada necessários para o uso do TBL (corpus de treino, classificação inicial e moldes de regras), esse capítulo também apresenta um pré-processamento – a lematização dos verbos – que foi efetuado na tentativa de melhorar a eficácia e uniformidade das regras aprendidas.

6.1 PRÉ-PROCESSAMENTO: LEMATIZAÇÃO DOS VERBOS

Para os experimentos realizados nesse trabalho, foi desenvolvido um sistema básico para lematização de verbos, que tem a função de receber um verbo e retornar o seu respectivo infinitivo. A TAB. 6.1 mostra alguns exemplos de verbos na forma finita e o seu respectivo formato no infinitivo.

TAB 6.1: Exemplo de verbos na forma finita e no infinitivo

Forma finita	Forma no infinitivo
cresço, cresceu, cresce	crescer
acordo, acorda, acordou	acordar
sumimos, sumido, sumíreis	sumir
comeste, comesse, comerão	comer

O programa lematizador foi desenvolvido em linguagem Java e pode ser acoplado como um módulo da ferramenta catTBL. O procedimento para a lematização é bastante simples, consistindo apenas de um processo de inclusão/exclusão de sufixos – realizado com o auxílio de expressões regulares – e a validação do verbo resultante, com o uso de uma base de verbos no infinitivo. A base de verbos utilizada foi adquirida do Banco de Conjugações de Verbos da Língua Portuguesa versão 1.1, que faz parte do software

Conjugue[8], desenvolvido por Ricardo Ueda Karpischek, e que se encontra disponível sob os termos da licença GNU GPL .

O lematizador de verbos é aplicado ao corpus tanto no treinamento quanto na aplicação das regras. A lematização dos verbos torna-se importante por o português ser uma lingua morfologicamente rica, onde um verbo pode aparecer em dezenas de formas, diferentemente do inglês. Acreditamos que a lematização possa contribuir para a geração de um conjunto de regras mais uniforme. Por exemplo, as três regras seguintes:

$$word[0;0](tpos = \text{PREP}) = \text{em} \;\; word[-1;-20](tpos = \text{V}) = \text{jogou} \rightarrow tsn = \text{O}$$
$$word[0;0](tpos = \text{PREP}) = \text{em} \;\; word[-1;-20](tpos = \text{V}) = \text{jogado} \rightarrow tsn = \text{O}$$
$$word[0;0](tpos = \text{PREP}) = \text{em} \;\; word[-1;-20](tpos = \text{V}) = \text{jogastes} \rightarrow tsn = \text{O}$$

poderiam ser compiladas, com a lematização, na seguinte regra:

$$word[0;0](tpos = \text{PREP}) = \text{em} \;\; word[-1;-20](tpos = \text{V}) = \text{jogar} \rightarrow tsn = \text{O}$$

6.2 DERIVAÇÃO DOS CORPORA DE TREINO E TESTE

6.2.1 ESCOLHA DO CORPUS

Os corpora de treino e teste utilizados nesse estudo foram derivados do Mac-Morpho, um "corpus de 1,1 milhão de palavras retiradas a partir de 1 ano de publicação (1994) do jornal brasileiro Folha de São Paulo" (MARCHI, 2003), disponibilizado via web pelo projeto Lacio-Web[9], do Núcleo Interinstitucional de Lingüística Computacional (NILC)[10]. Tal corpus está etiquetado morfossintaticamente com o conjunto de etiquetas (*tagset*) do projeto Lacio-Web, o LW Tagset. Dois motivos principais nos levaram à escolha desse corpus: primeiro porque este consiste de textos em português brasileiro e em quantidade suficiente para a tarefa de treinamento; segundo porque o conjunto de etiquetas morfossintáticas utilizadas nesse corpus foi desenvolvido tendo como meta a simplicidade e objetividade, na tentativa de garantir requisitos como recuperabilidade, consistência e facilidade de aprendizado automático da tarefa de anotação morfossintática.

O Mac-Morpho é composto por textos de 10 cadernos do jornal Folha de São Paulo, são eles: Brasil (3.877.787 ocorrências), Cotidiano (3.417.450), Dinheiro (3.254.763), Ilustrada (3.003.869), Mais, Agrofolha, Ciências, Informática, Esportes e Mundo. Segundo

[8] *www.ime.usp.br/~ueda/br.ispell*

[9] www.nilc.icmc.usp.br/lacioweb/

[10] www.nilc.icmc.usp.br

MARCHI (2003), a escolha privilegiou cadernos mais importantes, cadernos com estrutura sintática elaborada, cadernos dirigidos ao público adulto e com variedade de nomes próprios.

O LW Tagset (ver APÊNDICE 1), na sua décima versão, "possui um conjunto de 22 etiquetas para as classes gramaticais, além de outras 10 etiquetas complementares e prevê a ocorrência de 'polilexicais' " (MARCHI, 2003). Os chamados polilexicais são expressões multivocabulares (EMV), arbitrariamente escolhidas como itens lexicais, e que por isso recebem apenas um conjunto de etiquetas como se fossem palavras simples.

No website do projeto Lácio-Web, o Mac-Morpho é disponibilizado em dois formatos: um mais adequado para consultas lingüísticas e outro mais adequado para o treinamento de etiquetadores. A versão escolhida foi a versão mais apropriada para consultas, principalmente porque continha as marcações de EMVs (Ex: "São Paulo", "a partir de") que os torna uma única unidade (*token*). Mas, por outro lado, foi necessária a criação de um programa para retirar informações irrelevantes para o treinamento e que constam nessa versão tais como: as etiquetas indicativas de material não etiquetado <NA> ...</NA>; as etiquetas XML para nome de arquivo, título, etc.; as etiquetas complementares para data (DAT), dados (DAD), aposto (AP), hora (HOR), estrangeirismo (EST) e número de telefone (TEL). Retiramos ainda a etiqueta indicadora de contração (|+). A FIG. 6.1 identifica o formato dos arquivos do Mac-Morpho. No total são 109 arquivos, todos com o texto na horizontal, onde cada linha contém uma palavra e a sua etiqueta morfossintática separadas pelo caracter "_".

6.2.2 GERAÇÃO DAS ETIQUETAS IDENTIFICADORAS DOS SNS

Como o método TBL exige um corpus de treino contendo a classificação correta para o problema, necessitamos então de um corpus que contenha a etiqueta SN já associada de forma correta a cada palavra. Como foi utilizado o Mac-Morpho para derivar o corpus de treino, foi necessário encontrar uma forma de gerar essa nova etiqueta nos itens desse corpus.

Para a geração das etiquetas SN, seria necessária a identificação de todos os SNs presentes no Mac-Morpho, o que seria impraticável fazer de forma manual, visto que não dispúnhamos do tempo e mão-de-obra necessários para realizar tal tarefa. A melhor forma encontrada para se fazer essa tarefa automaticamente foi utilizar o analisador sintático de Eckhard Bick, o PALAVRAS (BICK (2000)), para fazer a análise sintática de todo o Mac-Morpho e a partir dessa análise identificar os SNs, visto que a anotação desse

```
Cerca_PREP
de_PREP
650_NUM
homens_N
de_PREP|+
o_ART
policiamento_N
de_PREP
choque_N
fizeram_V
ontem_ADV
uma_ART
megablitz_N|EST
em_PREP|+
as_ART
ruas_N
de_PREP|+
o_ART
centro_N
velho_ADJ
de_PREP
São=Paulo_NPROP
._.
```

FIG 6.1: Exemplo de um trecho do corpus Mac-Morpho. (Arquivo co94ja04.train.txt)

parser é bastante rica e contém informações sintáticas que delimitam os constituintes da sentença.

Uma vez reconhecidos os SNs, cada palavra do Mac-Morpho recebeu a etiqueta SN. Para a realização de todo esse procedimento foram desenvolvidos alguns programas, utilizando-se a linguagem Java. Tais ferramentas são descritas a seguir.

O primeiro programa, *RemoveTags.Java*, recebe como entrada um arquivo no formato do Mac-Morpho (texto etiquetado e na vertical, ver FIG. 6.1), e gera um arquivo no qual o texto foi colocado na forma horizontal, todas as etiquetas morfossintáticas foram eliminadas e foram realizadas as concatenações de palavras (Ex: de o => do). Um exemplo do arquivo de saída desse programa pode ser visto na FIG. 6.2.

Todos os arquivos do Mac-Morpho foram submetidos ao programa *RemoveTags.java* e os arquivos gerados foram submetidos à análise sintática do PALAVRAS, que foi acessado a partir do portal do projeto VISL[11]. O analisador sintático citado possui vários formatos

[11] http://visl.hum.sdu.dk/pt

```
Cerca de 650 homens do policiamento de choque fizeram
ontem uma megablitz nas ruas do centro velho de São
Paulo.
A operação foi determinada pelo comandante do
Policiamento Metropolitano, coronel Carlos Augusto de
Mello Araújo, que até quinta-feira passada respondia
pelo.
```

FIG 6.2: Trecho de um arquivo gerado pelo programa *RemoveTags.java*

de arquivo de saída. O formato utilizado nesse trabalho foi o *flat*[12]. A FIG. 6.3 apresenta um trecho do resultado da análise sintática para um dos arquivos do corpus.

```
Cerca=de [cerca=de] ADV @>A
650 [650] <card> NUM M P @>N
homens [homem] N M P @SUBJ>
de [de] <sam-> PRP @N<
o [o] <-sam> <artd> DET M S @>N
policiamento [policiamento] N M S @P<
de [de] PRP @N<
choque [choque] N M S @P<
fizeram [fazer] <fmc> V PS/MQP 3P IND VFIN @FMV
ontem [ontem] ADV @<ADVL
uma [um] <arti> DET F S @>N
megablitz [blitz] <DERP> N F S @<ACC
em [em] <sam-> PRP @<ADVL
as [o] <-sam> <artd> DET F P @>N
ruas [rua] N F P @P<
de [de] <sam-> PRP @N<
o [o] <-sam> <artd> DET M S @>N
centro [centro] N M S @P<
velho [velho] ADJ M S @N<
de [de] PRP @N<
São=Paulo [São=Paulo] PROP M S @P<
.
```

FIG 6.3: Exemplo da anotação gerada pela análise sintática do *parser* PALAVRAS

No *tagset* do PALAVRAS existem 14 classes principais de categorias de palavras que combinam com 24 etiquetas para categorias de inflexão. Segundo BONFANTE (2003),

[12]O PALAVRAS possui um tipo de formato de arquivo de saída no qual o resultado da análise sintática aparece em formato de árvore, inclusive com uma marcação mais clara dos SNs. Mas infelizmente, no período em que estávamos trabalhando na geração dos corpora (Maio-Junho de 2004), o portal VISL não estava suportando tal formato para a análise de arquivos via upload.

na descrição sintática do PALAVRAS existem informações tanto sobre função sintática (ex: @SUBJ e @ACC) quanto sobre estrutura de constituinte (forma sintática). Essa estrutura de constituinte é representada através dos marcadores de dependência (< e >), que apontam para o núcleo da unidade sintática ao qual pertence, formando uma espécie de fronteira, que delimita todos os constituintes na sentença. Quando um núcleo não for o verbo principal, ele será marcado no ponto da dependência (ex: N para núcleo nominal, A para núcleo do adjunto). Se há uma etiqueta de função (ex: @SUBJ, @ADVL,@N<PRED), o marcador de dependência será ligado àquela etiqueta. Em casos nos quais a função é incluída no status do modificador, o marcador é deixado sem etiqueta (ex: @>N para modificadores pre-nominais).

A partir da análise das informações sintáticas relativas à delimitação do constituinte SN na anotação do PALAVRAS, foram criados dois conjuntos de etiquetas: um incluindo as que poderiam iniciar um SN e outro constando das que poderiam estar dentro de um SN. Com o uso desses conjuntos foi possível desenvolver um programa para identificar SNs em um texto anotado pelo *parser* de Bick. Tal programa, o *IdentifySNs.java*, recebe como entrada o arquivo gerado pela análise sintática e gera um outro arquivo, com o texto verticalizado, onde cada linha contém uma palavra e a sua etiqueta SN, separadas pelo caractere "_". A FIG. 6.4 mostra um exemplo do arquivo gerado por *IdentifySNs.java*.

O programa *IdentifySNs.java* não identifica SNs que contenham orações subordinadas adjetivas e/ou que contenham vírgula, ou seja, quando uma oração adjetiva ou uma vírgula são encontradas, o SN é quebrado naquele ponto. A FIG. 6.5 apresenta um pseudo código para o algoritmo de identificação de SNs implementado no programa *IdentifySNs.java*. Esse algoritmo não é eficaz para alguns casos, o que já era esperado, mas funcionou bem para a maior parte das situações. A maioria dos erros foi devida à inconsistência nos dados do corpus, a erros na própria análise sintática do PALAVRAS e a erros de identificação pelo próprio programa *IdentifySNs.java*. Os erros mais comuns provenientes da análise sintática foram os relacionados à coordenação sindética aditiva, pois em alguns casos o *parser* não consegue distinguir se a conjunção está coordenando termos simples (SNs) ou orações; e os erros relativos à ligação do sintagma preposicionado, visto que o PALAVRAS nem sempre reconhece se um sintagma preposicionado, que aparece imediatamente após um SN, faz parte ou não desse SN.

Segundo os resultados da identificação de SNs realizados por VIEIRA (2000), o PALAVRAS obteve uma taxa de erro de aproximadamente 10%, em termos de precisão (quantidade de SNs errados em relação ao total identificado). Na geração dos corpora

```
Cerca_O
de_O
650_I
homens_I
de_I
o_I
policiamento_I
de_I
choque_I
fizeram_O
ontem_O
uma_I
megablitz_I
em_O
as_I
ruas_I
de_I
o_I
centro_I
velho_I
de_I
São_I
Paulo_I
._O
```

FIG 6.4: Trecho de um arquivo gerado pelo programa *IdentifySNs.java*

de treino e teste, como não estamos identificando SNs que contenham vírgula nem que contenham orações subordinadas, acreditamos que essa taxa de erro deva ser um pouco menor; mas não foi feita uma análise para descobrir um percentual de erro exato no corpus.

O terceiro programa desenvolvido foi o *AttachSNtag.java*, que recebe como entrada dois arquivos: um do Mac-Morpho (ver FIG. 6.1) e o seu correspondente gerado pelo programa *IdentifySN.java* (ver FIG. 6.4). O objetivo de tal programa é acrescentar a etiqueta SN a cada palavra do arquivo do Mac-Morpho a partir da comparação com o arquivo gerado pelo programa *IdentifySN.java*. A saída desse programa é um arquivo semelhante aos arquivos do Mac-Morpho, apenas acrescido da etiqueta SN, como pode ser visualizado na FIG. 6.6. Esse programa também pode gerar como saída um arquivo HTML, no qual o texto aparece na horizontal, sem etiquetas e com os SNs destacados pelas cores azul e vermelho (o vermelho é usado para identificar um SN que aparece imediatamente após um outro).

```
ETIQUETAS_INI = Conjunto de etiquetas sintáticas ou
morfológicas+sintáticas que podem iniciar um SN.
ETIQUETAS_MID = Conjunto de etiquetas sintáticas ou
morfológicas+sintáticas que podem aparecer dentro de
um SN.
SN_INICIADO = FALSO
Para cada linha do arquivo Faça

  • Se (SN_INICIADO == FALSO) e (Etiqueta sintática
    ou morfológica+sintática da palavra atual ∈
    ETIQUETAS_INI)
      – Etiquetar palavra atual com I;
      – SN_INICIADO = VERDADEIRO
  • Senão Se (SN_INICIADO == VERDADEIRO) e (Etiqueta
    sintática ou morfológica+sintática da palavra atual
    ∈ ETIQUETAS_MID)
      – Etiquetar palavra atual com I;
  • Senão
      – Etiquetar palavra atual com O;
      – SN_INICIADO = FALSO;
```

FIG 6.5: Pseudocódigo do algoritmo implementado no programa *IdentifySNs.java*

O programa *AttachSNtag.java* também identifica quando parênteses/aspas devem ser incluídos em SNs, o que não é verificado em *IdentifySN.java*. Esta identificação é realizada com a verificação de que o abre parêntese/aspa vem imediatamente após uma palavra que tem etiqueta SN igual a I e todas as palavras dentro dos parênteses/aspas também têm etiquetas SN iguais a I.

Um quarto programa foi desenvolvido para formatar os arquivos do Mac-Morpho que passaram pela aplicação dos três programas citados anteriormente. Esse programa, *FormatCorpus.java*, realiza as seguintes tarefas: a) separar as sentenças por uma linha em branco (o algoritmo considera os caracteres [. ! ?] como delimitadores de sentenças), e b) retirar etiquetas complementares (DAT, HOR, TEL, etc.) e indicadoras de ênclises, contrações, disjunção e mesóclises, as quais foram consideras desnecessárias para o tipo de tarefa proposta.

Com o uso dos programas anteriormente descritos foram gerados 2 corpora de treino, cada um contendo textos de todos os cadernos jornalísticos do Mac-Morpho. Os corpora de treino têm tamanhos diferentes: 500 mil e 200 mil *tokens*(palavras e sinais de pontuação), sendo que o corpus de 200 mil (200k) está contido no de 500 mil (500k). Foram

```
Cerca_PREP_O
de_PREP_O
650_NUM_I
homens_N_I
de_PREP_I
o_ART_I
policiamento_N_I
de_PREP_I
choque_N_I
fizeram_V_O
ontem_ADV_O
uma_ART_I
megablitz_N_I
em_PREP_O
as_ART_I
ruas_N_I
de_PREP_I
o_ART_I
centro_N_I
velho_ADJ_I
de_PREP_I
São_NPROP_I
Paulo_NPROP_I
._._O
```

FIG 6.6: Trecho de um arquivo gerado pelo programa *AttachSNtag.java*

construídos corpora de tamanhos diferentes para que possa ser verificada a influência do tamanho do corpus de treino nos resultados do aprendizado.

Foi construído também um corpus para testes, contendo 50 mil tokens, com textos distintos dos contidos nos corpora de treino. O corpus de teste foi constituído por arquivos dos seguintes cadernos: Agricultura (arquivo ag94ab12.train.txt), Brasil (br94ab02.train.txt), Cotidiano (co94ab02.train.txt), Dinheiro (di94ab02.train.txt), Esportes (es94ab02.train.txt) e Ilustrada (il94ma01.train.txt).

6.3 CLASSIFICAÇÃO INICIAL

O aprendizado com TBL inicia-se com a atribuição de uma classificação básica aos itens do corpus. Tal classificação é dependente do problema e, como já citado, geralmente é realizada com o uso de estatísticas de co-ocorrências observada no corpus de treino. Entretanto, essa classificação inicial pode ser tão sofisticada quanto se desejar. Por

exemplo, pode-se utilizar um classificador estocástico para tal tarefa. Quanto melhor for a performance dessa classificação básica, mais rápido será o aprendizado, visto que existirão menos erros e menos regras candidatas serão geradas, e os resultados serão melhores, em maior ou menor grau, dependendo do problema.

No caso da identificação de SNs, a classificação inicial consiste em atribuir uma etiqueta SN a cada item do corpus. Nos experimentos realizados nesse trabalho, foram testados dois métodos:

- o primeiro foi atribuir a cada item do corpus a etiqueta SN que foi mais freqüentemente associada à etiqueta morfossintática daquele item no corpus de treino. Chamaremos esse método de não lexicalizado;

- o segundo método diferencia-se do primeiro apenas no caso das preposições, cuja etiquetagem inicial foi realizada tomando-se em consideração a unidade léxica e não a etiqueta morfossintática. Desse modo, cada preposição recebeu a etiqueta SN que foi mais freqüentemente usada para ela no corpus de treino. Esse tratamento diferenciado para as preposições foi elaborado porque verificou-se que a preposição "de", que na maioria dos casos pertence a um SN, estava sendo classificada inicialmente como fora dos SNs ("O"), provocando muitos erros na classificação básica. Chamaremos esse procedimento de método lexicalizado;

Em nossas experiências com o método lexicalizado, a classificação inicial provocou menos erros, o que fez com que menos regras candidatas fossem geradas, contribuindo para a redução do tempo de treinamento e para uma pequena melhora nos resultados obtidos com as regras aprendidas.

6.4 MOLDES DE REGRAS

Os *moldes de regras* determinam os tipos de expressões condicionais que comporão as regras. Dessa forma são eles que determinam o espaço de regras que podem ser geradas, e por isso são os elementos que provocam maior impacto no comportamento do aprendizado com TBL. Os moldes de regras devem codificar as informações lingüísticas do contexto que são importantes para a classificação de um item, informações estas que são dependentes do problema tratado.

Para a identificação de SNs básicos do inglês com TBL, Ramshaw & Marcus em (RAMSHAW, 1995) utilizaram um conjunto contendo 100 moldes de regras formados por termos atômicos que referenciam combinações de etiquetas SN com unidades léxicas

e combinações de etiquetas SN com etiquetas morfossintáticas. Os 10 padrões que fazem referências às unidades léxicas são mostrados na TAB. 6.2. Os mesmos 10 padrões também são utilizados para fazer referência às etiquetas morfossintáticas, mudando-se apenas o traço *word* pelo traço *tpos*, obtendo-se *tpos*_0, *tpos*_−1, *tpos*_1, etc. Esses 20 padrões que referenciam as unidades léxicas e etiquetas morfossintáticas são combinados com os 5 padrões que referenciam as etiquetas SN, mostrados na TAB. 6.3, formando o conjunto de 100 moldes de regras, que é apresentado no APÊNDICE 2. As regras geradas por esse conjunto de moldes possuem uma janela de contexto de no máximo sete intens.

TAB 6.2: Padrões que fazem referência a palavras

Padrão	Significado
$word\_0$	unidade léxica (traço *word*) atual
$word\_-1$	1^a *word* à esquerda
$word\_1$	1^a *word* à direita
$word\_-1,\ word\_0$	1^a *word* à esquerda e *word* atual
$word\_0,\ word\_1$	*word* atual e 1^a *word* à direita
$word\_-1,\ word\_1$	1^a *word* à esquerda e 1^a *word* à direita
$word\_-2,\ word\_-1$	duas primeiras *word* à esquerda
$word\_1,\ word\_2$	duas primeiras *word* à direita
$word[-1;-3]$	1^a, 2^a ou 3^a *word* à esquerda
$word[1;3]$	1^a, 2^a ou 3^a *word* à direita

TAB 6.3: Padrões que fazem referência a etiquetas SN

Padrão	Significado
tsn_0	etiqueta SN atual
$tsn\_0,\ tsn\_-1$	etiqueta SN atual e 1^a etiqueta SN à esquerda
$tsn\_0,\ tsn\_1$	etiqueta SN atual e 1^a etiqueta SN à direita
$tsn\_-1,\ tsn\_-2$	duas primeiras etiquetas SN à esquerda
$tsn\_1,\ tsn\_2$	duas primeiras etiquetas SN à direita

Para a identificação de SNs do português foram realizados experimentos com os seguintes conjuntos de moldes:

(C1) o conjunto de moldes de regras de Ramshaw & Marcus descritos anteriormente (ver ANEXO 2);

(C2) uma versão estendida de C1, onde todos os TAs que tinham intervalo [-1,-3] e [1,3], foram estendidos para [-1,-6] e [1,6], respectivamente; além disso foram incluídos mais 24 moldes que fazem referência aos traços *tsn* e *tpos* num contexto local e ao traço *word* num contexto de até 8 itens para a direita e para a esquerda (ver APÊNDICE 4);

(C3) um conjunto contendo os 80 moldes de regras de Ramshaw & Marcus que não possuem TAs do tipo *ftr[índice_inicial;índice_final]*, juntamente com 6 moldes, contendo TAs com restrição, desenvolvidos especificamente para classificação de preposições, como mostrado na FIG. 6.7. Esses 6 moldes de regras foram projetados para checarem alguns itens que possam contribuir com informações para a resolução da ligação de sintagmas preposicionados, usando uma janela de contexto de até vinte itens para a esquerda, na tentativa de ligar a preposição ao verbo que a precede ou à primeira unidade léxica que está classificada como fora de um SN (etiquetada com "O"). Esse conjunto de 86 moldes é mostrado no APÊNDICE 1. Outras combinações de TAs com restrição também foram experimentadas, mas esse conjunto de 6 moldes foi o que proporcionou melhores resultados;

(C4) um conjunto de moldes de regras derivado de C3, onde os 6 moldes que usam TAs com restrição foram remodelados usando apenas TAs tradicionais como mostrado na FIG. 6.8.

```
1.  tsn_-1  tpos[0;0](tpos=PREP)  word[-1;-20](tsn=O)
2.  tsn_-1  word[0;0](tpos=PREP)  word[-1;-20](tsn=O)
3.  tsn_-1  tsn_1  tpos[0;0](tpos=PREP)  word[-1;-20](tpos=V)
4.  tsn_-1  tsn_1  word[0;0](tpos=PREP)  word[-1;-20](tpos=V)
5.  tsn_-1  tsn_1  tsn_0  tpos[0;0](tpos=PREP)  word[-2;-20](tpos=V)
6.  tsn_-1  tsn_1  tsn_0  word[0;0](tpos=PREP)  word[-2;-20](tpos=V)
```

FIG 6.7: Moldes de regras que contêm TAs com restrição destinados à classificação de preposições

Os objetivos do uso dos conjuntos C1, C2, C3 e C4 foram:

- verificar o desempenho da identificação de SNs do português usando um contexto local (conjunto C1) e um contexto ampliado (conjunto C2) usando apenas TAs tradicionais (TAs do tipo (a) e (b), vide seção 4.2);

```
1.  tsn_-1  tpos_0  word[-1;-20]
2.  tsn_-1  word_0  word[-1;-20]
3.  tsn_-1  tsn_1  tpos_0  word[-1;-20]
4.  tsn_-1  tsn_1  word_0  word[-1;-20]
5.  tsn_-1  tsn_1  tsn_0  tpos_0  word[-2;-20]
6.  tsn_-1  tsn_1  tsn_0  word_0  word[-2;-20]
```

FIG 6.8: Moldes de regras que usam TAs tradicionais

- verificar se o uso dos TAs com restrição (conjunto C3), propostos nesse trabalho, realmente pode melhorar a classificação específica das preposições, com relação aos outros tipos de TAs, na identificação de SNs do português;
- verificar as vantagens do uso dos TAs com restrição em relação aos TAs tradicionais, com a comparação dos resultados obtidos com o uso dos conjuntos C3 e C4.

A busca por um método de melhorar a classificação das preposições deve-se principalmente ao fato de que, nos resultados da classificação inicial e até mesmo da aplicação das regras aprendidas com o conjunto C1, os erros de preposições representaram, em média, mais de 45% dos erros totais. Essa grande proporção deve-se ao fato já citado de que a classificação das preposições exige um contexto mais abrangente.

6.5 EXPERIMENTOS E RESULTADOS

Essa seção apresenta os resultados obtidos na identificação dos SNs do corpus de teste com a aplicação de regras aprendidas no treinamento realizado com o uso dos três corpora de treino descritos anteriormente. Os resultados serão reportados em termos de *precisão geral*(*accuracy*) da classificação item-a-item, precisão da identificação de SNs (percentual de SNs identificados que estavam corretos), abrangência da identificação de SNs (percentual de SNs existentes no corpus de teste e que foram identificados corretamente) e medida F ($F_{\beta=1}$), formalizados pelas EQs. 6.1, 6.2, 6.3 e 6.4, respectivamente. Esses são os tipos de métricas mais utilizadas para a avaliação da performance de ferramentas de identificação de SNs.

$$Precisão\ Geral = \frac{total\ de\ itens\ classif.\ corretamente}{total\ de\ itens} \tag{6.1}$$

$$Precisão = \frac{total\ de\ SNs\ identificados\ corretamente}{total\ de\ SNs\ identificados} \tag{6.2}$$

$$Abrangência = \frac{total\ de\ SNs\ identificados\ corretamente}{total\ de\ SNs\ existentes\ no\ corpus} \tag{6.3}$$

$$F_{\beta=1} = \frac{(\beta^2 + 1)\ *\ Precisão\ *\ Abrangência}{\beta^2\ *\ Precisão\ +\ Abrangência} \tag{6.4}$$

Para a verificação dos resultados, foi desenvolvido um programa que compara a classificação correta do corpus de teste com a classificação atribuída pela aplicação das regras.

6.5.1 RESULTADO DA CLASSIFICAÇÃO INICIAL

A TAB. 6.4 mostra a performance obtida pela aplicação dos dois métodos de classificação inicial ao corpus de teste. Tais procedimentos foram descritos na seção 6.3.

Os resultados da TAB. 6.4 mostram que realmente o método lexicalizado produz melhores resultados; apesar de que essa melhora da classificação inicial influenciou apenas na redução do tempo de treinamento. Essa redução foi de 23% no caso do treinamento com o conjunto de moldes de regras (c) e o corpus de 500k.

TAB 6.4: Resultado da classificação inicial para o corpus de treino

Medida	Método Lexicalizado	Método Não Lexicalizado
Precisão Geral	93,12%	88,7%
Abrangência	73,9%	62,6%
Precisão	63,8%	43,0%
$F_{\beta=1}$	68,5%	51,0%

Os resultados da aplicação, ao corpus de testes, dos conjuntos de regras aprendidos com os dois métodos de classificação inicial foram praticamente os mesmos. Houve apenas uma pequena melhora com o uso do método lexicalizado.

A maior complexidade do problema de identificação de SNs do português em comparação com a identificação de SNs básicos do inglês pode ser percebida pela comparação dos resultados da classificação inicial mostrados na TAB. 6.4 com os resultados reportados por RAMSHAW (1995). Usando o método não lexicalizado, RAMSHAW (1995) consegue 81,9% de abrangência e 78,2% de precisão, enquanto que no nosso corpus de teste, o mesmo método não lexicalizado gera resultados bem inferiores, 62,6% de abrangência e 43% de precisão.

Em todos os experimentos reportados nas subseções seguintes foi utilizado o método de classificação inicial lexicalizado.

6.5.2 RESULTADOS DOS EXPERIMENTOS COM DIFERENTES CONJUNTOS DE MOLDES DE REGRAS

Os resultados da aplicação ao corpus de teste, das regras aprendidas usando os dois corpora e os conjuntos de moldes C1, C2, C3 e C4, são mostrados nas tabelas 6.5 e 6.6. Nessas tabelas, a quinta linha indica o número de preposições que foram etiquetadas de forma errada no corpus de teste.

Nos experimentos com o corpus de treino de 200k, foram aprendidas regras com pontuação maior ou igual a 2. Já nos treinamentos com o corpus de 500k, o limite mínimo de pontuação para que uma regra pudesse ser aprendida foi 3.

TAB 6.5: Resultados da aplicação das regras aprendidas usando o corpus de 200k e os diferentes conjuntos de moldes de regras

Medida	Conjunto de Moldes C1	Conjunto de Moldes C2	Conjunto de Moldes C3	Conjunto de Moldes C4
Precisão Geral	96,85%	96,93%	**97,17%**	96,85%
Abrangência	82,9%	83,3%	**84,6%**	83,1%
Precisão	83,0%	83,4%	**85,2%**	82,7%
$F_{\beta=1}$	83,0%	83,4%	**84,9%**	82,9%
Erros de Preposições	703	716	**591**	716

TAB 6.6: Resultados da aplicação das regras aprendidas usando o corpus de 500k e os diferentes conjuntos de moldes de regras

Medida	Conjunto de Moldes C1	Conjunto de Moldes C2	Conjunto de Moldes C3	Conjunto de Moldes C4
Precisão Geral	97,20%	97,36%	**97,41%**	97,2%
Abrangência	84,6%	85,5%	**85,9%**	84,3%
Precisão	84,8%	85,6%	**86,6%**	84,7%
$F_{\beta=1}$	84,7%	85,6%	**86,2%**	84,5%
Erros de Preposições	635	592	**521**	632

Observando as TABs. 6.5 e 6.6, pode-se verificar que os melhores resultados, para ambos os corpora, foram conseguidos com o uso do conjunto de moldes de regras contendo TAs com restrição (conjunto C3). No treinamento com o corpus de 200k e o conjunto de

moldes C3, houve um aumento de 2% de $F_{\beta=1}$ em relação aos experimentos usando os conjuntos C1 e C4, e um aumento de 1,5% em relação ao experimento usando o conjunto C2.

Em termos de redução de erros de classificação de preposições, os resultados demonstraram que o maior poder de especificação dos moldes de regras contendo TAs com restrição realmente determinou a diminuição significativa desse tipo específico de erro. O fator decisivo para isso foi que os 6 moldes contendo TAs com restrição foram designados à criação de regras que se aplicavam apenas a preposições, usando um contexto ampliado, mas limitando-se a coleta de informações consideradas como relevantes para o problema de ligação de sintagmas preposicionados. No treinamento com o corpus de 200k, usando o conjunto de moldes C3, a redução dos erros de preposições foi de 16%, 17,5% e 17,5% em relação ao uso dos conjuntos de moldes C1, C2 e C4, respectivamente.

Com a aplicação das regras aprendidas com o corpus de 500k e o conjunto de moldes C3, a redução de erros de preposições foi de 18%, 12% e 17,6% em relação aos conjuntos de moldes C1, C2 e C4, respectivamente.

Outro fato que pode ser destacado é que, na aplicação das regras aprendidas com o uso do conjunto de moldes de Ramshaw & Marcus (conjunto C1), os erros de preposições representaram 45,75% dos erros totais de classificação. Com o uso do conjunto de moldes C3, esse percentual foi reduzido para 40,51% do total de erros.

Outras vantagens dos TAs com restrição puderam ser observadas nos experimentos usando os conjuntos C3 e C4, cujos resultados de tempo de treinamento e quantidade de regras geradas são mostrados nas TABs. 6.7 e 6.8. Nessas tabelas as quantidades de regras geradas são referentes aos 6 moldes de regras do conjunto C3 mostrados na FIG. 6.7 e das suas respectivas versões em C4 mostrados na FIG. 6.8. Os resultados confirmam as nossas afirmações sobre os problemas do uso dos TAs tradicionais para a construção de regras que envolvem itens com contexto variável e extenso – vide seção 5.1 – e comprovam que os TAs com restrição podem solucionar tais problemas. Usando TAs com restrição o tempo de treinamento foi reduzido em 7 vezes no caso do corpus de 200k e de 4,25 vezes no caso do corpus de 500k. A quantidade de regras geradas a partir dos moldes que continham TAs com restrição foi bem menor que quantidade de regras geradas a partir dos mesmos moldes que continham TAs sem restrição. Essa redução de tempo de treinamento e número de regras aprendidas deve-se ao fato de que os 6 TAs com restrição do conjunto C3 geram regras apenas para a correção da classificação de preposições – induzindo a um menor número de regras candidatas – enquanto que os

respectivos 6 TAs do conjunto C4 geram regras para correção de qualquer tipo de erro de classificação – induzindo a um grande número de regras candidatas.

TAB 6.7: Resultados da aplicação das regras aprendidas usando o corpus de 200k e os conjuntos de moldes de regras C3 e C4

Medida	Conjunto de Moldes C3	Conjunto de Moldes C4
Duração do treinamento	10,6 mins	73,7 mins
Regras geradas com os 6 TAs	95	663

TAB 6.8: Resultados da aplicação das regras aprendidas usando o corpus de 500k e os conjuntos de moldes de regras C3 e C4

Medida	Conjunto de Moldes C3	Conjunto de Moldes C4
Duração do treinamento	36 mins	153,22 mins
Regras geradas com os 6 TAs	76	458

6.5.3 CONTRIBUIÇÃO DA LEMATIZAÇÃO DOS VERBOS

Para verificar a real contribuição da lematização dos verbos para os resultados do aprendizado, foram realizados experimentos com os dois corpora de treino e o conjunto de moldes de regras C3, sendo que os verbos não foram lematizados. Os resultados são mostrados na TAB. 6.9.

TAB 6.9: Resultado da identificação de SNs usando TAs com restrição C3 – verbos não lematizados

Medida	Resultados Corpus 200k	Resultados Corpus 500k
Precisão Geral	97,14%	97,41%
Abrangência	84,5%	85,8%
Precisão	84,9%	86,5%
$F_{\beta=1}$	84,7%	86,1%
Erros de Preposições	593	544

Comparando-se as TABs. 6.5, 6.6 e 6.9, pode-se verificar que o uso da lematização dos verbos não trouxe uma contribuição muito significativa em termos de eficácia no resultado

da aplicação das regras aprendidas. Entretanto, foi constatado que a lematização trouxe algumas melhoras em relação ao tempo de treinamento e ao nível de abrangência do conjunto de regras aprendidas.

No treinamento com o corpus de 500k, sem usar a lematização dos verbos, foi verificado um aumento de 15% do tempo despendido no treinamento. Esse tempo, que foi de 51 minutos utilizando os verbos lematizados, passou para 60 minutos. Esse aumento do tempo de treinamento deve-se principalmente ao fato de que, sem a lematização, aumenta a variedade de unidades léxicas, gerando um maior número de variações de contextos, o que provoca a criação de um maior número de regras candidatas.

Em relação ao conjunto de regras obtidas, verificou-se que houve um aumento nas regras geradas pelos moldes que continham TAs com restrição, principalmente os que faziam referência direta a verbos (vide FIG. 6.7, moldes de regra 3 a 6). No caso do treinamento com o corpus de 500k o aumento foi de 18%, passando de um total de 76 regras, com o uso da lematização, para 90 regras, sem o uso da mesma.

Outro resultado favorável do uso da lematização é que o conjunto de regras aprendidas fica mais abrangente, visto que tanto na regra quanto no texto em que a regra vai ser aplicada, o verbo será observado numa única forma.

7 CONCLUSÕES E SUGESTÕES DE FUTUROS TRABALHOS

O principal objetivo desse trabalho, a construção de uma ferramenta de identificação de sintagmas nominais do português brasileiro com o uso de uma técnica de AM, foi atingido com êxito. Para a realização de tal tarefa foram analisadas algumas técnicas de aprendizado de máquina (AM) e, dentre elas, foi escolhida a técnica conhecida como Aprendizado Baseado em Transformações (TBL).

A estrutura sintática do SN do português foi estudada, o que permitiu a verificação de que, para a obtenção de um conjunto de SNs mais ricos em termos de conteúdo, seria necessária a identificação de SNs contendo os pós-modificadores adjetivos e sintagmas preposicionados. Tal abordagem é diferente da quem tem sido freqüentemente utilizada nos trabalhos que tratam da identificação de SNs do inglês, que normalmente usam o conceito de SNs básicos.

Como o SN que desejávamos identificar é mais complexo do que o SN básico, e por isso sua identificação inclui como uma sub-tarefa o problema da ligação de sintagmas preposicionados, foi fácil verificar que o mesmo conjunto de moldes de regras usado para o inglês (o conjunto proposto por RAMSHAW (1995)) não geraria bons resultados para o português. Logo, foi procurada uma alternativa para tentar minimizar o grande número de erros de preposições que foi verificado com o uso dos moldes de RAMSHAW (1995). A alternativa criada foi os Termos Atômicos com restrição.

Os TAs com restrição mostraram-se um mecanismo viável para tratar problemas de PLN que requerem contextos de tamanhos variáveis ou simplesmente extensos. Eles também podem ser usados para a geração de regras que corrigem erros específicos. Nos experimentos mostrados na seção 6.5, o conjunto de moldes de regras usando TAs com restrição foi mais eficiente do que os conjuntos de moldes que usavam apenas TAs tradicionais, quanto à precisão geral, à abrangência e à precisão da identificação de SNs do português, contribuindo também para uma redução significativa dos erros de classificação de preposições.

Nesse trabalho também foi proposta uma classificação inicial diferente da que normalmente é encontrada na literatura sobre identificação de SNs. Em tal classificação cada preposição recebe a etiqueta SN tendo como base a contagem das suas ocorrências individuais no corpus, enquanto que os outros itens recebem a etiqueta SN de acordo com

as ocorrências da sua classe de palavras como um todo. Com o uso desse procedimento, houve uma redução significativa no tempo de treinamento e uma pequena melhora nos resultados.

A lematização dos verbos no treinamento e na aplicação das regras também trouxe alguns benefícios para a eficácia das regras aprendidas. Mas seus principais benefícios foram a redução do tempo de treinamento e a ampliação da abrangência do conjunto de regras aprendidas.

Levando-se em consideração a complexidade do problema e os erros existentes no corpus de treino devido à derivação automática, acreditamos que a performance atual da identificação de SNs do português com o uso da ferramenta catTBL seja um bom resultado inicial (86,6% de precisão e 85,9% de abrangência no melhor caso). Acreditamos também que mais testes com outras combinações de moldes específicos para a correção de erros de preposições possam melhorar os resultados, visto que o número de erros de preposições ainda é alto.

Uma característica importante da ferramenta desenvolvida é que, por ser o TBL um algoritmo de aprendizado geral, ela pode ser utilizada não apenas para a identificação de SNs, mas também para qualquer outra tarefa de PLN que possa ser tratada como um problema de classificação e desde que exista um corpus de treino no formato requerido pelo catTBL.

Outra contribuição importante do trabalho foi o desenvolvimento de um corpus de treino de textos em português brasileiro com os SNs identificados. Tal corpus foi derivado automaticamente de um corpus já existente, mas que não continha informações sobre SNs. O corpus foi derivado com o uso de alguns algoritmos desenvolvidos e do analisador sintático PALAVRAS (BICK, 2000). Não foi cogitada a anotação manual do corpus porque não dispúnhamos de recursos humanos suficientes para tal.

Acreditamos também que, tão importante quanto o corpus de treino, são as ferramentas que foram desenvolvidas para a criação deste, e que foram descritas na seção 6.2.2. Tais ferramentas podem ser melhoradas ou reaproveitadas para outras tarefas.

Como trabalhos futuros sugerimos os seguintes itens de pesquisa:

- O tempo de treinamento despendido pela ferramenta catTBL está num patamar aceitável, mas acreditamos que seja possível reduzir ainda mais esse tempo a partir de uma revisão no código fonte da ferramenta de treinamento.

- Como citado anteriormente, é possível que mais testes, com outras combinações de templates específicos para a correção de erros de preposições, possam melhorar os

resultados da identificação de SNs do português, uma vez que o número de erros na classificação de preposições ainda é elevado.

- Melhorias no código fonte do programa lematizador e a ampliação para que ele também possa lematizar nomes (substantivos e adjetivos), pode trazer bons resultados, tanto em relação à redução de tempo de treinamento, quanto em relação à eficácia e abrangência das regras aprendidas.

- A correção manual do corpus de treino, cujos SNs foram marcados automaticamente, é um fator que certamente trará melhorias para a eficácia das regras aprendidas. Tal trabalho já começou a ser realizado pelas companheiras do CLIC (Centro de Lingüística Computacional da PUC-Rio).

- É conveniente treinar a ferramenta catTBL para a tarefa de etiquetagem morfossintática, utilizando o corpus Mac-Morpho para o treino. Dessa forma, todas as fases da identificação de SNs de um texto poderão ser realizadas pelo programa catTBL.

- Para tornar ainda mais clara a contribuição do presente trabalho com respeito ao aprendizado de SNs, seria importante usar outras técnicas de AM para identificar SNs do português, utilizando os mesmos corpora de treino e testes que foram desenvolvidos nesse trabalho, de forma que o desempenho de várias técnicas possam ser comparadas para esse problema.

- Acreditamos que o uso de TAs com restrição pode melhorar a performance de outras tarefas de PLN que têm sido abordadas com TBL. Por isso, uma continuação dessa pesquisa seria testar essa nova tecnologia para outros problemas de PLN, tais como análise sintática parcial, reconhecimento de entidades mencionadas, desambigüação do significado de palavras e etiquetagem de papéis semânticos.

8 REFERÊNCIAS BIBLIOGRÁFICAS

AIRES, R. V. X. **Implementação, adaptação, combinação e avaliação de etiquetadores para o português do brasil**. Dissertação de Mestrado, ICMC-USP, São Carlos - SP, setembro 2000.

BAEZA-YATES, R. e NETO, B. R. **Modern Information Retrieval**. Addison Wesley, 1999.

BARROS, F. A. e ROBIN, J. **Processamento de linguagem natural**. Tutorial apresentado na Jornada de Atualização em Informática no Congresso da Sociedade Brasileira de Computação, 1997.

BEAN, D. L. e RILOFF, E. **Corpus-based identification of non-anaphoric noun phrases**. Em *37th Annual Meeting of the Association for Computational Linguistics*, 1999.

BERGER, A. **Statistical machine learning for information retrieval**. Tese de Doutorado, School of Computer Science - Carnegie Mellon University, Pittsburgh, PA, April 2001.

BICK, E. **The Parsing System Palavras: Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework**. Tese de Doutorado, Aarhus University, 2000.

BIDERMAN, M. T. C. **Conceito lingüístico de palavra**. *Palavra*, 5:81–97, 1999.

BONFANTE, A. G. **Parsing Probabilístico para o Português do Brasil**. Tese de Doutorado, ICMC-USP, 2003.

BRILL, E. **A simple rule-based part of speech tagger**. Em *Proceedings of the Third Conference on Applied Natural Language Process*, Trento, Italy, 1992. Association for Computational Linguistics.

BRILL, E. **Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging**. *Computational Linguistics*, 21(4):543–565, 1995.

BRILL, E. **Recent Advances in Parsing Technology**, chapter Learning to Parse With Transformations. Kluwer Academic Publishers, 1996.

BRILL, E. e RESNIK, P. **A rule-based approach to prepositional phrase attachment disambiguation**. Em *Proceedings of COLING'94*, Kyoto, Japan, 1994.

CARDIE, C. e PIERCE, D. **Error-driven pruning of treebank grammars for base noun-phrase identification**. Em *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*, págs. 218–224, Ithaca-NY, 1998.

CARDIE, C. e WAGSTAFF, K. **Noun phrase coreference as clustering**. Em *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, págs. 82–89, Ithaca-NY, 1999.

CORSTON-OLIVER, S. e GAMON, M. **Combining decision trees and transformation-based learning to correct transferred linguistic representations**. Em *Proceedings of the Ninth Machine Tranlsation Summit*, págs. 55–62, New Orleans, USA, 2003. Association for Machine Translation in the Americas.

EVANS, D. A. e ZHAI, C. **Noun-phrase analysis in unrestricted text for information retrieval**. Em *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, págs. 17–24, Pittsburgh, 1996.

FLORIAN, R., HENDERSON, J. e NGAI, G. **Coaxing confidence from an old friend: Probabilistic classifications from transformation rule lists**. Em *Proceedings of Joint Sigdat Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*, Hong Kong University of Science and Technology, October 2000.

FLORIAN, R. e NGAI, G. **Fast Transformation-Based Learning Toolkit**. Johns Hopkins University, USA, September 2001.

FREITAS, M. C. **Elaboraçao automática de uma ontologia baseada em corpus da língua portuguesa**. Proposta de tese de doutorado (qualificação) apresentada no curso de Pós-Graduação em Letras da PUC-Rio, dezembro 2004.

HEPPLE, M. **Independence and commitment:assumptions for rapid training and execution of rule-based pos taggers**. Em *Proceedings of the 38th Annual Meeting of the ACL*, págs. 278–285, Hong Kong, 2000. Association for Computational Linguistics.

KOCH, I. V. e SILVA, M. C. P. S. **Lingüística aplicada ao português: sintaxe**. Cortez, São Paulo, 1985.

KOELING, R. **Chunking with maximum entropy models**. Em *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal, 2000.

KURAMOTO, H. **Uma abordagem alternativa para o tratamento e a recuperação de informação textual: os sintagmas nominais**. *Ciência da Informação*, 25 (2), 1995.

LOBATO, L. M. P. **Sintaxe Gerativa do português: da teoria padrão à teoria da regência e ligação**. Editora Vigília, Belo Horizonte, 1986.

MANGU, L. e BRILL, E. **Automatic rule acquisition for spelling correction**. Em *Proceedings of The Fourteenth International Conference on Machine Learning, ICML 97*. Morgan Kaufmann, 1997.

MANNING, C. D. e SCHÜTZE, H. **Foundations of statistical natural language processing**. The MIT Press, Cambridge, Massachusetts, 1999.

MARCHI, A. R. **Projeto lacio-web: Desafios na construção de um corpus de 1,1 milhão de palavras de textos jornalísticos em português do brasil**. Em *51º Seminário do Grupo de Estudos Lingüísticos do Estado de São Paulo*, São Paulo, Brasil, 2003.

MEGYESI, B. **Shallow parsing with pos taggers and linguistic features**. *Journal of Machine Learning Research*, 2:639–668, 2002.

MIORELLI, S. T. **Ed-cer: Extração do sintagma nominal em sentenças em português**. Dissertação de Mestrado, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2001.

MITCHELL, T. M. **Machine Learning**. McGraw-Hill, New York, March 1997. ISBN 0070428077.

MOLINA, A. e PLA, F. **Shallow parsing using specialized hmms**. *Journal of Machine Learning Research*, págs. 595–613, 2002.

MÀRQUEZ, L. **Machine learning and natural language processing**. Complementary documentation for the conference "Aprendizaje automático aplicado al procesamiento del lenguaje natural", Soria, july 2000.

NGAI, G. e FLORIAN, R. **Transformation-based learning in the fast lane**. Em *Proceedings of North American Chapter of the Association for Computational Linguistics*, págs. 40–47, June 2001.

NGAI, G. e YAROWSKY, D. **Rule writing or annotation: Cost-efficient resource usage for base noun phrase chunking**. Em *Proceedings of the 38th Annual Meeting of the ACL*, Hong Kong, 2000. Association for Computational Linguistics.

OLIVEIRA, C. e QUENTAL, V. S. D. B. **Aplicações do processamento automático de linguagem natural na recuperação de informações**. Em *Anais do III Congresso Internacional da ABRALIN*, págs. 949–955, Rio de Janeiro, 2003. UFRJ/Abralin.

OSBORNE, M. **Shallow parsing as part-of-speech tagging**. Em *Proceedings of CoNLL-2000 and LLL-2000*, págs. 145–147, 2000.

PERINI, M. A. **Gramática Descritiva do Português**. Editora Ática, São Paulo, 4 edition, 2003. ISBN 85-08-05550-1.

PINILLA, A., RIGONI, C. e INDIANI, M. T. **Site do projeto: Português - Ensino a Distândia (PEAD) do Departamento de Letras da UFRJ**. Disponível em http://www.pead.letras.ufrj.br/ [acessado em 15 set. 2004], 2004.

RAMSHAW, L. e MARCUS, M. **Text chunking using transformation-based learning**. Em YAROVSKY, D. E CHURCH, K., editores, *Proceedings of the Third Workshop on Very Large Corpora*, págs. 82–94, New Jersey, USA, 1995. Association for Computational Linguistics.

RATNAPARKHI, A. **A simple introduction to maximum entropy models for natural language processing**. Technical Report 08, University of Pennsylvania, Philadelphia, May 1997.

RATNAPARKHI, A. **Maximum Entropy Models for Natural Language Ambiguity Resolution**. Tese de Doutorado, University of Pennsylvania, 1998.

ROCHE, E. e SCHABES, Y. **Deterministic part-of-speech tagging with finite-state transducers**. *Computational Linguistics*, 21(2):227–253, 1995.

SAMUEL, K. **Lazy transformation-based learning**. Em *Proceedings of the 11th International Florida Artificial Intelligence Research Symposium Conference*, págs. 235–239, Florida, USA, 1998a.

SAMUEL, K., CARBERRY, S. e VIJAY-SHANKER, K. **Dialogue act tagging with transformation-based learning**. Em *Proceedings of COLING/ACL'98*, págs. 1150–1156, 1998b.

SANG, E. T. K. **Noun phrase recognition by system combination**. Em *Proceedings of ANLP-NAACL 2000*, Seattle, USA, 2000a. Morgan Kaufman.

SANG, E. T. K. **Text chunking by system combination**. Em *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal, 2000b.

SANG, E. T. K. **Memory-based shallow parsing**. *Journal of Machine Learning Research*, 2:559–594, 2002.

SATTA, G. e BRILL, E. **Efficient transformation-based parsing**. Em *Proceedings of the 34th conference on Association for Computational Linguistics*, págs. 255–262, California, USA, 1996. Association for Computational Linguistics.

SKUT, W. e BRANTS, T. **A maximum entropy partial parser for unrestricted text**. Em *Proceedings of the Sixth Workshop on Very Large Corpora*, Montréal, Quebéc, 1998.

SOON, W. M., LIM, D. C. Y. e NG, H. T. **A machine learning approach to coreference resolution of noun phrases**. *Association for Computational Linguistics*, 2001.

STAMATATOS, E., FAKOTAKIS, N. e KOKKINAKIS, G. **Automatic extraction of rules for sentence boundary disambiguation**. Em *Proceedings of the Workshop on Machine Learning in Human Language Technology, Advanced Course on Artificial Intelligence (ACAI '99)*, Chania, Greece, 1999.

UCHIMOTO, K., MA, Q., MURATA, M., OZAKU, H. e ISAHARA, H. **Named entity extraction based on a maximum entropy model and transformation rules**. Em *Proceedings of the ACL*, 2000.

VEENSTRA, J. **Fast np chunking using memory-based learning techniques**. Em VERDENIUS, F. E VAN DEN BROEK, W., editores, *Proceedings of BENELEARN-98*, Wageningen, The Netherlands, 1998.

VEENSTRA, J. e VAN DEN BOSCH, A. **Single-classifier memory-based phrase chunking**. Em *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal, 2000.

VIEIRA, R., CHISHMAN, R., GORZIZA, F., ROSSONI, R., ROSSI, D. e PINHEIRO, C. **Extração de sintagmas nominais para o processamento de co-referência**. Em *Anais do V Encontro para o Processamento Computacional do Português Escrito e Falado PROPOR*, págs. 19–22, Atibaia, Brazil, November 2000.

VOUTILAINEN, A. **Nptool, a detector of english noun phrases**. Em *Proceedings of the Workshop on Very Large Corpora*, págs. 48–57. Association for Computational Linguistics, 1993.

ZHAI, C. **Fast statistical parsing of noun phrases for document index**. Em *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Washington, DC, 1997. URL `citeseer.ist.psu.edu/article/zhai97fast.html`.

ZHOU, G. e SU, J. **Named entity recognition using an hmm-based chunk tagger**. Em *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, págs. 473–480, Philadelphia, July 2002.

9 APÊNDICES

9.1 APÊNDICE 1: CONJUNTO DE MOLDES DE REGRAS QUE OBTEVE OS MELHORES RESULTADOS NA IDENTIFICAÇÃO DE SNS DO PORTUGUÊS - CONJUNTO C3

```
tsn_-1  tpos[0;0](tpos=PREP)  word[-1;-20](tsn=O)
tsn_-1  word[0;0](tpos=PREP)  word[-1;-20](tsn=O)
tsn_-1  tsn_1  tpos[0;0](tpos=PREP)  word[-1;-20](tpos=V)
tsn_-1  tsn_1  word[0;0](tpos=PREP)  word[-1;-20](tpos=V)
tsn_-1  tsn_1  tsn_0  tpos[0;0](tpos=PREP)  word[-2;-20](tpos=V)
tsn_-1  tsn_1  tsn_0  word[0;0](tpos=PREP)  word[-2;-20](tpos=V)
tsn_0 tpos_-1
tsn_0 tpos_1
tsn_0 tpos_-1 tpos_0
tsn_0 tpos_0 tpos_1
tsn_0 tpos_-1 tpos_1
tsn_0 tpos_-2 tpos_-1
tsn_0 tpos_1 tpos_2
tsn_-1 tsn_0 tpos_0
tsn_-1 tsn_0 tpos_-1
tsn_-1 tsn_0 tpos_1
tsn_-1 tsn_0 tpos_-1 tpos_0
tsn_-1 tsn_0 tpos_0 tpos_1
tsn_-1 tsn_0 tpos_-1 tpos_1
tsn_-1 tsn_0 tpos_-2 tpos_-1
tsn_-1 tsn_0 tpos_1 tpos_2
tsn_0 tsn_1 tpos_0
tsn_0 tsn_1 tpos_-1
tsn_0 tsn_1 tpos_1
tsn_0 tsn_1 tpos_-1 tpos_0
tsn_0 tsn_1 tpos_0 tpos_1
tsn_0 tsn_1 tpos_-1 tpos_1
tsn_0 tsn_1 tpos_-2 tpos_-1
```

```
tsn_0 tsn_1 tpos_1 tpos_2
tsn_-2 tsn_-1 tpos_0
tsn_-2 tsn_-1 tpos_-1
tsn_-2 tsn_-1 tpos_1
tsn_-2 tsn_-1 tpos_-1 tpos_0
tsn_-2 tsn_-1 tpos_0 tpos_1
tsn_-2 tsn_-1 tpos_-1 tpos_1
tsn_-2 tsn_-1 tpos_-2 tpos_-1
tsn_-2 tsn_-1 tpos_1 tpos_2
tsn_1 tsn_2 tpos_0
tsn_1 tsn_2 tpos_-1
tsn_1 tsn_2 tpos_1
tsn_1 tsn_2 tpos_-1 tpos_0
tsn_1 tsn_2 tpos_0 tpos_1
tsn_1 tsn_2 tpos_-1 tpos_1
tsn_1 tsn_2 tpos_-2 tpos_-1
tsn_1 tsn_2 tpos_1 tpos_2
tsn_0 word_0
tsn_0 word_-1
tsn_0 word_1
tsn_0 word_-1 word_0
tsn_0 word_0 word_1
tsn_0 word_-1 word_1
tsn_0 word_-2 word_-1
tsn_0 word_1 word_2
tsn_-1 tsn_0 word_0
tsn_-1 tsn_0 word_-1
tsn_-1 tsn_0 word_1
tsn_-1 tsn_0 word_-1 word_0
tsn_-1 tsn_0 word_0 word_1
tsn_-1 tsn_0 word_-1 word_1
tsn_-1 tsn_0 word_-2 word_-1
tsn_-1 tsn_0 word_1 word_2
```

```
tsn_0 tsn_1 word_0
tsn_0 tsn_1 word_-1
tsn_0 tsn_1 word_1
tsn_0 tsn_1 word_-1 word_0
tsn_0 tsn_1 word_0 word_1
tsn_0 tsn_1 word_-1 word_1
tsn_0 tsn_1 word_-2 word_-1
tsn_0 tsn_1 word_1 word_2
tsn_-2 tsn_-1 word_0
tsn_-2 tsn_-1 word_-1
tsn_-2 tsn_-1 word_1
tsn_-2 tsn_-1 word_-1 word_0
tsn_-2 tsn_-1 word_0 word_1
tsn_-2 tsn_-1 word_-1 word_1
tsn_-2 tsn_-1 word_-2 word_-1
tsn_-2 tsn_-1 word_1 word_2
tsn_1 tsn_2 word_0
tsn_1 tsn_2 word_-1
tsn_1 tsn_2 word_1
tsn_1 tsn_2 word_-1 word_0
tsn_1 tsn_2 word_0 word_1
tsn_1 tsn_2 word_-1 word_1
tsn_1 tsn_2 word_-2 word_-1
tsn_1 tsn_2 word_1 word_2
```

9.2 APÊNDICE 2: LISTA DAS PRIMEIRAS 96 REGRAS APRENDIDAS PARA A IDENTIFICAÇÃO DE SNS DO PORTUGUÊS. USANDO O CORPUS DE TREINO DE 200K E O CONJUNTO DE MOLDES MOSTRADO NO APÊNDICE 1.

```
tsn_-1=O tsn_0=I tpos_0=PREP -> tsn=O
tsn_0=I tsn_1=O tpos_-1=V tpos_0=ADJ -> tsn=O
tsn_0=I word_0=se -> tsn=O
tsn_-1=I tsn_1=I tpos[0;0](tpos=PREP)=PREP word[-1;-20](tpos=V)=ZZZ ->tsn=I
tsn_-1=I tsn_0=O tpos_0=PCP -> tsn=I
tsn_0=I tsn_1=O tpos_0=PREP -> tsn=O
tsn_-1=I tsn_0=O tpos_0=KC tpos_1=N -> tsn=I
tsn_1=O tsn_2=I tpos_0=PCP tpos_1=PREP -> tsn=O
tsn_-1=I tsn_0=O tpos_0=ADV tpos_1=ADJ -> tsn=I
tsn_-2=O tsn_-1=I word_0=-> tsn=I
tsn_0=O tsn_1=I tpos_-1=NPROP tpos_0=KC -> tsn=I
tsn_0=I tsn_1=O tpos_0=( -> tsn=O
tsn_-1=I tsn_0=O tpos_1=NPROP tpos_2=-> tsn=I
tsn_-2=O tsn_-1=I tpos_-1=NPROP tpos_0=-> tsn=O
tsn_0=I tsn_1=O tpos_0=PROSUB tpos_1=V -> tsn=O
tsn_-1=I tsn_0=I tpos_-1=N tpos_0=ART -> tsn=B
tsn_1=I tsn_2=O tpos_0=( -> tsn=O
tsn_-1=O tsn_0=I tpos_0=) -> tsn=O
tsn_-1=O tsn_0=I tpos_-1=ADV tpos_0=ADJ -> tsn=O
tsn_-2=I tsn_-1=O tpos_0=NUM tpos_1=, -> tsn=O
tsn_0=I tpos_-1=V tpos_0=PROPESS -> tsn=O
tsn_-1=I tsn_1=I tsn_0=O tpos[0;0](tpos=PREP)=PREP word[-2;-20](tpos=V)=ser
-> tsn=I
tsn_-1=I tsn_0=I tpos_-1=PROSUB tpos_0=PREP -> tsn=O
tsn_-2=I tsn_-1=I tpos_0=KC tpos_1=ADJ -> tsn=I
tsn_0=I tsn_1=O word_-1=ser word_1=de -> tsn=O
tsn_-2=O tsn_-1=I word_0=) -> tsn=O
```

```
tsn_-2=I tsn_-1=I word_0=e word_1=de -> tsn=I
tsn_-1=I tsn_0=O word_-1=e word_0=de -> tsn=I
tsn_0=I tsn_1=I word_-1=) -> tsn=O
tsn_-1=O tsn_0=I tpos_0=ADJ tpos_1=PREP -> tsn=O
tsn_0=I tpos_-1=ZZZ tpos_0=PROSUB -> tsn=O
tsn_-1=O tsn_0=I tpos_0=( -> tsn=O
tsn_-2=O tsn_-1=O tpos_0=PREP -> tsn=O
tsn_-1=O tsn_0=I word_0=todos word_1=os -> tsn=O
tsn_-1=I tsn_0=O tpos_0="tpos_1=N -> tsn=I
tsn_-2=O tsn_-1=I tpos_-1=N tpos_0=-> tsn=O
tsn_-2=I tsn_-1=I word_0=em word_1=a -> tsn=I
tsn_0=I tsn_1=O word_0=me -> tsn=O
tsn_-2=O tsn_-1=I tpos_0=) -> tsn=O
tsn_-2=I tsn_-1=I word_0=em word_1=o -> tsn=I
tsn_-1=I tsn_0=I tpos_-1=NPROP tpos_0=ART -> tsn=B
tsn_-2=I tsn_-1=O word_-1=de word_0=-> tsn=I
tsn_1=I tsn_2=I tpos_0=PREP tpos_1=-> tsn=I
tsn_-1=O tsn_0=I tpos_0=ADJ tpos_1=ART -> tsn=O
tsn_0=I tsn_1=O tpos_-1=KS tpos_0=PROSUB -> tsn=O
tsn_-1=I tpos[0;0](tpos=PREP)=PREP word[-1;-20](tsn=O)=que -> tsn=I
tsn_0=I tsn_1=O tpos_-1=, tpos_0=ADJ -> tsn=O
tsn_0=I tsn_1=O tpos_-1=, tpos_0=PROSUB -> tsn=O
tsn_0=I word_0=nos -> tsn=O
tsn_-1=I tsn_0=O tpos_0="tpos_1=ADJ -> tsn=I
tsn_-2=I tsn_-1=I tpos_0=PREP tpos_1=ADV -> tsn=I
tsn_-2=I tsn_-1=I tpos_-1=PREP -> tsn=I
tsn_0=I tpos_-1=( tpos_0=PREP -> tsn=O
tsn_0=I tsn_1=O word_0=( -> tsn=O
tsn_1=I tsn_2=I tpos_-1=NUM tpos_0=KC -> tsn=I
tsn_0=I tsn_1=O word_0=lhe -> tsn=O
tsn_-1=I tsn_1=I word[0;0](tpos=PREP)=em word[-1;-20](tpos=V)=ter -> tsn=O
tsn_0=I tsn_1=I word_0=como -> tsn=O
tsn_0=O word_-1=acesso word_0=a -> tsn=I
tsn_-1=O tsn_0=I tpos_0=-> tsn=O
tsn_-2=O tsn_-1=O word_0=mais -> tsn=O
```

```
tsn_-2=I tsn_-1=I tpos_-2="tpos_-1=NPROP -> tsn=I
tsn_0=I word_1=vez -> tsn=O
tsn_-1=I tsn_0=O word_0=de word_1=as -> tsn=I
tsn_0=O tsn_1=I tpos_0=ART -> tsn=I
tsn_-2=I tsn_-1=I word_-1=dia word_0=anterior -> tsn=O
tsn_-2=I tsn_-1=I word_-1="word_0=de -> tsn=I
tsn_-1=O tsn_0=I tpos_0=ADJ tpos_1=, -> tsn=O
tsn_-1=I word[0;0](tpos=PREP)=a word[-1;-20](tsn=O)=de -> tsn=O
tsn_-1=I tsn_1=I word[0;0](tpos=PREP)=em word[-1;-20](tpos=V)=haver ->tsn=O
tsn_-1=I tsn_1=I word[0;0](tpos=PREP)=por word[-1;-20](tpos=V)=ZZZ -> tsn=O
tsn_0=I tpos_-1="tpos_0=) -> tsn=O
tsn_0=I tsn_1=I tpos_-1=N tpos_0=PROADJ -> tsn=B
tsn_1=O tsn_2=I word_-1=que word_0=se -> tsn=I
tsn_-1=I tsn_0=I word_-1=sede -> tsn=O
tsn_-1=I tsn_0=I tpos_-1=NPROP tpos_0=PROPESS -> tsn=O
tsn_-1=I tsn_0=O word_0=de word_1=os -> tsn=I
tsn_0=I tsn_1=I tpos_-1=) tpos_0=PREP -> tsn=O
tsn_-1=I tsn_0=B tpos_-2=N tpos_-1=N -> tsn=I
tsn_-1=I tsn_0=O tpos_0=ADV tpos_1=PCP -> tsn=I
tsn_0=I tsn_1=O word_0=não -> tsn=O
tsn_0=I word_-1=disposto word_0=em -> tsn=O
tsn_0=O tpos_-1=PREP tpos_0=PCP -> tsn=I
tsn_-1=I word[0;0](tpos=PREP)=em word[-1;-20](tsn=O)=contra -> tsn=O
tsn_-2=O tsn_-1=I tpos_-1=ADJ tpos_0=-> tsn=O
tsn_-2=I tsn_-1=I word_-1=combate word_0=a -> tsn=I
tsn_-2=O tsn_-1=I word_0=mesmo word_1=tempo -> tsn=O
tsn_-1=O tsn_0=I tpos_0=ADJ tpos_1=KC -> tsn=O
tsn_-1=I tsn_0=O word_0=ainda word_1=não -> tsn=I
tsn_1=O tsn_2=O word_-1=que word_0=o -> tsn=O
tsn_0=B tsn_1=I word_0=A -> tsn=I
tsn_-1=O tsn_0=I tpos_0=PROSUB tpos_1=VAUX -> tsn=O
tsn_-2=O tsn_-1=O tpos_0=ADJ tpos_1=-> tsn=O
tsn_-2=O tsn_-1=O tpos_0=ADJ tpos_1=.  -> tsn=O
tsn_-2=O tsn_-1=I word_0=em -> tsn=O
tsn_0=O tsn_1=I tpos_-1=ART tpos_0=PREP -> tsn=I
```

9.3 APÊNDICE 3: FRAGMENTO DO CORPUS DE TESTE COM OS SNS IDENTIFICADOS PELA APLICAÇÃO DAS REGRAS APRENDIDAS

Em [o dia 15], [Dia da Conservação do Solo], [o único fato] a festejar pode ser [a Convenção Internacional sobre Desertificação].
[A produção brasileira de pintos de corte] totalizou, em [fevereiro último], [166 milhões], [volume 6,79% superior a o registrado] em [fevereiro de 93], segundo [dados de a Associação Brasileira dos Produtores de Pinto de Corte (Apinco)] .
[A Apinco] destaca em [seu boletim mensal] que [o setor avícola] está otimista com [o atual programa de estabilização econômica].
"[A melhoria de o padrão aquisitivo] resulta em [maior demanda de carne de frango]".
[O secretário de a agricultura paulista], [Roberto Rodrigues], aprovou [o pacote de o trigo], anunciado em [o final de março] por [o governo federal].
[A nova alíquota para importação de o produto (17%)] garante [a competitividade de a triticultura nacional], disse [o secretário].
[agricultores de o Vale do Paranapanema], em [São Paulo], iniciaram ontem [o plantio de trigo duro].
[A variedade], para [a fabricação de massas], foi selecionada por [o Instituto Agronômico de Campinas].
[O governador Mário Pereira], de [o Paraná], e [o secretário de a agricultura José Carlos Tibúrcio] aproveitaram [o palanque de a Exposição Agropecuária de Londrina] para cobrar de [o ministro de a Agricultura], [Synval Guazzelli], [um novo modelo de política agrícola], capaz de estimular [investimentos] e [o crescimento de a produção].
[Antonio Ambrósio Amaro] é [o novo diretor de o Instituto de Economia Agrícola da Secretaria de Agricultura e Abastecimento de São Paulo].
[Ele] substitui [Pérsio de Carvalho Junqueira], que se aposentou após [34 anos de serviço pesquisador científico].
[As doenças] que estão atacando [as lavouras gaúchas de soja] podem ocasionar [a extinção de a cultura em o Estado].
[O alerta] é de [o pesquisador Nelson Neto], de [Cruz Alta], que participou em [a semana] passada de [o Forum Nacional da Soja] em [Porto Alegre].
Em [o decorrer de os últimos três anos], [a legislação tributária rural brasileira] vem passando por [sucessivas reformas].
Por [o conteúdo de cada nova lei] que entra em [vigor], nota se [um aperfeiçoamento de a malha jurídica] em [essa área] e também [o afunilamento de o cerco fiscal] em torno de [o contribuinte].
[Exemplo de isso] é [a nova lei de o ITR (nº 8.847/94)], que criou [o Cadastro Fiscal dos Imóveis Rurais da Secretaria da Receita Federal] - [Cafir].
[Essa fúria fiscalista] começou em [o governo Collor], com [a lei 8.022/90], que passou para [a competência de a o lançamento] e [a arrecadação de o ITR], que antes era de [o Incra].

9.4 APÊNDICE 4: CONJUNTO DE MOLDES DE REGRAS C2 – EXTENSÃO DO CONJUNTO DE MOLDES DE RAMSHAW & MARCUS

Para compor esse conjunto, foram utilizados todos os moldes de regras mostrados no ANEXO 2, sendo que todos os TAs que tinham intervalo [-1,-3] e [1,3] foram estendidos para [-1,-6] e [1,6], respectivamente. E foram incluídos os 24 moldes de regras mostrados no quadro abaixo.

```
tsn_-1 tsn_0 word_0 word_-3
tsn_-1 tsn_0 word_0 word_-4
tsn_-1 tsn_0 word_0 word_-5
tsn_-1 tsn_0 word_0 word_-6
tsn_-1 tsn_0 word_0 word_-7
tsn_-1 tsn_0 word_0 word_-8
tsn_-1 tsn_0 word_0 word_3
tsn_-1 tsn_0 word_0 word_4
tsn_-1 tsn_0 word_0 word_5
tsn_-1 tsn_0 word_0 word_6
tsn_-1 tsn_0 word_0 word_7
tsn_-1 tsn_0 word_0 word_8
tsn_-1 tsn_0 tpos_0 word_-3
tsn_-1 tsn_0 tpos_0 word_-4
tsn_-1 tsn_0 tpos_0 word_-5
tsn_-1 tsn_0 tpos_0 word_-6
tsn_-1 tsn_0 tpos_0 word_-7
tsn_-1 tsn_0 tpos_0 word_-8
tsn_-1 tsn_0 tpos_0 word_3
tsn_-1 tsn_0 tpos_0 word_4
tsn_-1 tsn_0 tpos_0 word_5
tsn_-1 tsn_0 tpos_0 word_6
tsn_-1 tsn_0 tpos_0 word_7
tsn_-1 tsn_0 tpos_0 word_8
```

9.5 APÊNDICE 5: TREINAMENTO COM A FERRAMENTA CATTBL

Para efetuar um treinamento com a ferramenta catTBL deve-se usar o seguinte comando:

java CatTBL_Trainer *<arq_conftraços> <arq_corpus> <separador> <arq_class_inicial> <arq_moldes> <pontuação_limite> > <arq_regras>*

onde:

- *<arq_conftraços>* é o nome do arquivo de configuração dos traços. Segue o formato mostrado na FIG. 9.1;
- *<arq_corpus>* é o nome do arquivo do corpus de treino. Tal arquivo deve seguir o formato apresentado na FIG. 9.2. Pode ser usado também o nome de um diretório, cujos arquivos serão agrupados para formarem o corpus de treino;
- *<separador>* deve ser o número 0 (zero) ou o 1(um), onde: 0 indica que os traços de um item no corpus estão separados pelo caracter "_" (*underscore*), e 1 indica que o separador é o espaço em branco;
- *<arq_class_inicial>* é o arquivo que contém informações necessárias para a classificação inicial. Segue o formato mostrado na FIG. 9.3;
- *<arq_moldes>* é o nome arquivo que contém os moldes que serão usados para a criação das expressões condicionais das regras. O formato desse arquivo é mostrado na FIG. 9.4;
- *<pontuação_limite>* deve ser número inteiro positivo que indica o critério de parada do treinamento. Serão aprendidas apenas as regras que tenham pontuação (valor da função objetivo f) maior que esse número. Quando não existirem mais regras que atendam a esse critério, o processo de aprendizado terminará;
- *<arq_regras>* é nome do arquivo de regras que será gerado. Segue o formato apresentado na FIG. 9.5.

A seguir, os formatos dos arquivos de entrada e saída usados pelo programa catTBL são descritos brevemente.

ARQUIVO DE CONFIGURAÇÃO DOS TRAÇOS

O *arquivo de configuração dos traços* contém uma lista dos identificadores (nomes) dos traços, onde a posição em que eles aparecem na lista deve coincidir com a posição em que eles se encontram nos itens do corpus. Algumas funções que os traços desempenham são indicadas por símbolos especiais prefixados aos seus nomes. Esses símbolos especiais, bem como o significado do seu uso são os seguintes:

* : indica que o traço será utilizado para o teste na escolha da classificação inicial. A ausência de um traço com prefixo (*) indica que não será usado o classificador inicial, e nesse caso, os itens do corpus necessitam ter uma classificação prévia;

\+ : indica o traço que contém a classificação verdadeira para o item (usado apenas no treinamento);

\- : indica o traço que está sendo classificado, ou seja, é o traço que receberá a classificação inicial (caso necessário), e que recebe o valor que aparece no lado direito de uma regra que é aplicada.

Um exemplo de arquivo de configuração dos traços para o corpus de treino utilizado nesse trabalho, para a identificação de SNs, pode ser visto na FIG. 9.1.

```
word *tpos +truthTSN -tsn
```

FIG 9.1: Exemplo de um arquivo de configuração dos traços

ARQUIVO DO CORPUS

No arquivo do corpus interpretado pela ferramenta catTBL cada linha deve representar um item, cujos *traços* devem estar separados por caracteres "_" ou por um espaço em branco. Outro detalhe importante é que as linhas em brancos são consideradas como os divisores das sentenças. A FIG. 9.2 mostra um exemplo de um arquivo do corpus, que inclusive poderia ser usado com as configurações de traços do arquivo mostrado na FIG. 9.1.

ARQUIVO PARA A CLASSIFICAÇÃO INICIAL

Esse arquivo descreve a classificação inicial que deve ser atribuída aos itens do corpus. Essa classificação é atribuída de acordo com o valor de um traço que deve ser testado no item. No caso da identificação de SNs geralmente usa-se a etiqueta morfossintática

```
Ele_PROPESS_I
poderá_VAUX_O
remanejar_V_O
seus_PROADJ_I
dados_N_I
para_PREP_O
enquadramento_N_I
em_PREP_I
alíquotas_N_I
reduzidas_PCP_I
._._O

O_ART_I
Cafir_NPROP_I
fechou_V_O
também_PDEN_O
mais_PROADJ_O
uma_NUM_I
porteira_N_I
de_PREP_I
sonegação_N_I
fiscal_ADJ_I
em_PREP_O
o_ART_I
setor_N_I
fundiário_ADJ_I
._._O
```

FIG 9.2: Trecho de um arquivo de corpus aceito pela ferramenta catTBL

(traço *tpos*) como o traço de teste, e atribui-se para cada item do corpus a etiqueta de SN (I, O ou B) que foi mais freqüentemente associada à sua etiqueta morfossintática no corpus de treino.

Um exemplo de arquivo para a classificação inicial pode ser visto na FIG. 9.3. Cada linha do arquivo deve seguir o formato:

$$\langle valor\_traço\_teste \rangle \;\; \langle classif \rangle$$

onde:$\langle valor\_traço\_teste \rangle$ é o valor do traço que será testado nos itens, e $\langle classif \rangle$ é classificação inicial que deve ser atribuída ao item cujo traço de teste tem valor igual a $\langle valor\_traço\_teste \rangle$.

Nos experimentos reportados nesse trabalho, o arquivo usado na classificação inicial foi gerado por um programa que desenvolvemos. Tal programa recebe como entrada o(s)

arquivo(s) do corpus de treino e efetua as estatísticas de co-ocorrência dos valores dos traços de teste e do que está sendo classificado.

```
PCP I
V O
VAUX O
N I
...  O
PROSUB I
? O
IN I
; O
NPROP I
ADV O
NUM I
ADJ I
PDEN O
CUR I
KS O
PROPESS I
KC I
ADV-KS O
PRO-KS O
```

FIG 9.3: Exemplo de um arquivo usado para a classificação inicial

ARQUIVO DE MOLDES DE REGRAS

Como foi mostrado na seção 5.1, o formato de um molde de regra é o seguinte:

$$\langle t_1 \rangle \ \langle t_2 \rangle \ \langle t_3 \rangle \ \ldots \ \langle t_n \rangle$$

onde cada $\langle t_i \rangle$ é um termo atômico.

No arquivo de moldes de regras cada linha representará um molde diferente, e o formato dos TAs seguem os mesmos padrões mostrados na seção 5.1, e que são relembrados a seguir:

- *traço_indice*
- *traço[ind_inicial;ind_final]*
- *traçoX[ind_inicial;ind_final](traçoY=valY)*

```
tsn_0 tpos_0
tsn_0 tpos_-1
tsn_0 tsn_-1 tpos_0 tpos_1
tsn_0 tsn_1 tpos[1;3]
tsn_1 tsn_2 tpos[-1;-3]
tsn_0 word_0
tsn_0 word_-1
tsn_0 tsn_-1 word_0 word_1
tsn_0 tsn_1 word[1;3]
tsn_1 tsn_2 word[-1;-3]
tsn_-1 tsn_0 word[0,0](tpos=PREP) word[-2,-10](tpos=V)
tsn_0 word[0,0](tpos=PREP) word[-1,-7](tpos=N)
```

FIG 9.4: Exemplo de um arquivo de moldes de regras

A figura FIG. 9.4 mostra um exemplo de um arquivo de molde.

ARQUIVO DE REGRAS APRENDIDAS

O arquivo de regras aprendidas, que é gerado pela ferramenta de treinamento, tem o formato que é mostrado na FIG. 9.5, onde cada linha representa uma regra diferente, e cada regra segue o formato mostrado na seção 5.1:

$$\langle t_1 \rangle = val_1 \;\; \langle t_2 \rangle = val_2 \;\; \langle t_3 \rangle = val_3 \;\; ... \;\; \langle t_n \rangle = val_n \;\; \rightarrow \;\; \langle traço \rangle = val$$

```
tsn_-1=O tsn_0=I tpos_0=PREP -> tsn=O
tsn_0=I tsn_1=O tpos_-1=V tpos_0=ADJ -> tsn=O
tsn_0=I word_0=se -> tsn=O
tsn_-1=I tsn_0=O tpos_0=PCP -> tsn=I
tsn_0=I tsn_1=O tpos_0=PREP -> tsn=O
tsn_-1=I tsn_0=O tpos_0=KC tpos_1=N -> tsn=I
tsn_-2=O tsn_-1=I word_0=-> tsn=I
tsn_-1=I tsn_0=O tpos_0=ADV tpos_1=ADJ -> tsn=I
tsn_1=O tsn_2=I tpos_0=PCP tpos_1=PREP -> tsn=O
tsn_-1=I tpos_[0;0](tpos=PREP)=PREP stem_[-1;-20](tpos=V)=ZZZ -> tsn=I
```

FIG 9.5: Formato do arquivo de regras aprendidas

9.6 APÊNDICE 6: APLICAÇÃO DAS REGRAS APRENDIDAS COM A FERRAMENTA CATTBL

O formato do arquivo de regras aceito segue o mesmo formato do arquivo gerado pelo treinador. Esse padrão de arquivo pode ser visualizado na FIG. 9.5.

Para executar a ferramenta de aplicação de regras o seguinte comando deve ser usado:

java CatTBL_Apply <*arq_conftraços*> <*arq_corpus*> <*separador*> <*arq_class_inicial*> <*arq_regras*> > <*arquivo_de_saída*>

onde:

- <*arq_conftraços*> é nome do arquivo de configuração dos traços (vide FIG. 9.1);
- <*arq_corpus*> é nome do arquivo do corpus/texto a ser classificado e que deve seguir o formato mostrado na FIG. 9.2;
- <*separador*> deve ser o número 0 (zero) ou o 1(um), onde: 0 indica que os traços de um item no corpus estão separados pelo caracter "_" (*underscore*), e 1 indica que o separador é o espaço em branco;
- <*arq_class_inicial*> é nome do arquivo com informações para a classificação inicial (vide FIG. 9.3);
- <*arq_regras*> é nome do arquivo que contém a lista das regras que devem ser aplicadas. Tal arquivo deve seguir o formato mostrado na FIG. 9.5;
- <*arquivo_de_saída*> é nome do arquivo que será gerado pela aplicação das regras, o qual conterá os mesmos dados do corpus e ainda um novo traço, contendo a classificação atribuída aos itens.

10 ANEXOS

10.1 ANEXO 1: CONJUNTO DE ETIQUETAS MORFOSSINTÁTICAS UTILIZADO PELO CORPUS MAC-MORPHO

O corpus Mac-Morpho utiliza o conjunto de etiquetas morfossintáticas conhecido como LW Tagset. A seguir é dada uma relação das etiquetas que compõe LW Tagset, com os respectivos significados.

CLASSE GRAMATICAL	ETIQUETA
Adjetivo	ADJ
Advérbio	ADV
Advérbio conectivo subordinativo	ADV-KS
Advérbio relativo subordinativo	ADV-KS-REL
Artigo (definido ou indefinido)	ART
Conjunção coordenativa	KC
Conjunção subordinativa	KS
Interjeição	IN
Nome	N
Nome próprio	NPROP
Numeral	NUM
Particípio	PCP
Palavra denotativa	PDEN
Preposição	PREP
Pronome adjetivo	PROADJ
Pronome conectivo subordinativo	PRO-KS
Pronome pessoal	PROPESS
Pronome relativo conectivo subordinativo	PRO-KS-REL
Pronome substantivo	PROSUB
Verbo	V
Verbo auxiliar	VAUX
Símbolo de moeda corrente	CUR

ETIQUETAS COMPLEMENTARES DO LW TAGSET	
Estrangeirismos	|EST
Apostos	|AP
Dados	|DAD
Números de Telefone	|TEL
Datas	|DAT
Horas	|HOR
Disjunção	|[|]
Contrações e ênclises	|+
Mesóclises	|!

10.2 ANEXO 2: MOLDES DE REGRAS USADOS POR RAMSHAW & MARCUS PARA A IDENTIFICAÇÃO DE SNS BÁSICOS DO INGLÊS

```
tsn_0 tpos_0
tsn_0 tpos_-1
tsn_0 tpos_1
tsn_0 tpos_-1 tpos_0
tsn_0 tpos_0 tpos_1
tsn_0 tpos_-1 tpos_1
tsn_0 tpos_-2 tpos_-1
tsn_0 tpos_1 tpos_2
tsn_0 tpos[-1;-3]
tsn_0 tpos[1;3]
tsn_-1 tsn_0 tpos_0
tsn_-1 tsn_0 tpos_-1
tsn_-1 tsn_0 tpos_1
tsn_-1 tsn_0 tpos_-1 tpos_0
tsn_-1 tsn_0 tpos_0 tpos_1
tsn_-1 tsn_0 tpos_-1 tpos_1
tsn_-1 tsn_0 tpos_-2 tpos_-1
tsn_-1 tsn_0 tpos_1 tpos_2
tsn_-1 tsn_0 tpos[-1;-3]
tsn_-1 tsn_0 tpos[1;3]
tsn_0 tsn_1 tpos_0
tsn_0 tsn_1 tpos_-1
tsn_0 tsn_1 tpos_1
tsn_0 tsn_1 tpos_-1 tpos_0
tsn_0 tsn_1 tpos_0 tpos_1
tsn_0 tsn_1 tpos_-1 tpos_1
tsn_0 tsn_1 tpos_-2 tpos_-1
tsn_0 tsn_1 tpos_1 tpos_2
tsn_0 tsn_1 tpos[-1;-3]
tsn_0 tsn_1 tpos[1;3]
```

```
tsn_-2 tsn_-1 tpos_0
tsn_-2 tsn_-1 tpos_-1
tsn_-2 tsn_-1 tpos_1
tsn_-2 tsn_-1 tpos_-1 tpos_0
tsn_-2 tsn_-1 tpos_0 tpos_1
tsn_-2 tsn_-1 tpos_-1 tpos_1
tsn_-2 tsn_-1 tpos_-2 tpos_-1
tsn_-2 tsn_-1 tpos_1 tpos_2
tsn_-2 tsn_-1 tpos[-1;-3]
tsn_-2 tsn_-1 tpos[1;3]
tsn_1 tsn_2 tpos_0
tsn_1 tsn_2 tpos_-1
tsn_1 tsn_2 tpos_1
tsn_1 tsn_2 tpos_-1 tpos_0
tsn_1 tsn_2 tpos_0 tpos_1
tsn_1 tsn_2 tpos_-1 tpos_1
tsn_1 tsn_2 tpos_-2 tpos_-1
tsn_1 tsn_2 tpos_1 tpos_2
tsn_1 tsn_2 tpos[-1;-3]
tsn_1 tsn_2 tpos[1;3]
tsn_0 word_0
tsn_0 word_-1
tsn_0 word_1
tsn_0 word_-1 word_0
tsn_0 word_0 word_1
tsn_0 word_-1 word_1
tsn_0 word_-2 word_-1
tsn_0 word_1 word_2
tsn_0 word[-1;-3]
tsn_0 word[1;3]
tsn_-1 tsn_0 word_0
tsn_-1 tsn_0 word_-1
tsn_-1 tsn_0 word_1
tsn_-1 tsn_0 word_-1 word_0
tsn_-1 tsn_0 word_0 word_1
```

```
tsn_-1 tsn_0 word_-1 word_1
tsn_-1 tsn_0 word_-2 word_-1
tsn_-1 tsn_0 word_1 word_2
tsn_-1 tsn_0 word[-1;-3]
tsn_-1 tsn_0 word[1;3]
tsn_0 tsn_1 word_0
tsn_0 tsn_1 word_-1
tsn_0 tsn_1 word_1
tsn_0 tsn_1 word_-1 word_0
tsn_0 tsn_1 word_0 word_1
tsn_0 tsn_1 word_-1 word_1
tsn_0 tsn_1 word_-2 word_-1
tsn_0 tsn_1 word_1 word_2
tsn_0 tsn_1 word[-1;-3]
tsn_0 tsn_1 word[1;3]
tsn_-2 tsn_-1 word_0
tsn_-2 tsn_-1 word_-1
tsn_-2 tsn_-1 word_1
tsn_-2 tsn_-1 word_-1 word_0
tsn_-2 tsn_-1 word_0 word_1
tsn_-2 tsn_-1 word_-1 word_1
tsn_-2 tsn_-1 word_-2 word_-1
tsn_-2 tsn_-1 word_1 word_2
tsn_-2 tsn_-1 word[-1;-3]
tsn_-2 tsn_-1 word[1;3]
tsn_1 tsn_2 word_0
tsn_1 tsn_2 word_-1
tsn_1 tsn_2 word_1
tsn_1 tsn_2 word_-1 word_0
tsn_1 tsn_2 word_0 word_1
tsn_1 tsn_2 word_-1 word_1
tsn_1 tsn_2 word_-2 word_-1
tsn_1 tsn_2 word_1 word_2
tsn_1 tsn_2 word[-1;-3]
tsn_1 tsn_2 word[1;3]
```

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo