Instituto de Ciências Matemáticas e de Computação

ISSN - 0103-2569

Extração de Informação de Referências Bibliográficas usando POS-*tagging*

Alberto Cáceres Álvarez

Alneu de Andrade Lopes

Nº 281

RELATÓRIOS TÉCNICOS DO ICMC

São Carlos

Setembro/2006

Extração de Informação de Referências Bibliográficas usando POS-*tagging*

Alberto Cáceres Álvarez
Alneu de Andrade Lopes

Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação
C.P. 668, 13560-970 - São Carlos, SP - Brasil
e-mail: {beto, alneu}@icmc.usp.br

Resumo:

As técnicas relacionadas à área de extração de informações atuam sobre um conjunto de dados não estruturados e visam localizar informações relevantes em um documento ou coleção de documentos, extraí-las e estruturá-las a fim de facilitar sua manipulação e análise. O objetivo específico deste projeto é induzir, de forma automática, um conjunto de regras para extração de informações das referências bibliográficas de artigos científicos. A proposta para extrair automaticamente informações das referências, baseia-se no mapeamento do problema de *part-of-speech* (POS) *tagging* ao problema de Extração de Informação (EI). O mapeamento para a extração de informações de um texto consiste em, inicialmente, etiquetar todos os termos do texto, selecionando alguma etiqueta de um conjunto pré-definido de etiquetas e, posteriormente, combinar e extrair as informações de acordo com as etiquetas dos termos. Neste trabalho são apresentados os resultados da utilização desta abordagem para o problema da extração de informações de referências bibliográficas.

Palavras Chaves: Aprendizado de Máquina, Extração de Informação, Processamento de Língua Natural.

---

setembro 2006

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

1 Introdução

Extração de Informação (EI) consiste na tarefa de encontrar informações específicas a partir de grandes volumes de documentos. Tais documentos podem apresentar algum nível de estruturação na apresentação das informações, como também podem ser totalmente livres. O objetivo da pesquisa em EI é construir sistemas que encontrem e combinem informações relevantes enquanto ignoram informações insignificantes e irrelevantes (Cowie & Lehnert 1996; Ma, Uchimoto, Murata, & Isahara 1999).

Técnicas de Processamento de Língua Natural (PLN) têm sido amplamente utilizadas no processo de extração de informações de documentos semi-estruturados e livres (Cowie & Lehnert 1996; Soderland 1999). O objetivo do uso dessas técnicas de PLN no contexto de EI é tentar compreender textos em alguma língua natural, a fim de encontrar informações relevantes a serem extraídas. Sistemas de Extração baseados em PLN têm sido definidos para diferentes domínios, contando com etapas de processamento comuns aos sistemas de PLN em geral, e mais alguns módulos específicos para Extração de Informação (Rajman & Besançon 1997).

A extração de informações a partir de artigos científicos depara-se com diversos desafios relacionados com a identificação, no texto, de informações, tais como: título, ano, autores, palavras chaves, resumo e elementos de referências bibliográficas. As referências, normalmente, possuem os seguintes elementos: autores, título, publicação (periódicos, revistas, etc.) ou evento (simpósios, congressos, etc.), editora, páginas e ano.

A abordagem para extrair automaticamente informações de artigos científicos, proposta neste trabalho, consiste em valer-se de uma técnica baseada em etiquetagem para a tarefa de extração. Por exemplo, um possível mapeamento do problema de *part-of-speech* (POS) *tagging* ao problema de extração de informações de referências bibliográficas, resume-se em etiquetar todos os termos das referências, selecionando alguma etiqueta de uma lista de possíveis etiquetas, esta lista poderia conter as seguintes etiquetas: AUTOR, TÍTULO, REVISTA, EVENTO, LOCAL, PÁGINAS e ANO. Ao final, as informações de etiquetas correspondentes são combinadas e extraídas. Dentro desta abordagem, um objetivo específico deste trabalho é descrever em detalhes todos os procedimentos necessários à realização dos experimentos a fim de avaliar o comportamento do etiquetador TBL de Eric Brill (Brill 1994; Brill 1995; Brill 1997), frente à tarefa de etiquetar referências bibliográficas de artigos científicos. Ressalta-se, entretanto, que a técnica proposta não depende unicamente do etiquetador TBL. Outros etiquetadores poderiam ser usados. O corpus utilizado nos experimentos é composto por uma pequena parcela etiquetada manualmente e por outra automaticamente etiquetada utilizando informações contidas em arquivos BibTEX.

Este trabalho faz parte de um projeto, uma ferramenta chamada FIP (Ferramenta Inteligente de Apoio à Pesquisa), que tem como objetivo a recuperação automática de artigos científicos sobre áreas pré-selecionadas na Web; análise dos artigos por meio de técnicas de Aprendizado de Máquina e Mineração de Dados; a construção de um mapa

representativo do conhecimento das sub-áreas, tópicos e temas de pesquisas; importância relativa dos artigos e autores, entre outros (Brasil & Lopes 2004; Melo & Lopes 2004a; Melo & Lopes 2004b; Melo & Lopes 2005).

Dentro do contexto dessa ferramenta (FIP), é de suma importância a tarefa de extração de informações de artigos científicos recuperados, de tal forma que se possa estruturar e indexar a base de dados (repositório de artigos científicos) da ferramenta.

Este trabalho está organizado da seguinte forma: na Seção 2 são apresentados uma introdução a área de Extração de Informação descrevendo os principais conceitos e métricas de avaliação, trabalhos relacionados e a proposta de extração baseada em POS-*tagging*; na Seção 3 é apresentado o problema da etiquetagem automática de textos, além de descrever o funcionamento do etiquetador TBL; na Seção 4 são apresentados o processo de etiquetagem do corpus, o pré-processamento realizado, algumas decisões de projeto tomadas e os resultados experimentais obtidos; na Seção 5 são apresentadas as considerações finais.

2 Extração de Informação

2.1 Objetivo da Extração de Informação

A área de Extração de Informação visa localizar e extrair, de forma automática, informações relevantes em um documento ou coleção de documentos expressos em língua natural e estruturar tais informações para os padrões de saída, por exemplo em Banco de Dados ou textos em língua natural[1], a fim de facilitar sua manipulação e análise (Grishman 1997). Sistemas de EI não realizam o entendimento completo dos documentos, pelo contrário, tais sistemas analisam partes dos documentos que possuam informações relevantes.

O objetivo da pesquisa em EI é construir sistemas que encontrem e combinem informações relevantes enquanto ignoram informações insignificantes e irrelevantes (Cowie & Lehnert 1996). É importante ressaltar que a informação extraída é determinada por um conjunto de padrões ou regras de extração específicas ao seu domínio. A definição de tais padrões pode ser feita manualmente, por algum especialista, ou com diferentes graus de automação.

Extração de Informação não deve ser confundida com a desenvolvida área de Recuperação de Informação (RI), a qual seleciona, de uma grande coleção, um subconjunto de documentos relevantes baseados em uma consulta do usuário. O contraste entre os objetivos dos sistemas de EI e RI pode ser declarado da seguinte forma: RI recupera documentos relevantes de uma coleção, enquanto EI extrai informações relevantes dos documentos. Portanto, as duas técnicas são complementares e quando combinadas podem produzir ferramentas interessantes para processamento de texto (Gaizauskas & Wilks

[1]essa forma de estruturação pode envolver, por exemplo, trabalhos relacionados com a sumarização automática de documentos (Lin & Hovy 2003)

1998).

Os textos ou documentos dos quais são extraídas as informações de interesse podem apresentar algum nível de estruturação na apresentação dos dados, como também podem ser totalmente livres. O tipo de texto de onde é feita a extração tem grande influência sobre a escolha da técnica a ser utilizada na construção de sistemas de EI, pois tal técnica pode se basear apenas na estrutura do texto, quando existente. A seguir é dada uma breve descrição dos possíveis tipos de textos.

Estruturado: um texto é considerado estruturado quando apresenta regularidade no formato de apresentação das informações. Essa regularidade, facilmente capturada por sistemas para EI, permite que cada elemento de interesse seja identificado com base em regras uniformes, que consideram marcadores textuais tais como delimitadores, e/ou ordem de apresentação dos elementos. Como exemplo, pode-se citar um formulário preenchido.

Semi-estruturado: os textos semi-estruturados são aqueles que apresentam alguma regularidade na disposição dos dados. Alguns dados do texto podem apresentar uma formatação, enquanto outras informações aparecem de forma irregular. É o caso da primeira página de um artigo que, em geral, não segue um formato rígido, permitindo variações na ordem e na maneira com que as informações são apresentadas. Por exemplo, para mais de um autor, quando e-mails possuem o mesmo domínio, geralmente são informados de uma vez, separados por vírgula e entre chaves.

Não-estruturado: os textos não estruturados (livres) são aqueles que não exibem regularidade na apresentação dos dados. Neste caso, os dados a serem extraídos não são facilmente detectados, a menos que se tenha um conhecimento lingüístico sobre eles. Como exemplo deste tipo de texto, pode-se citar uma página Web.

2.2 Arquitetura de um Sistema de Extração de Informação

O processo de extração de informação possui duas etapas principais. Primeiro, o sistema extrai fatos (peças de informação) do texto de um documento através da análise local do texto. Segundo, o sistema integra e combina esses fatos produzindo fatos maiores ou novos fatos (por alguma inferência). Ao final, os fatos considerados relevantes ao domínio são estruturados para o padrão de saída.

Para estruturar as informações ao padrão de saída, as técnicas de EI baseadas em PLN utilizam *templates* que são estruturas com campos a serem preenchidos pelas informações nos *slots*. *Slots* são itens de dados ou partes de informação (formados por pares atributo-valor), representando as informações que devem ser extraídas de um texto.

Com base na arquitetura definida por Grishman (Grishman 1997) foram identificados seis módulos principais presentes em sistemas para EI baseados em PLN: processador léxico, reconhecimento de nomes, analisador sintático/semântico, padrões de extração, analisador do discurso e integração/preenchimento de templates - Figura 1.

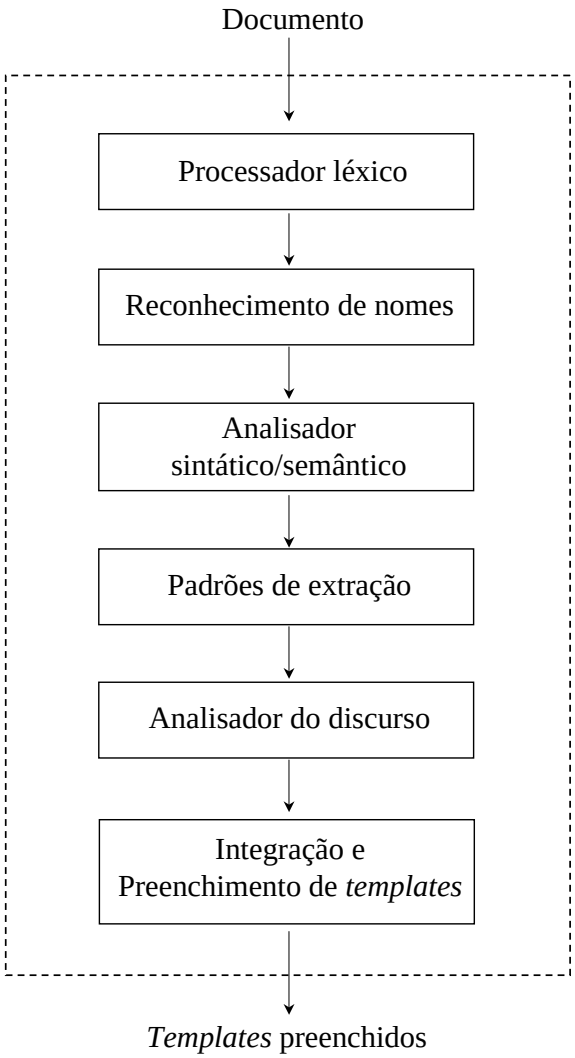


Figura 1: Estrutura de um sistema de Extração de Informação baseado em Processamento de Língua Natural

Inicialmente o texto é dividido em sentenças e em termos. Essa separação (tokenização) dos termos é realizada, normalmente, pelo reconhecimento de espaços em branco e outros sinais de pontuação que delimitam os termos. Após a separação é feita uma análise léxica e morfológica dos termos para determinar a sua possível classe morfológica (substantivo, verbo, artigo, etc.) e demais características (feminino, plural, etc.). Nesse módulo é comum a utilização de autômatos finitos para o reconhecimento das informações (Hobbs et al. 1997).

O próxima etapa de processamento identifica vários tipos de nomes próprios e outros itens que possam ter uma estrutura interna, tais como data e hora. Nomes são identificados por um conjunto de expressões regulares, expressos em função das classes morfológica (*part-of-speech*) e das características sintáticas e ortográficas (letras maiúsculas) presentes nos termos. Cowie e Lehnert citam como exemplos de reconhecimento de nomes próprios, palavras que iniciam com uma letra maiúscula e por, geralmente, virem próximas de termos como "Senhor" e ltda (Cowie & Lehnert 1996).

O módulo de análise sintática e semântica é responsável por receber uma seqüência de

itens léxicos e tentar construir uma estrutura sintática, juntamente com alguma informação semântica, para cada sentença do texto. Nas sentenças são identificados vários níveis de constituintes (segmentos de texto) e para cada constituinte dos grupos nominais e verbais conhecidos, são associadas algumas características que podem ser combinadas nas fases seguintes com os padrões de extração. Por exemplo, nos grupos nominais, pode-se incluir informações sobre a raiz do núcleo, se ele corresponde ou não a um nome próprio, assim como o seu papel semântico no contexto da frase. O papel semântico de um grupo nominal inclui informações (grupos nominais relacionados) que auxiliam a sua compressão no contexto da frase (Wiebe et al. 1996).

A construção de regras ou padrões de extração consiste na criação de um conjunto de regras de extração específico para o domínio tratado. Em geral, esses padrões baseiam-se em restrições sintáticas e semânticas, aplicadas aos constituintes das sentenças (Muslea 1999). A etapa de análise do discurso tem como objetivo relacionar diferentes elementos do texto. Esta fase considera o relacionamento entre as sentenças, ao contrário das anteriores. Caso for necessário realizar algum processo de inferência sobre a informação, tornando-a explícita, pode ser realizado nesta etapa. Este módulo inclui a tratamento das seguintes tarefas:

- análise de frases nominais, que se refere à tarefa de reconhecer e interpretar apostos e outros grupos nominais complexos,

- resolução de correferência, que trata o problema de identificar quando uma nova frase nominal, normalmente um pronome, se refere a outra já citada anteriormente, e

- descoberta de relacionamento entre as partes do texto, que objetiva estruturar as palavras do texto em uma rede associativa, fornecendo suporte à tarefa de extração.

Finalmente, as informações parciais são combinadas e os *templates*, definidos pela aplicação, são preenchidos com as informações relevantes ao domínio.

2.3 Métricas de Avaliação

A necessidade por medidas de avaliação para o problema de extração de informação, surgiu com as Conferências de Entendimento de Mensagens (MUC) em 1987. Inicialmente estas medidas foram desenvolvidas baseadas nas medidas de precisão e de cobertura, da área de RI. Contudo, as definições das medidas de EI sofreram alterações em relação as usadas em RI, apesar dos nomes serem mantidos. Essas alterações permitiram considerar generalizações em EI, nas quais diferentemente de RI, dados não presentes na entrada podem ser erroneamente produzidos (Gaizauskas & Wilks 1998).

Na tarefa de extração de informação, cobertura é definida como a quantidade de informações corretamente extraídas sobre todas as informações relevantes nos textos. Precisão

é definida como a quantidade de informações corretamente extraídas sobre todas as informações extraídas. Mais precisamente, precisão (P) e cobertura (C) são definidos como:

$$C = \frac{N_c}{N_t} \qquad e \qquad P = \frac{N_c}{N_p}$$

onde, N_c é o número de *slots* corretamente preenchidos pelo sistema, N_t o número total de *slots* relevantes a serem preenchidos e N_p é o número total de *slots* preenchidos pelo sistema. Estas medidas, no entanto, tendem a ser inversamente relacionadas: quando ocorre um aumento na cobertura, a precisão tende a diminuir e vice-versa.

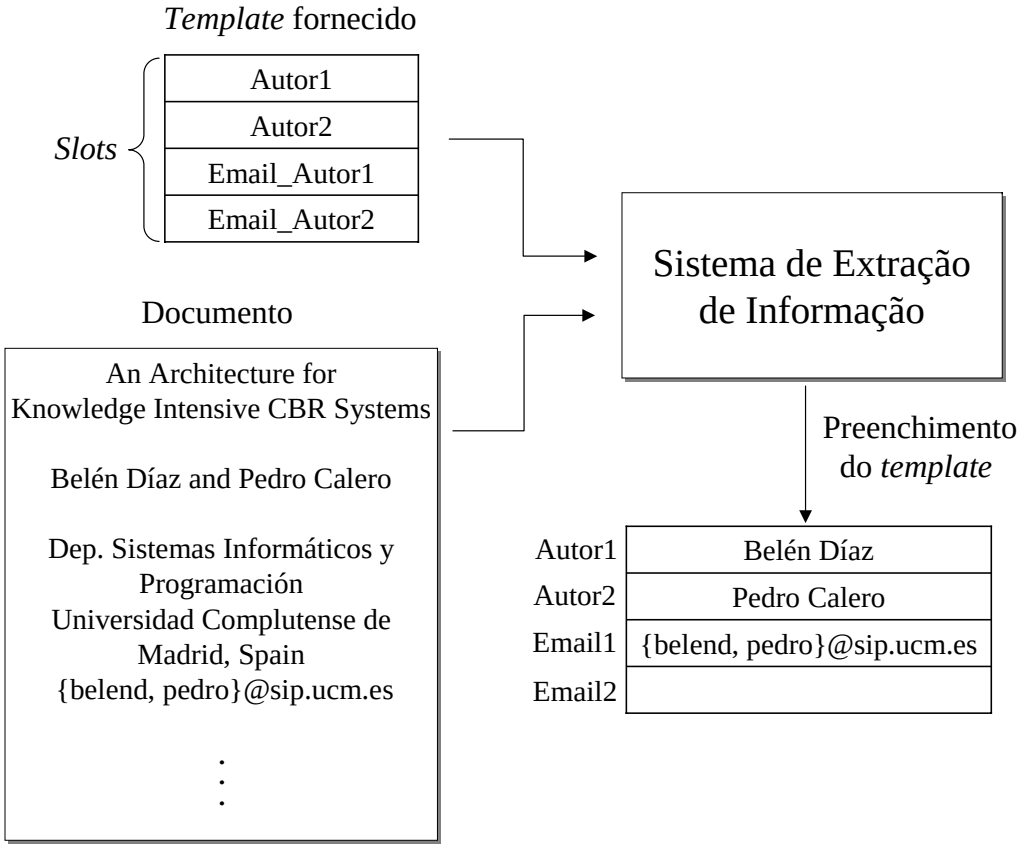


Figura 2: Exemplo de extração das informações de um documento

Para exemplificar o cálculo da precisão e cobertura, considere o exemplo de extração das informações de um documento representado na Figura 2. O documento, do qual são extraídos os autores e seus e-mails, corresponde a primeira página de um artigo científico. No exemplo, o sistema preenche corretamente dois *slots* do total de quatro *slots* relevantes, dessa forma a cobertura do sistema é igual a 2/4 (50%). Contudo, o sistema preenche erroneamente o *slot* referente ao e-mail do primeiro autor ({belend, pedro}@sip.ucm.es) e a sua precisão é igual a 2/3 (0.667%).

Na tentativa de avaliar um sistema de EI levando em consideração a cobertura e a precisão, pode-se utilizar uma outra medida chamada *F-measure*, que combina as medidas

anteriores, definida da seguinte forma:

$$F = \frac{(\beta^2 + 1) * C * P}{\beta^2 * (C + P)}$$

onde, o parâmetro β quantifica a preferência da cobertura sobre a precisão. Freqüentemente é usado $\beta = 1$ como forma de avaliar sistemas de EI balanceando as duas medidas.

2.4 Extração de Informações de Artigos Científicos

Por agir de acordo com padrões implícitos nos textos, um sistema de extração de informações geralmente é criado para atuar sobre um tipo específico de texto, por exemplo, documento Web ou publicações científicas como é o caso neste relatório, visando desse modo tratar tanto aspectos gerais do texto como exceções. Entretanto, ainda não é possível obter resultados perfeitos (Cowie & Lehnert 1996). Essa especificidade torna-se um problema, pois passa a ser impossível desenvolver um sistema genérico, que analise vários tipos de textos, automaticamente; outro problema enfrentado está relacionado com as várias formas de se representar uma mesma informação, além de possíveis erros.

A extração e estruturação dessas informações a partir dos artigos científicos deparam-se com diversos desafios relacionados com a identificação, no texto, de cada uma das informações abaixo:

1. título;
2. dados dos autores (instituição, e-mail, etc);
3. resumo (abstract);
4. palavras chaves;
5. referências bibliográficas
   (título, autor(es), ano de publicação, evento, local, editores, etc).

O processo de extração e estruturação das informações dos artigos, normalmente, não é trivial, um conjunto de regras deve ser criado de tal forma que se saiba, por exemplo, onde termina o título; onde começa e termina o nome do primeiro autor; tratamento de caracteres especiais, por exemplo, índice de nota de rodapé no nome dos autores; etc. Deve-se ainda levar em consideração o fato de muitas vezes ser necessário, ao programa, uma adaptação da regra utilizada. Por exemplo, a maneira como extrair e-mails (compare os artigos das Figuras 3 e 2), para mais de um autor, quando e-mails possuem mesmo domínio, geralmente são informados deixando entre chaves os nomes dos usuários. Essas variações também podem ocorrer com as outras informações.

A extração dos elementos das referências bibliográficas é um problema mais complexo: diferentes formatos, abreviações e padrões usados podem diferenciar referências que na verdade correspondem a um mesmo artigo.

Representing Knowledge for Case-Based Reasoning: The ROCADE System	<TÍTULO>
Béatrice Fuchsl and Alain Mille2	<AUTOR1>, <AUTOR2>
1 Université Lyon III, IAE-Modeme 15 quai Claude Bernard, 69 007 Lyon,France 2 Université Lyon I, LISI Bât. 710, 43 bd du 11 novembre 1918, 69 100 Villeurbanne	<INSTITUICAO_AUTOR1>, <INSTITUICAO_AUTOR2>
fuchs@univ-lyon3.fr amille@bat710.univ-lyonl.fr	<EMAIL_AUTOR1>, <EMAIL_AUTOR2>
Althoff et al., 1995b. Althoff K.-D., Auriol, E., Bergmann, R., Breen, S., Dittrich, S.,Johnston, R., Manago, M., Traphoner, R., and Wess, S. (1995b). Case-Based Reason-ing for Decision Support and Diagnostic Problem Solving: The INRECA Approach. In Proc. of the 3rd Workshop of the German special Interest Group on CBR at the 3rd German Expert System Conference.	<REFERENCIA>
Armengol and Plaza, 1994. Armengol, E. and Plaza, E. (1994). A Knowledge Level Model of Case-Based Reasoning. In Richter, M. M., Wess, S., Althoff K.-D., and Maurer, F., editors, First European Workshop on Case-Based Reasoning - EWGBR-93, pages 53–64, Kaiserslautern, Germany. LNAI, vol. 837, Springer, Berlin.	<REFERENCIA>
...	<REFERENCIA>
CBR KNOWLEDGE REPRESENTATION ...	<PALAVRA CHAVE 1> <PALAVRA CHAVE 2> ...

Figura 3: Estruturação das informações dos artigos

Diversos trabalhos relacionados com a extração de informações de artigos foram propostos na literatura. Em geral esses métodos se enquadram em duas categorias: abordagem baseada em regras e abordagem baseada em Aprendizado de Máquina.

Baseada em Regras Giuffrida et al. (Giuffrida et al. 2000), por exemplo, desenvolveram um sistema baseado em geras para extrair informações do cabeçalho[2] de artigos científicos, em formato PostScript. Foram utilizadas regras baseadas na análise da natureza espacial dos artigos, tal como "*títulos* são freqüentemente localizados na região superior da primeira página e possuem o maior tamanho de fonte". Ding et al. (Ding et al. 1999) utilizaram uma técnica chamada mineração de templates para extrair importantes informações de artigos e, também, de suas referências. Ding et al. usaram um template para a extração de informações do corpo do artigos e outros três templates para a extração das referências bibliográficas, e obtiveram um resultado satisfatório. Contudo, as referências analisadas pertenciam a um único estilo. Day et al. (Day et al. 2005) propuseram um método baseado em conhecimento para extrair informações de referências; foi adotado uma ontologia denominada INFOMAP (estrutura de representação de conhecimento), capaz de extrair informações de pelo menos seis estilos de referências bibliográficas com uma alta precisão. Mao et al. (Mao et al. 2004) descreveram um sistema que utiliza um conjunto de características geométricas e contextuais de artigos médicos, para realizar a extração de informações presente no cabeçalho dos artigos. (Melo et al.) (Melo et al. 2003) trabalharam na extração e principalmente na identificação automática das referências bibliográficas, o trabalho apresentou bons resultados na identificação, porém era inicialmente computacionalmente muito custoso. Algum tempo depois esse problema

[2]O cabeçalho (Seymore et al. 1999) é o conjunto todas as palavras do início do artigo até a primeira seção (normalmente a *introdução*) ou até o final da primeira página, qual ocorrer primeiro.

foi estudado e resolvido (Melo & Lopes 2004a; Melo & Lopes 2005).

Aprendizado de Máquina Connan e Omlin treinaram *Hidden Markov Models* (HMM) para reconhecer quando uma referência foi gerada por um dos seguintes estilos bibliográficos AAAI, NEWAPA, IEEE. A correta identificação do estilo possibilita que as informações (autores, título, editora, ano, etc.) sejam extraídas com precisão de até 97% (Connan & Omlin 2000). Yin et al. (Yin et al. 2004) aplicaram bigram HMM ao problema de extrair informações de referências com vários estilos, ou seja, sem o conhecimento a priori de qual estilo bibliográfico foi utilizado. A estrutura e os parâmetros do modelo HMM são automaticamente aprendidos a partir de exemplos de treinamento[3], e o sistema é capaz de alcançar uma precisão global acima de 90%. Takasu propôs um modelo estatístico chamado *Dual and Variable-length output Hidden Markov Model* (DVHMM) para a extração de atributos de referências na língua japonesa, capturadas utilizando um software OCR (Takasu 2003). O modelo gerado tem a vantagem de representar a estrutura sintática das referências e os padrões de erros causados pelo OCR. O AUTOBIB é um outro trabalho de extração e integração de informações bibliográficas de artigos recuperados na Web (Geng & Yang 2004). O AUTOBIB utiliza-se de um pequeno banco de dados de referências já estruturadas para prover um conjunto de treino para um *parser* baseado em HMM. O processo de extração é praticamente automático e utiliza referências contidas em páginas HTML. Seymore et al. (Seymore et al. 1999) também utilizaram HMM para extrair informações importantes do cabeçalho de artigos científicos em computação, e alcançaram uma precisão global de 92,9%. Seymore et al. definiram 15 informações (classes) que podem ocorrer em um cabeçalho de um documento.

Han et al. (Han et al. 2003) lidaram com o problema da extração de informações de artigos científicos como sendo um problema de classificação. A abordagem proposta utiliza Support Vector Machines (SVM) para classificar cada linha do cabeçalho em uma ou mais de 15 classes. Eles utilizaram principalmente informações lingüísticas para os atributos e, também, algum conhecimento a priori do domínio. O método foi inicialmente proposto para melhorar a qualidade da extração dos repositórios Citeseer (Lawrence et al. 1999) e eBizSearch (Petinot et al. 2003). Peng et al. (Peng & McCallum 2004) empregaram Conditional Random Fields (CRF) para extrair várias informações do cabeçalho e das referências de artigos científicos. O corpus de referências utilizado foi criado pelo projeto Cora (McCallum et al. 2000), o qual contêm 500 referências categorizadas em 13 elementos: *author*, *title*, *editor*, *booktitle*, *date*, *journal*, *volume*, *tech*, *institution*, *pages*, *location*, *publisher* e *note*.

[3]O corpos de treino é um conjunto aleatório de 250 artigos, capturados da Web, em formato PDF.

2.5 Abordagem Proposta neste Trabalho

Este projeto objetiva extrair automaticamente informações de referências bibliográficas. As referências, normalmente, possuem as seguintes informações: autores, título, publicação (periódicos, revistas, etc.) ou evento (simpósios, congressos, etc.), editora, páginas e ano.

A proposta para extrair automaticamente informações de documentos textuais, baseia-se no mapeamento do problema de *part-of-speech* (POS) *tagging* ao problema de extração de informação. Ilustrado na Figura 4, o mapeamento para a extração de informações de um documento consiste em, inicialmente, etiquetar todos os termos do documento selecionando alguma etiqueta de um conjunto pré-definido de etiquetas e, posteriormente, combinar e extrair as informações de etiquetas correspondentes. Neste mapeamento, um termo com uma etiqueta ou combinação de termos consecutivos com a mesma etiqueta equivalem a uma unidade de informação a ser extraída do documento.

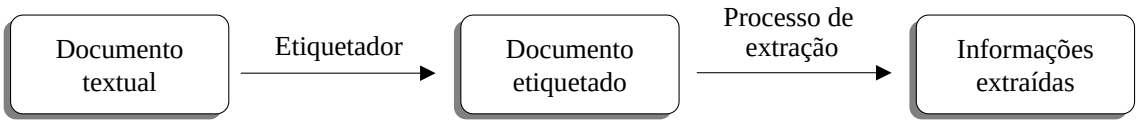


Figura 4: Mapeamento do problema de *part-of-speech* (POS) *tagging* ao problema de extração de informação

Este mapeamento para o problema de extração de informações de referências bibliográficas, consiste em etiquetar todos os termos das referências selecionando alguma etiqueta de um conjunto de possíveis etiquetas, por exemplo AUTOR, TÍTULO, REVISTA, EVENTO, LOCAL, PÁGINAS e ANO, e, posteriormente, combinar e extrair as informações (etiquetas correspondentes) das referências. Esta última etapa, extração propriamente dita, é descrita com exemplos na Seção 4.6 na página 41.

3 Etiquetagem Automática de Corpus

Borko (1967) define Processamento de Língua Natural (PLN) como sendo a manipulação, ou codificação, de uma língua para propósitos específicos tais como comunicação, tradução, armazenamento e recuperação de informações. O objetivo do processamento automático de língua natural é facilitar a manipulação e análise destas utilizando métodos computacionais.

A área de lingüística de corpus abrange uma grande variedade de níveis de análise, por exemplo um tipo de análise poderia identificar todas as ocorrências de uma determinada cadeia de caracteres em um corpus, permitindo que estas sejam ao final manipuladas. No entanto, alguns tipos de análises necessitam de mais informações a respeito das palavras presentes no corpus, como por exemplo, informações de natureza gramatical. Dessa forma, a partir de um corpus etiquetado é possível a uma ferramenta analisar e recuperar informações importantes contidas no corpus. Tais etiquetas (rótulos) são, principalmente, as

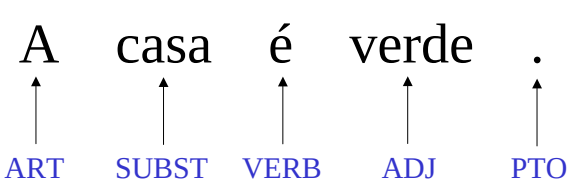
categorias gramaticais (morfossintáticas) das palavras no corpus. Pode-se, por exemplo, a partir de um corpus etiquetado, recuperar o conjunto de palavras que pertençam a uma determinada categoria gramatical.

A etiquetagem de corpus, geralmente, não é uma tarefa simples. Existem diversas pesquisas em etiquetagem de corpus com informação lingüística, incluindo categorias gramaticais e estruturas sintáticas (Marcus et al. 1994; Leech et al. 2004). A etiquetagem automática é uma tarefa bem conhecida e bastante explorada em Processamento de Língua Natural e pode ser aplicada em várias áreas do processamento de informação, tais como pré-processamento para sumarização automática, pós-processamento para reconhecimento ótico de caracteres (OCR) e reconhecimento de fala, análise sintática (*parsing*), tradução automática e recuperação de informações, e mesmo para a etiquetagem de corpus, pois a etiquetagem manual de textos grandes é uma tarefa custosa.

As ferramentas utilizadas na etiquetagem automática de corpus são os etiquetadores (*taggers*). Na Seção 3.1 são apresentados os principais conceitos sobre etiquetagem morfossintática de textos, as etapas que compõem a tarefa de etiquetagem e as abordagens de etiquetagem existentes.

3.1 Etiquetagem Morfossintática de Textos

A etiquetagem morfossintática[4] de um texto em uma dada língua significa atribuir um rótulo ou etiqueta (*tag*), pertencente a um conjunto definido de etiquetas (*tagset*), a cada palavra, símbolo de pontuação, palavra estrangeira, ou fórmula matemática presente no texto, conforme o contexto em que essas informações aparecem. Para palavras da língua, utiliza-se uma etiqueta referente a sua categoria gramatical (adjetivo, advérbio, artigo, substantivo, etc.); para símbolos de pontuação (vírgula, ponto, ponto-e-vírgula, parênteses, aspas, etc.) freqüentemente utiliza-se o próprio símbolo; palavras estrangeiras, fórmulas matemáticas, ou algum outra denominação no texto, geralmente são rotuladas com uma única etiqueta (EAGLES - Expert Advisory Group on Language Engineering Standards 1996). Um exemplo simples da etiquetagem morfossintática, de uma sentença, para a língua portuguesa é apresentado na Figura 5. É ilustrada também a possibilidade de utilizar-se uma etiqueta diferente (PTO) para designar o ponto final de uma sentença.



ART	artigo
SUBST	substantivo
VERB	verbo
ADJ	adjetivo
PTO	ponto

Figura 5: Exemplo de *part-of-speech tagging*

As etiquetas para cada classe gramatical das palavras podem, por sua vez, ser refinadas

[4]Denominada, no inglês, de *part-of-speech* (POS) *tagging*.

com atributos referentes a cada classe. Para a língua portuguesa, por exemplo, o atributo substantivo pode ser refinado com relação ao tipo (comum, próprio), grau (aumentativo, diminutivo), gênero (masculino, feminino), e assim por diante. Pode-se inclusive criar uma etiqueta para uma única palavra específica no texto. O processo geral de etiquetagem de um texto é apresentado na Figura 6. Dado um conjunto de etiquetas e uma seqüência de termos do texto, o processo geral de etiquetagem consiste em associar a cada termo a sua respectiva etiqueta.

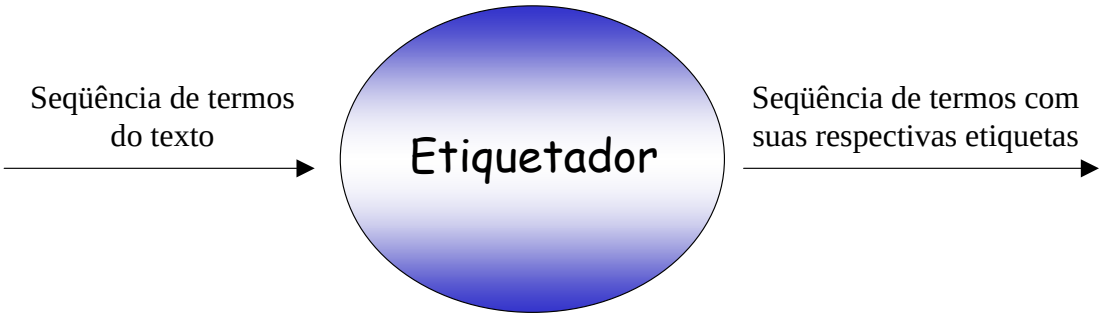


Figura 6: Processo geral de etiquetagem de texto

A tarefa de etiquetagem automática é dividida em três módulos: *escrutinador léxico*, *classificador gramatical* e *desambigüizador*. O *escrutinador léxico* é responsável por tokenizar e identificar as orações (sentenças) no texto. A tokenização separa cada termo (palavras, sinais de pontuação, etc.) no texto, normalmente, através do reconhecimento de espaços em branco. A maioria dos etiquetadores consideram que os textos de entrada estão no formado ideal e, dessa forma, não possuem um escrutinador léxico. O *classificador gramatical* designa classes gramaticais aos termos no texto com o auxílio, por exemplo, de um léxico e de um conjunto de informações para reconhecer termos (normalmente palavras) que não estão contidos no léxico. O *desambiguizador* utiliza informações relativas ao contexto para associar uma, e somente uma, classe gramatical a cada termo do corpus. Este método objetiva resolver o problema de ambigüidade lexical, situação em que uma mesma palavra possua mais de uma categoria gramatical, e, nestes casos, o contexto deve ser considerado para a desambiguação. Tanto o léxico, quanto as informações utilizadas para avaliar o contexto integram o modelo da língua utilizado por cada etiquetador.

Os etiquetadores podem ser construídos automaticamente, a partir de um corpus de treino etiquetado, ou (manualmente) elaborados por lingüistas que, utilizando conhecimentos lingüísticos, desenvolvem um conjunto de regras de etiquetagem. Conforme o tipo de conhecimento utilizado para representar o modelo da língua, os etiquetadores podem ser classificados nas seguintes abordagens.

Simbólica: os etiquetadores da abordagem simbólica (ou lingüística) são baseados em regras (Voutilainen 1995; Lopes & Jorge 2000), casos (Daelemans et al. 1996), restrições (Chanod & Tapanainen 1995) e árvores de decisão não probabilísticas.

Probabilística: etiquetadores probabilísticos fundamentam-se em técnicas como redes

neurais (Ma et al. 1999), máxima entropia (Ratnaparkhi 1996), Modelo de Markov (Wilkens & Kupiec 1996) e árvores de decisão probabilísticas (Schmid 1995).

Híbrida: os etiquetadores da abordagem híbrida empregam tanto conhecimento simbólico quanto probabilístico no processo de etiquetagem. Como um exemplo desta abordagem pode-se citar o etiquetador TBL (baseado em transformação dirigida por erro) (Brill 1994; Brill 1995; Brill 1997).

Atualmente, os etiquetadores mais utilizados são os probabilísticos. Seu funcionamento basicamente consiste na construção de um modelo estatístico da língua que é utilizado para desambigüisar um conjunto de palavras. Este modelo, em geral, aparece como um conjunto de freqüências de diferentes tipos de fenômenos lingüísticos e é construído através da observação de n-gramas, sendo que o mais comum é a modelagem na forma de unigramas, bigramas e trigramas. Os n-gramas são seqüências de n termos adjacentes.

De forma a maximizar as vantagens encontradas nas abordagens simbólica e probabilística e minimizar as desvantagens, Eric Brill desenvolveu um etiquetador híbrido que contém um componente estatístico e outro simbólico.

Neste trabalho optou-se pelo uso do etiquetador de Eric Brill por ser bastante conhecido, disponível e de uso relativamente fácil.

3.2 Etiquetador Baseado em Transformação Dirigida por Erro

O etiquetador desenvolvido por Eric Brill, denominado TBL, baseia-se no algoritmo de Aprendizado Baseado em Transformação Dirigida por Erro (Brill 1994; Brill 1995; Brill 1997). Tal algoritmo, ilustrado na Figura 7, pode ser aplicado a vários problemas, tais como a etiquetagem morfossintática e a análise sintática.

3.2.1 Aprendizado Baseado em Transformação Dirigida por Erro

Conforme apresentado na Figura 7, o corpus não-etiquetado passa inicialmente por uma etapa de etiquetagem. O sucesso do algoritmo depende em grande parte do sucesso desta etapa inicial, a qual pode ser baseada na freqüência de categorias gramaticais para cada palavra de um léxico. Após ter sido etiquetado, o corpus é comparado com o mesmo corpus manualmente etiquetado, e uma lista de transformações é aprendida para ser aplicada à saída do etiquetador inicial. Cada transformação é composta por uma regra de reescrita e pelo contexto que irá desencadear esta regra[5]. A cada iteração de aprendizado, de acordo com alguma função objetivo, uma transformação que melhora o resultado da etiquetagem é encontrada e adicionada à lista ordenada de transformações, e o corpus etiquetado é atualizado aplicando sobre ele a transformação aprendida. Esse processo continua até que não seja mais encontrada uma transformação que melhore o corpus etiquetado.

[5]Observar que algumas palavras têm apenas um papel morfossintático (*tag*), enquanto outras podem ter mais de um e, neste caso, o contexto tem que ser considerado para a desambiguação

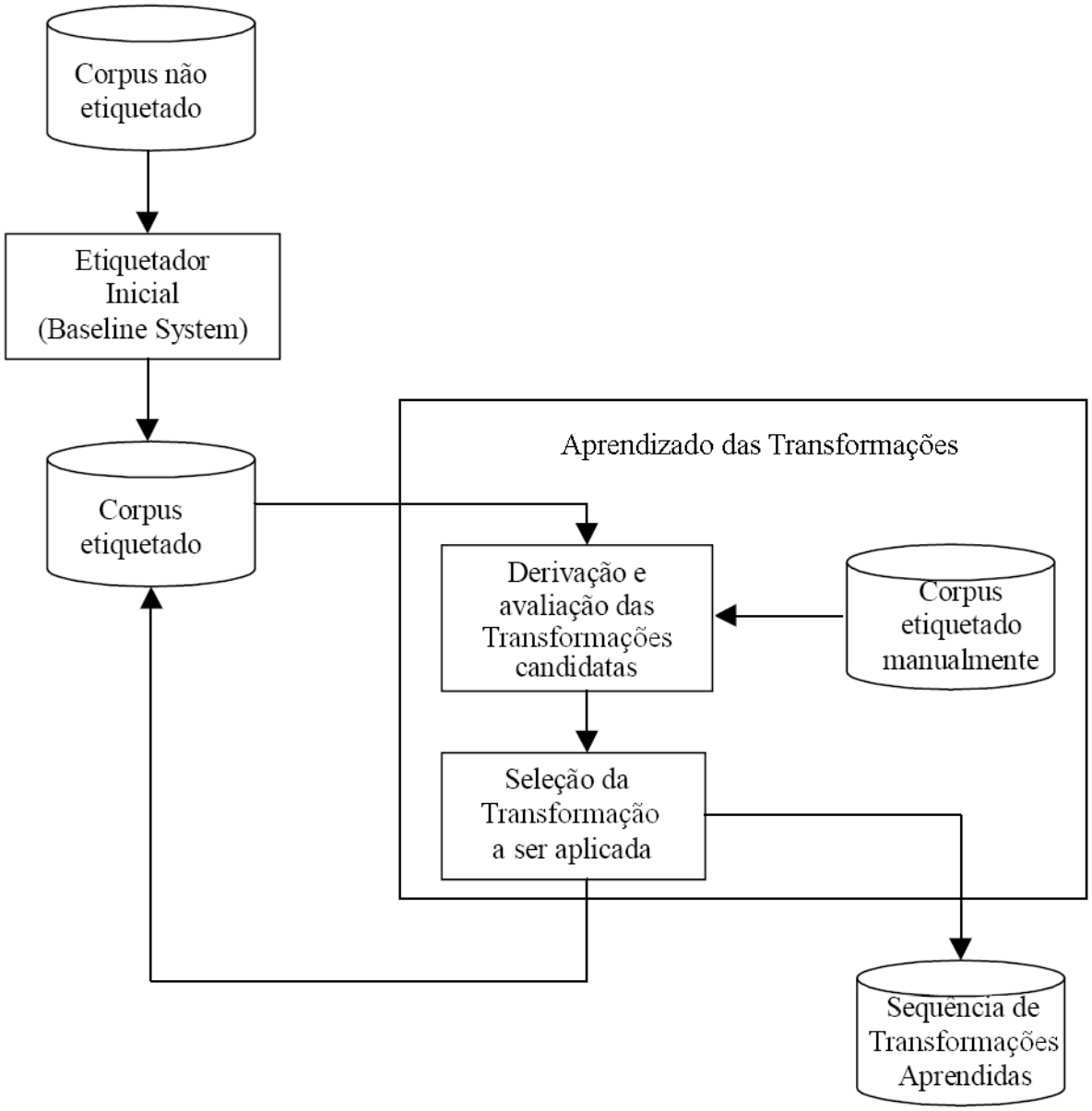


Figura 7: Algoritmo de Aprendizado Baseado em Transformação Dirigida por Erro

Um exemplo de aprendizado baseado em transformações é ilustrado na Figura 8. Neste exemplo assume-se que existam apenas quatro transformações possíveis (T1,T2,T3 e T4) e que a função objetivo utilizada seja o número total de erros. O corpus não-etiquetado passa pela etiquetagem inicial, e o resultado é um corpus etiquetado com 5100 erros. Em seguida, cada uma das transformações selecionadas são aplicadas *em ordem*. Neste exemplo, T2 foi a primeira transformação da lista, pois foi a transformação que possibilitou a maior redução de erros. É então aplicada a todo o corpus e o aprendizado continua. Em seguida a transformação que mais diminuiu o número de erros foi T3, logo T3 é aprendida como a segunda transformação da lista. Partindo de T3 nota-se que não existem mais reduções no número de erros aplicando transformações, então o processo termina.

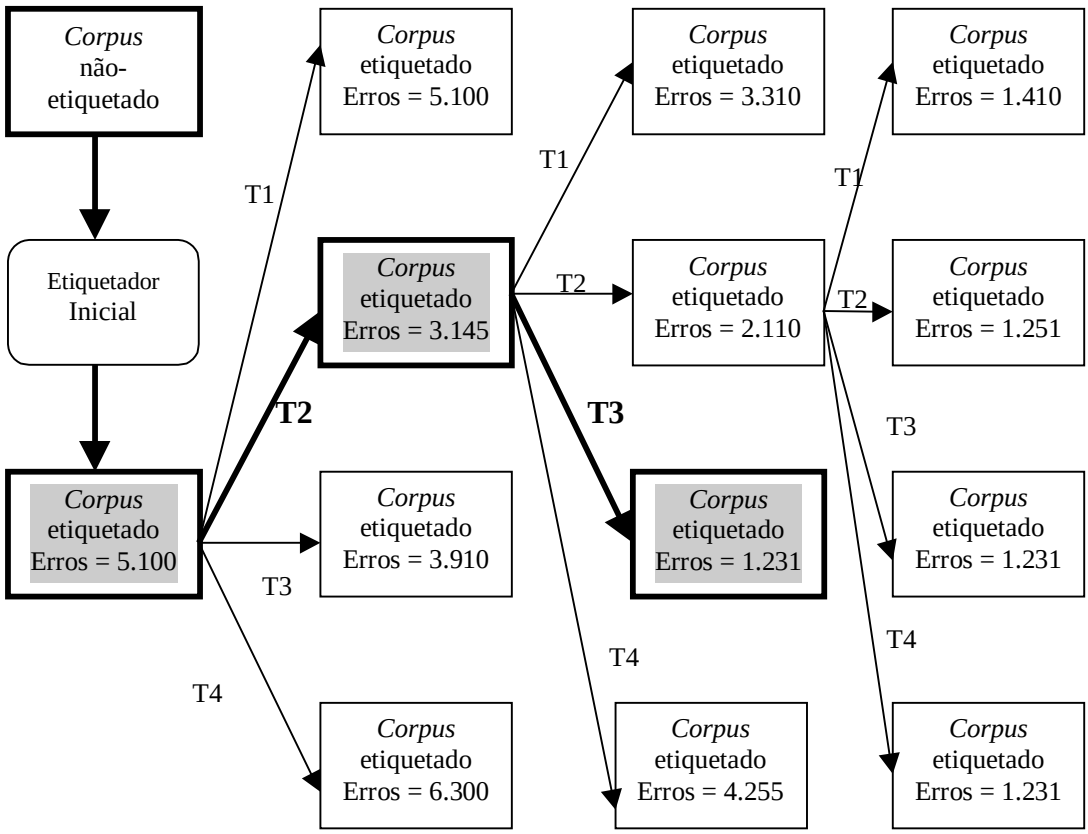


Figura 8: Exemplo de Aprendizado Baseado em Transformação Dirigida por Erro

3.2.2 Etiquetador TBL

O etiquetador TBL possui dois módulos de etiquetagem: um módulo que gera regras, não contextuais, para determinar o conjunto de etiquetas mais prováveis de palavras (conhecidas e desconhecidas) e outro módulo que gera regras contextuais para melhorar a precisão da etiquetagem. O etiquetador apenas altera a etiqueta de uma palavra de X para Y se:

- a palavra não aparece no corpus de treinamento, ou
- a palavra foi etiquetada com Y pelo menos uma vez no corpus de treinamento.

O primeiro módulo, responsável pela etapa de etiquetagem inicial, constrói um léxico a partir do corpus manualmente etiquetado com a finalidade de atribuir a etiqueta mais provável para cada palavra - Figura 9. O léxico é formado pela palavra seguida de sua etiqueta mais comum.

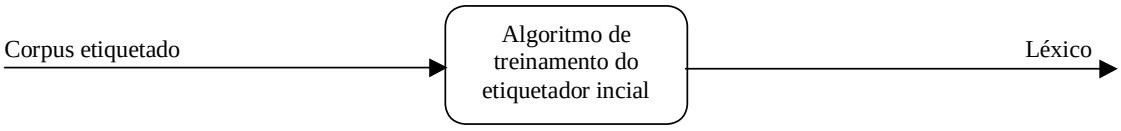


Figura 9: Etiquetador Baseado em Transformação Dirigida por Erro: treinamento do etiquetador inicial

Para etiquetar palavras desconhecidas (não presentes no léxico), o etiquetador inicial utiliza outro procedimento. Inicialmente, define que as palavras desconhecidas que iniciam por letra maiúscula tendem a ser substantivos próprios e as que iniciam por letras minúsculas tendem a ser substantivo comum. Em seguida o etiquetador gera regras léxicas para minimizar o erro desta etiquetagem inicial, utilizando apenas informações intrínsecas das palavras, ou seja, sem considerar o contexto. Alguns modelos de regras para palavras desconhecidas podem ser vistos na Tabela 1.

Mude a etiqueta de uma palavra desconhecida de A para B se:
1. Removendo prefixo (sufixo) **l**, |**l**| <= 4, resulta em uma palavra no léxico (**l** é qualquer *string* de tamanho 1 a 4)
2. O primeiro (último) (1,2,3,4) caracteres da palavra é **c**
3. Adicionando a string **m** como prefixo (sufixo) resulta em uma palavra (|**m**|<=4)
4. A palavra **m** nunca aparece imediatamente a esquerda (direita) da palavra
5. O caractere **c** aparece na palavra

Tabela 1: Regras para palavras desconhecidas

Mude a etiqueta de uma palavra de A para B quando:
1. A palavra anterior (posterior) foi etiquetada como **z**
2. A palavra duas posições antes (depois) foi etiquetada como **z**
3. Uma das duas palavras anteriores (posteriores) foi etiquetada como **z**
4. Uma das três palavras anteriores (posteriores) foi etiquetada como **z**
5. A palavra anterior foi etiquetada como **z** a posterior como **x**
6. A palavra anterior (posterior) foi etiquetada como **z** e a palavra "duas posições antes" foi etiquetada como **x**
7. A palavra que vem antes (depois) é **w**
8. A segunda palavra que vem antes (depois) é **w**
9. Se uma das duas palavras que vem antes (depois) é **w**
10. Se a palavra atual é **w** e a anterior (posterior) é **y**
11. Se a palavra atual é **w** e a palavra anterior (posterior) é classificada como **t**
12. Se a palavra atual é **w**
13. Se a palavra anterior (posterior) é **w** e a classificação anterior (posterior) é **t**
14. Se a palavra atual é **w** e a palavra anterior (posterior) é **y** e a classificação anterior (posterior) é **t**

Tabela 2: Regras que utilizam informações sobre o contexto

O segundo módulo, etiquetador contextual, infere automaticamente as regras relativas ao contexto, a partir do corpus de treinamento etiquetado. Isto é feito realizando a etiquetagem do corpus de treinamento, utilizando o etiquetador inicial, e em seguida comparando automaticamente os resultados e gerando a lista ordenada de transformações. A partir da aplicação desta lista de transformações no corpus, são geradas as regras de contexto. As aplicações que gerarem o melhor resultado são utilizadas como regras de contexto - Figura 10. Dessa forma, obtém-se um conjunto de regras que analisa a

atribuição das etiquetas feita pelo etiquetador inicial e as corrige conforme o contexto no qual as palavras aparecem.

Como apresentado na Tabela 2, as regras contextuais podem fazer referência a etiquetas anteriores/posteriores (regras não lexicalizadas) ou a palavras anteriores/posteriores (regras lexicalizadas). Na Tabela 2 as variáveis **z**, **x** e **t** referem-se a todas as classes gramaticais (etiquetas) e as variáveis **w** e **y** abrangem todas as palavras do corpus de treinamento.

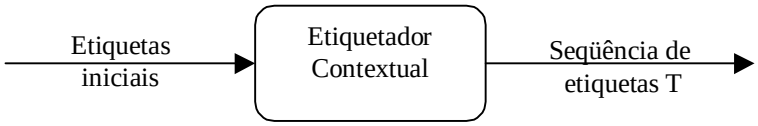


Figura 10: Etiquetador Baseado em Transformação Dirigida por Erro: etiquetador contextual

Resumindo, a etiquetagem de um novo corpus, assumindo que o etiquetador esteja treinado, consiste em submeter o corpus ao etiquetador inicial para a etiquetagem de palavras conhecidas (léxico) e desconhecidas, e posteriormente aplicar (uma de cada vez) a lista de regras contextuais no corpus. As transformações que estão mais próximas do início da lista são as transformações que produzem um melhor resultado no corpus etiquetado. É importante ressaltar que uma transformação feita no início, pode ocasionar outras transformações posteriormente.

4 Extração de Informação Utilizando POS-tagging

Nas próximas seções são descritos todos os procedimentos necessários à realização de experimentos para extração de informações de referências bibliográficas baseando-se no mapeamento descrito na Seção 2.5.

4.1 Definições de Projeto

Inicialmente, foi necessário adotar algumas decisões de projeto para a realização dos experimentos. É importante ressaltar, que essas decisões foram tomadas de acordo com as necessidades da FIP, comentada brevemente na introdução (Brasil & Lopes 2004; Melo & Lopes 2004a; Melo & Lopes 2004b; Melo & Lopes 2005).

Para a definição do conjunto de etiquetas das referências bibliográficas foi necessário verificar quais são as informações que constituem uma referência. Um estudo sobre as principais normas (padrões) para a elaboração de referências bibliográficas foi realizado e optou-se por utilizar, neste trabalho, as normas técnicas definidas pela NBR6023 da Associação Brasileira de Normas Técnicas (2002), as normas do *Chicago Manual of Style Guide* baseadas no *Chicago Manual Style* (1993) e as especificações técnicas da *American Psychological Association Publication Manual* (1994). O conjunto de etiquetas também foi

baseado no BibTEX, que é um formato para referências bibliográficas usado em combinação como o sistema de processamento de texto TEX/LATEX[6].

Esses padrões basicamente são compostos por tipos de referências (livros, artigos, revistas, eventos, etc.) e as informações que devem ser preenchidas em cada tipo, divididas em obrigatórias e opcionais. Neste trabalho não houve distinção entre tais informações, visto que a FIP deve ser capaz de lidar com todos os tipos de informações e, conseqüentemente, com todos os tipos de referências. Os tipos de referências bibliográficas, juntamente com a suas etiquetas, considerados neste projeto são descritos a seguir.

- ARTIGO
  - AUTHOR: *autor(es) do artigo;*
  - TITLE: *título do artigo;*
  - JOURNAL: *revista que publicou o artigo;*
  - YEAR: *ano de publicação;*
  - VOLUME: *volume da revista;*
  - NUMBER: *número da revista;*
  - INITPAGE: *página inicial do artigo na revista;*
  - FINALPAGE: *página final do artigo na revista;*
  - MONTH: *mês de publicação;*
  - URL: *link para o arquivo (pdf, ps);*
  - URLACCESSDATE: *data em que a url foi acessada;*
  - NOTE: *alguma informação adicional;*
  - ISSN: *numeração internacional para publicações periódicas*
- LIVRO
  - AUTHOR: *autor(es) do livro;*
  - EDITOR: *editor(es) do livro;*
  - TITLE: *título do livro;*
  - PUBLISHER: *editora do livro;*
  - YEAR: *ano de publicação;*
  - VOLUME: *volume do livro;*
  - NUMBER: *número do livro;*
  - SERIES: *série que publicou o livro;*
  - ADDRESS: *endereço completo, ou apenas a cidade, da editora;*

[6] TEX/LATEX é, de fato, um padrão para publicações em várias áreas de pesquisa.

  - EDITION: *edição do livro;*
  - MONTH: *mês de publicação;*
  - NOTE: *alguma informação adicional;*
  - ISBN: *numeração internacional para livros*

- CAPÍTULO DE LIVRO
  - AUTHOR: *autor(es) do livro;*
  - EDITOR: *editor(es) do livro;*
  - TITLE: *título do capítulo;*
  - BOOKTITLE: *título do livro;*
  - CHAPTER: *número do capítulo;*
  - INITPAGE: *página inicial;*
  - FINALPAGE: *página final;*
  - PUBLISHER: *editora do livro;*
  - YEAR: *ano de publicação;*
  - VOLUME: *volume do livro;*
  - NUMBER: *número do livro;*
  - SERIES: *série que publicou o livro;*
  - TYPE: *algo sobre o tipo do documento, por ex.: a seção;*
  - ADDRESS: *endereço completo, ou apenas a cidade, da editora;*
  - EDITION: *edição do livro;*
  - MONTH: *mês de publicação;*
  - NOTE: *alguma informação adicional;*
  - ISBN: *numeração internacional para livros*

- CONGRESSOS, CONFERÊNCIAS, SIMPÓSIOS, WORKSHOPS
  - EDITOR: *editor(es) do evento;*
  - TITLE: *título do evento;*
  - PUBLISHER: *editora do evento;*
  - YEAR: *ano de evento;*
  - SERIES: *série do evento;*
  - ORGANIZATION: *órgão patrocinador do evento;*
  - ADDRESS: *endereço completo, ou apenas a cidade, da editora;*

  - MONTH: *mês de publicação;*
  - NOTE: *alguma informação adicional*

- TRABALHOS APRESENTADOS EM EVENTOS
  - AUTHOR: *autor(es) do trabalho;*
  - TITLE: *título do trabalho;*
  - BOOKTITLE: *título do evento;*
  - YEAR: *ano de publicação;*
  - EDITOR: *editor(es) do trabalho;*
  - SERIES: *série do evento;*
  - INITPAGE: *página inicial;*
  - FINALPAGE: *página final;*
  - PUBLISHER: *editora;*
  - ORGANIZATION: *órgão patrocinador do evento;*
  - ADDRESS: *endereço completo, ou apenas a cidade, da editora;*
  - MONTH: *mês de publicação;*
  - URL: *link para o arquivo (pdf, ps);*
  - URLACCESSDATE: *data em que a url foi acessada;*
  - NOTE: *alguma informação adicional;*
  - ISSN: *numeração internacional para publicações periódicas*;
  - ISBN: *numeração internacional para livros*

- DISSERTAÇÕES E TESES
  - AUTHOR: *autor da monografia;*
  - TITLE: *título da monografia;*
  - SCHOOL: *universidade no qual a monografia foi escrita;*
  - YEAR: *ano de publicação;*
  - TYPE: *algo que indique se é uma dissertação ou tese;*
  - ADDRESS: *departamento no qual a monografia foi escrita;*
  - MONTH: *mês de publicação;*
  - URL: *link para o arquivo (pdf, ps);*
  - URLACCESSDATE: *data em que a url foi acessada;*
  - NOTE: *alguma informação adicional;*

  - ISBN: *numeração internacional para livros*

- RELATÓRIO TÉCNICO
  - AUTHOR: *autor do relatório;*
  - TITLE: *título do relatório;*
  - INSTITUTION: *instituição no qual o relatório foi escrito;*
  - YEAR: *ano de publicação;*
  - TYPE: *algo que indique se é um relatório técnico;*
  - NUMBER: *número do relatório;*
  - ADDRESS: *departamento no qual o relatório foi escrito;*
  - MONTH: *mês de publicação;*
  - NOTE: *alguma informação adicional*

- MÍDIA (CD-ROM)
  - AUTHOR: *autor da mídia;*
  - TITLE: *título da mídia;*
  - PUBLISHER: *editora;*
  - ADDRESS: *endereço completo, ou apenas a cidade, da editora;*
  - NOTE: *alguma informação adicional*

- PÁGINA DE INTERNET
  - AUTHOR: *autor da página;*
  - TITLE: *título da página;*
  - URL: *link para a página;*
  - URLACCESSDATE: *data em que a url foi acessada;*
  - NOTE: *alguma informação adicional*

Uma descrição do conjunto de etiquetas criado, denominado FIP tagset, é dado na Apêndice A. É importante ressaltar que mesmo elaborando-o de forma cuidadosa, o FIP tagset sofreu alterações durante o processo de etiquetagem manual do corpus. Uma das modificações foi a inclusão de uma etiqueta, nominada INDICATOR, que engloba todos os termos que indicam a presença de informações, por exemplo: pp, In, No, pages, etc.

Nos experimentos foi utilizado o etiquetador TBL, desenvolvido por Eric Brill, que se baseia no algoritmo de Aprendizado Baseado em Transformação Dirigida por Erro (Brill 1994; Brill 1995; Brill 1997).

4.2 Pré-processamento

O corpus utilizado nos experimentos possui referências bibliográficas da área de Computação e abrange os estilos bibliográficos Plain, Alpha, Abbrv, Apalike e Chicago. Na Seção 4.3, uma descrição completa do processo de construção e etiquetagem do corpus é apresentada. Inicialmente, para que o corpus pudesse ser utilizado vários ajustes, relacionados à tarefa de pré-processamento, foram realizados:

- correção de erros provenientes do software[7] de conversão de arquivos no formato .PDF ou .PS para o formato .TXT;
- remoção de espaços duplos;
- padronização de um conjunto de caracteres, por ex.: todos os caracteres similares ao caractere hífen, foram substituídos pelo mesmo;
- tokenização do corpus;
- substituição da barra (/) pelo símbolo $b, pois o etiquetador TBL utiliza a barra para separar a palavra de sua etiqueta;
- formatação do corpus resultando em uma referência completa por linha

O software na maioria dos casos realiza perfeitamente a conversão, entretanto em algumas situações podem ocorrer alguns erros, por ex.: substituição do conjunto de caracteres "ffi" por Æ. A idéia de padronizar um conjunto de caracteres, para a tarefa de etiquetagem/extração, surgiu das seguintes observações:

- um corpus que contenha uma grande variedade[8] de sinais de pontuação poderia dificultar a tarefa do etiquetador, e
- a padronização de caracteres não afetaria o resultado do sistema de extração, ou seja, as informações que deveriam ser extraídas, com ou sem padronização, seriam as mesmas

Antes da etapa de pré-processamento, cada referência corresponde a várias linhas consecutivas no corpus. Para identificar uma referência completa, foram criados vários modelos (padrões), a partir da análise do corpus, que informam quando uma linha é início/fim de uma referência. A grande maioria das referências no corpus possuem indicadores de início de referência, o que, conseqüentemente, facilita a criação dos modelos de identificação. Esses indicadores normalmente assumem os seguintes formatos:

[7]Para a conversão dos arquivos foram utilizados os comandos *pdftotext* versão 3.0 (aplicativo linux) e *pstotext* versão 1.9 (aplicativo windows)

[8]O corpus possui caracteres similares aos caracteres: hífen, aspas simples e aspas duplas.

número. *corpo da referência*,
[número]. *corpo da referência*,
[conjunto de caracteres alfanuméricos, ano] *corpo da referência* e
(conjunto de caracteres alfanuméricos, ano) *corpo da referência*.

No entanto, para algumas referências que não foi possível criar modelos, foram criadas heurísticas na tentativa de identificá-las. Algumas das heurísticas utilizadas são apresentadas a seguir.

1. Uma linha do corpus não é início de referência quando possui tamanho menor que 40 e não contém as palavras http e ftp; uma referência, muito raramente, pode ser composta apenas por um *link* de tamanho pequeno.

2. Uma linha que começa por uma preposição não é início de referência.

3. Caso uma linha termine por uma preposição, a linha seguinte não é início de referência.

4. Uma linha que começa por uma letra minúscula não é início de referência.

5. Uma linha é inicio de referência quando contém um ano entre parênteses.

Observando as heurísticas acima é possível notar que existe uma maior facilidade em criar regras, para determinar quando uma linha do corpus não é início de referência. Vale ressaltar também que a ordem na qual as heurísticas são utilizadas influenciam no resultado final, ou seja, dependendo da ordem de execução destas, algumas referências, antes identificadas corretamente, podem ser erroneamente identificadas. O programa denominado "fixrefs", escrito em Perl, que realiza todas as tarefas de pré-processamento descritas anteriormente, pode ser visto no Apêndice B.

Ao final desta etapa constatou-se que alguns erros ainda não haviam sido corrigidos, isso se deve principalmente a grande quantidade de padrões e formatos, e também por existirem algumas referências que possuem estruturas raras, tal como uma tabela. Foi realizada mais uma inspeção no corpus e vários erros foram corrigidos manualmente. Acredita-se que ainda possam existir alguns erros, porém em uma porcentagem bem pequena.

4.3 Processo de Etiquetagem do Corpus

Antes de realmente iniciar os experimentos com o etiquetador TBL é necessário ter um corpus etiquetado, razoavelmente grande[9], que contenha uma grande variedade de modelos. O corpus utilizado nos experimentos é composto por uma pequena parcela etiquetada manualmente e por outra automaticamente etiquetada utilizando informações contidas em arquivos BibTEX. Como comentado anteriormente, o corpus possui referências bibliográficas da área de Computação e abrange os estilos bibliográficos Plain, Alpha,

[9]Na Figura 12 pode ser observada a relação do tamanho do corpus e a precisão do etiquetador

Abbrv, Apalike e Chicago. Algumas informações, consideradas relevantes, sobre o corpus são apresentadas na Tabela 3.

	Estilo bibliográfico	Quantidade de palavras	Quantidade de referências	Etiquetagem
1	Plain	215726	5000	Automática
2	Alpha	267679	5000	Automática
3	Abbrv	219061	4996	Automática
4	Chicago	220810	4993	Automática
5	Apalike	177326	3992	Automática
6	Aleatório	34384	947	Manual

Tabela 3: Informações sobre o corpus

Na Seção 4.3.1 é descrito em detalhes o processo de construção e etiquetagem automática de grande parte do corpus - linha 2 a 6 na Tabela 3. O processo de etiquetagem manual de uma pequena parcela do corpus é apresentado na Seção 4.3.2.

4.3.1 Etiquetagem Automática

Um arquivo com extensão .BIB, chamado de base BibTEX, apresenta informações de referências bibliográficas separadas por campos, denominados campos BibTEX. A partir de um arquivo .BIB etiquetado (as informações de cada campo estão etiquetadas) é possível criar um conjunto de referências etiquetadas. Posteriormente, o procedimento utilizado para a etiquetagem de arquivos .BIB é apresentado.

Para a utilização do BibTEX é necessário especificar dois parâmetros: o estilo bibliográfico (arquivo .BST) e a base (arquivo .BIB). Atualmente, existe uma grande variedade de arquivos de estilos bibliográficos para o BibTEX disponíveis na WWW. Optou-se por utilizar neste trabalho os estilos Plain, Alpha, Abbrv, Apalike e Chicago, disponibilizados pela CTAN[10], pois estão entre os estilos mais utilizados pela comunidade científica. Os estilos Plain, Alpha, Abbrv e Unsrt são os chamados *estilos bibliográficos padrão* que serviram de base para a criação de todos os outros estilos existentes. O estilo Unsrt não foi utilizado por ser como o Plain, exceto que as referências não são alfabeticamente ordenadas.

A base utilizada foi recuperada do repositório *The Collection of Computer Science Bibliographies*[11]. *The Collection of Computer Science Bibliographies* é uma coleção de literaturas científicas disponíveis, abrangendo os principais tópicos em Ciência da Computação. A base utilizada é composta por vários arquivos .BIB de diferentes tamanhos.

O procedimento para a construção de um conjunto de referências automaticamente etiquetadas é descrito a seguir. O BibTEX em sua implementação aceita no máximo 5000 citações, ou seja, um arquivo .BIB tem que possuir no máximo 5000 entradas. As entradas são os possíveis tipos de referências, por exemplo article, book, inbook, inproceedings, etc. Dentre os vários arquivos .BIB da base foram selecionados cinco[12], um para cada

[10] http://ctan.org/

[11] http://liinwww.ira.uka.de/bibliography/index.html

[12] Inicialmente um arquivo com mais de 5000 entradas é selecionado e, posteriormente, através de uma inspeção manual, seu número é reduzido próximo de 5000.

estilo, cada um com no máximo 5000 entradas. Em seguida é utilizado o programa JabRef[13], versão 1.8.1, para verificar se existe erros sintáticos e inconsistências nos arquivos .BIB selecionados. Após a definição e ajustes dos arquivos .BIB, o próximo passo é a tarefa de etiquetagem propriamente dita. Para uma melhor compreensão é ilustrado na Figura 11, um exemplo da etiquetagem da entrada INCOLLECTION contida em um arquivo chamado 00.bib da base.

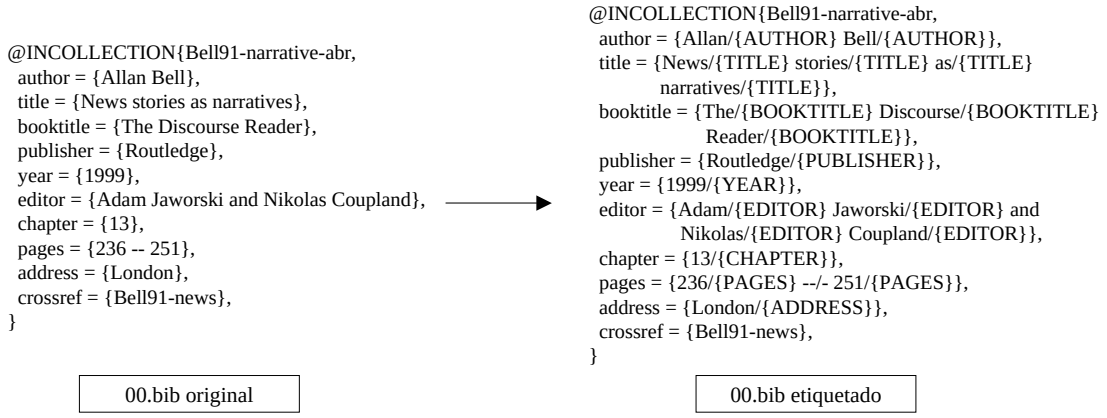


Figura 11: Exemplo da etiquetagem de um arquivo .BIB

Conforme observado neste exemplo, a etiquetagem consiste em para cada campo (author, title, booktitle, publisher, etc.) etiquetar as suas informações, delimitadas por chaves, com o rótulo correspondente ao campo. Algumas informações por terem um significado especial para o BibTEX não são etiquetadas, por exemplo a palavra 'and' localizada no campo author ou editor não deve ser etiquetada. O programa denominado "bibtex", escrito em Perl, que realiza a etiquetagem das informações de cada campo para cada entrada de um arquivo .bib, pode ser visto no Apêndice B. Finalizando, para construir um conjunto de referências automaticamente etiquetadas basta apenas selecionar um arquivo BibTEX etiquetado e um estilo bibliográfico, e utilizar o BibTEX em combinação com o LATEX para gerar um documento que contenha todas as suas referências corretamente etiquetadas. Utilizando o método acima descrito, foi possível construir grande parte do corpus etiquetado.

4.3.2 Etiquetagem Manual

Como dito anteriormente, uma pequena parcela do corpus foi etiquetada manualmente. Esta fração do corpus foi inicialmente recuperada pelo WebMiner (Brasil & Lopes 2004) e possui referências bibliográficas das áreas de Recuperação de Informação e Processamento de Língua Natural.

Em trabalhos sobre etiquetagem morfossintática, no qual, inicialmente não exista um corpus etiquetado (*gold standard*), normalmente realiza-se um extenso trabalho manual. Existe também a possibilidade de utilizar uma ferramenta semi-automática de auxílio à etiquetagem. Infelizmente para este trabalho, no momento da etiquetagem desta porção

[13]http://jabref.sourceforge.net/

do corpus, não existia um corpus inicialmente etiquetado e nem uma ferramenta de auxílio à etiquetagem, no entanto tal problema foi resolvido através de um processo iterativo e interativo de etiquetagem semi-automático[14], sintetizado no Algoritmo 1.

Algoritmo 1: Criação do Corpus Etiquetado

```
Input: Tr, Ts, ε
/* Conjunto inicial de treino etiquetado (100 referências), Conjunto
   de teste, e Erro máximo                                          */
Output: C (Corpus Etiquetado)
// Conjunto de treino e teste etiquetado
begin
    /* Inicia Corpus Etiquetado                                     */
    C ← Tr;
    // Inicia conjunto de regras de etiquetagem
    Regras ← ∅;
    // Induz primeiro conjunto de regras
    Regras ← Induzir_Etiquetador(Tr);
    /* Com Regras, etiqueta o conjunto de teste e determina Erro na
       etiquetagem                                                  */
    Etiquetar(Ts);
    while Erro na etiquetagem > ε do
        Usa Regras e etiqueta Novas_Ref (100 novas referências) ;
        Corrige manualmente os erros destas novas e determina Erro na
        etiquetagem ;
        // adiciona Novas_Ref corrigido ao treino
        C ← C ∪ Novas_Ref ;
        // novo conjunto de regras
        Regras ← Induzir_Etiquetador(C) ;
    end
end
```

Após a primeira iteração, o conjunto de treino para o TBL são 200 referências, as quais já estão revisadas e tal processo descrito acima é repetido. Alcançando um conjunto de 500 à 700 referências, surgiram alguns problemas:

- o etiquetador não gerava regras contextuais e a mensagem "*Segmentation Fault, core dumped*" era exibida, e
- testes com configurações iguais, ou seja sem modificar os parâmetros do etiquetador e utilizando o mesmo corpora de treino, apresentavam saídas diferentes

[14]Este processo de construção de um corpus manualmente etiquetado foi sugerido por Eric Brill, na documentação do seu etiquetador.

Os problemas foram resolvidos modificando o formato de todos arquivos utilizados pelo etiquetador, para UNIX. Até o presente momento foi etiquetado e corrigido manualmente um conjunto de 947 referências bibliográficas, totalizando 34384 palavras. O gráfico comparativo da precisão do etiquetador com linha de tendência durante esse processo de etiquetagem manual é apresentado na Figura 12. O etiquetador geralmente aumenta a sua precisão quando um corpus maior de treino é usado, porém para os conjuntos de treino com 500 e 700 referências, o etiquetador obteve um decréscimo em sua precisão. A justificativa para este problema decorre da grande quantidade de novas palavras e, principalmente, de novos modelos de referências, presentes no corpus de teste. No entanto, com uma precisão do etiquetador de no mínimo 74%, este processo de construção de um corpus torna-se mais viável do que etiquetar manualmente palavra a palavra no corpus.

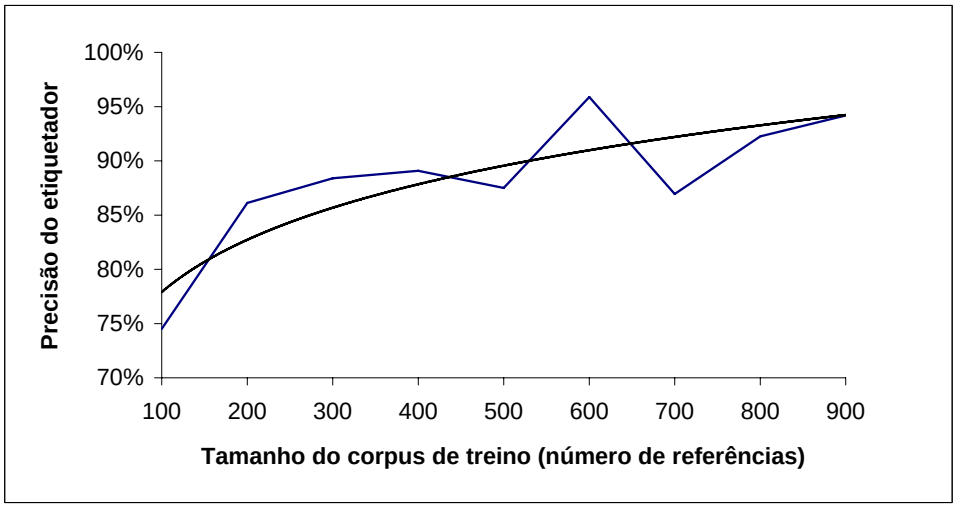


Figura 12: Gráfico comparativo de um processo interativo e iterativo de etiquetagem manual

4.4 Descrição dos Experimentos

Concluída a etapa de pré-processamento, o corpus para treinamento e teste semi-automaticamente etiquetados ficaram com 24928 sentenças (referências) e 1134986 palavras. Para avaliar o comportamento do etiquetador TBL frente à tarefa de etiquetagem, foram realizados dois experimentos: primeiro utilizando a porção etiquetada manualmente (947 sentenças) e segundo utilizando o corpus completo (24928 sentenças). Para o primeiro experimento, foi utilizado o método de avaliação *Cross-validation*[15] com dez partições, dessa forma foram criados dez corpus de treino e dez de teste, totalizando vinte novos corpus, a partir do corpus original (manualmente etiquetado). O número de palavras e de referências para essa divisão do corpus são apresentadas na Tabela 4.

[15]Este método é também conhecido como *k-fold Cross-validation* sendo k o número de partições geradas aleatoriamente a partir da amostra de exemplos para treinar e testar o sistema, sendo que a amostra de exemplos é dividida em k partições mutuamente exclusivas. A cada iteração uma partição diferente é utilizada para testar o sistema e todas as outras k-1 partições são utilizadas para treinar o sistema. A taxa de erro é a média das taxas de erro calculadas dadas as diversas partições.

Corpus	1	2	3	4	5	6	7	8	9	10
Treino (pal)	31262	31465	30996	31087	30814	30115	31464	31101	30599	30553
Treino (refs)	853	866	855	853	849	835	865	859	847	841
Teste (pal)	3122	2919	3388	3297	3570	4269	2920	3283	3785	3831
Teste (refs)	94	81	92	94	98	112	82	88	100	106

Tabela 4: Número de palavras e de referências em cada divisão do corpus

O código fonte do programa escrito em Perl que realiza essa divisão do corpus, pode ser visto no Apêndice B. Os dez arquivos de treino (cp_treino1, cp_treino2, ..., cp_treino10) e os dez de teste (cp_teste1, cp_teste2, ..., cp_teste10) são criados em um diretório chamado "corpora".

Para o segundo experimento foi utilizado o método de avaliação *avaliar-e-testar* com uma divisão do corpus completo em 70% para treinar o etiquetador e 30% para avaliar a sua precisão. Nos experimentos foi utilizado um PC com a configuração Athlon XP 2400MHz com 256Mb memória RAM e sistema operacional Linux 2.4.18. A seguir são descritas informações, consideradas relevantes, relacionadas ao software etiquetador e aos experimentos realizados.

Etiquetador Baseado em Transformação (TBL)

O etiquetador TBL possui dois módulos: um módulo que gera regras não contextuais, para determinar o conjunto de etiquetas mais prováveis de palavras (conhecidas e desconhecidas), e outro módulo que gera regras contextuais para melhorar a precisão do etiquetador. Para palavras desconhecidas, como já comentado o etiquetador inicialmente utiliza uma heurística que consiste em etiquetar tais palavras como nome comum (N) ou nome próprio (NP), caso comecem com letra maiúscula. Neste trabalho, foi feita uma alteração no código fonte para o que etiquetador pudesse rotular todas as palavras desconhecidas como TITLE, visto que esta é a etiqueta mais freqüente do corpus formado pelas referências bibliográficas.

A árvore de diretórios utilizada nos experimentos é apresentada na Figura 13. O diretório "Bin_and_Data" contém os programas compilados e o arquivo de instruções para executar um experimento, ou seja, instruções para treino e teste do etiquetador. Os diretórios "Learner_Code" e "Utilities" contêm códigos escritos em Perl que são necessários para a tarefa de etiquetagem.

Antes de iniciarem os experimentos foram criados vários diretórios, um para cada divisão do corpus. Em um diretório chamado "corpora" estão todos os corpus de treino e de teste utilizados. Foi criado um Shell Script, chamado "criadir", para criar os diretórios com seus correspondentes corpus de treino e teste, por ex.: os arquivos cp_teste1 e cp_treino1 são copiados para o diretório corpus_1. Para executar os experimentos diretamente dos diretórios, foram criados *links* para o diretório Bin_and_Data e os corpus de treino e de teste foram renomeados, respectivamente, para TRAINING-CORPUS e TEST-CORPUS. O código correspondente ao Script "criadir" é apresentado a seguir.

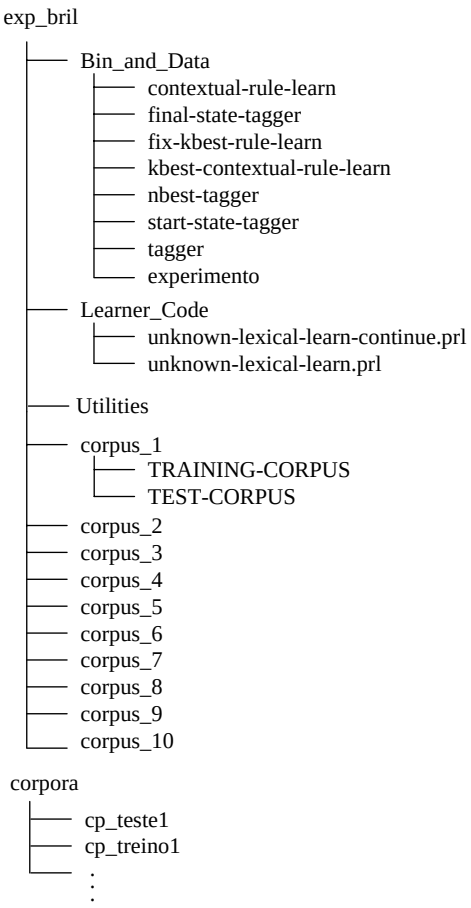


Figura 13: Estrutura de diretórios

```
for dir in 1 2 3 4 5 6 7 8 9 10
do
mkdir "corpus_"$dir

cp ../corpora/"cp_treino"$dir "corpus_"$dir/TRAINING-CORPUS
cp ../corpora/"cp_teste"$dir "corpus_"$dir/TEST-CORPUS

cd "corpus_"$dir/
ln -s ../Bin_and_Data/* ./
cd ..

done
```

Para treinar e testar o etiquetador foi criado um Shell Script, chamado “experimento”, que contém todas as instruções necessárias à realização dos experimentos. O Script, ilustrado logo abaixo, foi elaborado conforme as descrições da documentação que acompanha o software - Apêndice C.

```
1   cat TRAINING-CORPUS | perl ../Utilities/tagged-to-untagged.prl > UNTAGGED-CORPUS
2   cat TRAINING-CORPUS | perl ../Utilities/divide-in50-rand.prl TAGGED-CORPUS-2 TAGGED-CORPUS
3   cat TAGGED-CORPUS-2 | perl ../Utilities/tagged-to-untagged.prl > UNTAGGED-CORPUS-2
4   cat TEST-CORPUS | perl ../Utilities/tagged-to-untagged.prl > UNTAGGED-TEST-CORPUS
5
6   echo "1. wordlist-make.prl"
7   cat UNTAGGED-CORPUS | perl ../Utilities/wordlist-make.prl | sort +1 -rn | awk '{ print $1}' > BIGWORDLIST
8
9   echo "2. word-tag-count.prl"
```

```
10  cat TAGGED-CORPUS | perl ../Utilities/word-tag-count.prl | sort +2 -rn > SMALLWORDTAGLIST
11
12  echo "3. bigram-generate.prl"
13  cat UNTAGGED-CORPUS | perl ../Utilities/bigram-generate.prl | awk '{ print $1,$2}' > BIGBIGRAMLIST
14
15  echo "4. unknown-lexical-learn.prl"
16  perl ../Learner_Code/unknown-lexical-learn.prl BIGWORDLIST SMALLWORDTAGLIST BIGBIGRAMLIST
17  300 LEXRULEOUTFILE
18  perl ../Learner_Code/unknown-lexical-learn-continue.prl BIGWORDLIST SMALLWORDTAGLIST BIGBIGRAMLIST
19  300 NEWLEXRULEOUTFILE LEXRULEOUTFILE
20  cat < NEWLEXRULEOUTFILE >> LEXRULEOUTFILE
21
22  echo "5. make-restricted-lexicon.prl (TRAINING.LEXICON)"
23  cat TAGGED-CORPUS | perl  ../Utilities/make-restricted-lexicon.prl > TRAINING.LEXICON
24
25  echo "6. make-restricted-lexicon.prl (FINAL.LEXICON)"
26  cat TRAINING-CORPUS | perl  ../Utilities/make-restricted-lexicon.prl > FINAL.LEXICON
27
28  PATH=.:$PATH
29
30  echo "7. tagger"
31  ./tagger TRAINING.LEXICON UNTAGGED-CORPUS-2 BIGBIGRAMLIST LEXRULEOUTFILE /dev/null -w BIGWORDLIST  -i
32  DUMMY-TAGGED-CORPUS > /dev/null
33
34  echo "8. contextual-rule-learn"
35  ./contextual-rule-learn TAGGED-CORPUS-2 DUMMY-TAGGED-CORPUS CONTEXT-RULEFILE TRAINING.LEXICON
36
37  echo "9. Testing.."
38  ./tagger FINAL.LEXICON UNTAGGED-TEST-CORPUS /dev/null LEXRULEOUTFILE CONTEXT-RULEFILE > TAGGED-TEST-CORPUS
```

O processo de aprendizagem corresponde a praticamente todo o Script, da linha 1 até a linha 35. O produto da fase de treino são três arquivos: FINAL.LEXICON, LEXRULE-OUTFILE e CONTEXT-RULEFILE. O arquivo FINAL.LEXICON exibe cada palavra do corpus de treino (TRAINING-CORPUS) seguida pela etiqueta mais provável e por outras etiquetas sem uma ordem específica. O arquivo TRAINING-CORPUS foi dividido aleatoriamente em duas partes (linha 2), onde uma parte é utilizada na construção do arquivo LEXRULEOUTFILE e a outra na construção do arquivo CONTEXT-RULEFILE.

O arquivo LEXRULEOUTFILE gera regras para o aprendizado de palavras desconhecidas [16]. Este arquivo é gerado pelas instruções das linhas 15 a 20, o valor 300 significa que apenas serão usados bigramas em que pelo menos uma das palavras for uma das 300 mais freqüentes. O arquivo contém regras contextuais que servem para melhorar a precisão do etiquetador. A instrução da linha 38 é utilizada para testar o etiquetador treinado, ou seja, etiquetar o corpus de teste sem as etiquetas. Dessa maneira, como produto final tem-se dois arquivos, um etiquetado manualmente e outro etiquetado pelo software. Para gerar esses dois arquivos nos diretórios do corpus, basta executar o seguinte Script.

```
for dir in 1 2 3 4 5 6 7 8 9 10
do
cd corpus_$dir
./experimento
cd ..
done
```

[16] Palavras que apenas estão presentes no corpus de teste.

Concluída as etapas descritas acima, obteve-se, em cada diretório, o corpus de teste etiquetado manualmente - "TEST-CORPUS" -, o mesmo corpus etiquetado pelo software - "TAGGED-TEST-CORPUS", os arquivos gerados na fase de treino e os *links* para os programas. O objetivo dessa segunda fase é avaliar o desempenho do etiquetador. Para determinar a precisão global e por etiqueta do software foi feita uma comparação entre os arquivos "TEST-CORPUS" e "TAGGED-TEST-CORPUS" de cada diretório, através do arquivo de instruções apresentado a seguir.

```
for dir in 1 2 3 4 5 6 7 8 9 10
do

1   echo >>LIST-OF-ERRORS; echo "***Treino "$dir >>LIST-OF-ERRORS
2   echo >>FREQ-OF-ERRORS; echo "***Treino "$dir >>FREQ-OF-ERRORS
3   echo >>FREQ-OF-TAGS; echo "***Treino "$dir >>FREQ-OF-TAGS
4
5   cd corpus"_"$dir
6   ../evaluate TAGGED-TEST-CORPUS TEST-CORPUS ../TAGS-ERRORS-TMP ../TAGS-FREQ-TMP >../SUMMARY-TMP
7   cd ..
8
9   cat SUMMARY-TMP | grep erradas >>FREQ-OF-ERRORS
10  cat SUMMARY-TMP | grep palavras >>FREQ-OF-TAGS
11
12  cat TAGS-ERRORS-TMP | sort >TAGS-ERRORS-TMP2
13  cat TAGS-ERRORS-TMP2 >>LIST-OF-ERRORS
14  ./count_errors <TAGS-ERRORS-TMP2 >>FREQ-OF-ERRORS
15  ./global_errors.exe < FREQ-OF-ERRORS > FREQ-GLOBAL-ERRORS
16
17  cat TAGS-FREQ-TMP | sort +1nr >>FREQ-OF-TAGS
18  rm TAGS-ERRORS-TMP TAGS-ERRORS-TMP2 TAGS-FREQ-TMP SUMMARY-TMP
19
20  done
21
22  ./calc_val_rel <FREQ-OF-TAGS | sort +0 -1 +1 -2n >FREQ-VAL-REL-TMP
23  ./table 10 <FREQ-VAL-REL-TMP >TABLE-FREQ-TAGS
24
25  ./calc_val_rel <FREQ-OF-ERRORS | sort +0 -1 +1 -2n >ERROS-VAL-REL-TMP
26  ./table 10 <ERROS-VAL-REL-TMP >TABLE-FREQ-ERRORS
27
28  ./calc_val_rel <FREQ-GLOBAL-ERRORS | sort +0 -1 +1 -2n >GLOBAL-ERROS-VAL-REL-TMP
29  ./table 10 <GLOBAL-ERROS-VAL-REL-TMP >TABLE-FREQ-GLOBAL-ERRORS
30
31  ./calc_val_mod <FREQ-GLOBAL-ERRORS | sort +0 -1 +1 -2n >ERROS-VAL-RES-TMP
32  cat FREQ-VAL-REL-TMP >> ERROS-VAL-RES-TMP
33  cat ERROS-VAL-RES-TMP | sort +0 -1 +1 -2n >ERROS-VAL-RES-SORT
34  ./table 10 <ERROS-VAL-RES-SORT >TABLE-TAG-PRECISION
35
36  rm FREQ-VAL-REL-TMP ERROS-VAL-REL-TMP GLOBAL-ERROS-VAL-REL-TMP
37  rm ERROS-VAL-RES-TMP ERROS-VAL-RES-SORT
```

As informações obtidas para cada corpus são concatenadas em um arquivo e separadas pela string "***Treino número-corpus". Os principais arquivos resultantes desta fase são: FREQ-OF-TAGS, LIST-OF-ERRORS, FREQ-OF-ERRORS e FREQ-GLOBAL-ERRORS. O arquivo FREQ-OF-TAGS contém a freqüência de cada etiqueta em cada corpus de teste. LIST-OF-ERRORS contém a lista de erros cometidos pelo software etiquetador em cada corpus de teste. A lista de erros cometidos apresenta a etiqueta correta,

a etiqueta colocada pelo software e a palavra que foi etiquetada. O arquivo FREQ-OF-ERRORS armazena a freqüência de cada erro cometido, ou seja, o número de vezes que uma etiqueta correta X foi substituída por uma etiqueta errada Y. O arquivo FREQ-GLOBAL-ERRORS armazena a freqüência global de cada erro, ou seja, o número de vezes que uma etiqueta correta foi substituída por alguma etiqueta errada. As instruções da linha 22 a 37 são para gerar arquivos em forma de tabela utilizando as informações obtidas anteriormente. O código fonte dos programas utilizados (evaluate, count_errors, global_errors, etc.) escritos em linguagem C, são apresentados no Apêndice B.

4.5 Resultados Obtidos

4.5.1 Primeiro Experimento

Os resultados obtidos com o primeiro experimento são descritos a seguir. O número de palavras, número de etiquetas erradas e porcentagem de etiquetas corretas e erradas, para cada corpus de treino e teste são apresentados na Tabela 5:

Corpus com 947 referências (34384 palavras)										
Experimento	1	2	3	4	5	6	7	8	9	10
Corpus de treino (pal)	31262	31465	30996	31087	30814	30115	31464	31101	30599	30553
Corpus de teste (pal)	3122	2919	3388	3297	3570	4269	2920	3283	3785	3831
Etiquetas erradas (pal)	190	173	206	221	253	282	201	254	208	205
Etiquetas erradas (%)	6,09%	5,93%	6,08%	6,70%	7,09%	6,61%	6,88%	7,74%	5,50%	5,35%
Etiquetas corretas (%)	93,91%	94,07%	93,92%	93,30%	92,91%	93,39%	93,12%	92,26%	94,51%	94,65%

Tabela 5: Número de palavras e a taxa de acerto para cada divisão do corpus manualmente etiquetado

O número médio de palavras nos corpus de treino e teste, a precisão global (média das taxas de acerto) do etiquetador e o desvio padrão, calculado a partir da Fórmula 1, são apresentados na Tabela 6.

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2} \tag{1}$$

Número médio de palavras no corpus de treino	30946
Número médio de palavras no corpus de teste	3438
Precisão global do etiquetador	93,604%
Desvio padrão da amostra	0,7431%

Tabela 6: Desvio padrão e precisão global do etiquetador

A lista com o número de vezes que cada etiqueta foi substituída por outra foi unida com a lista de freqüência de etiquetas. Assim, foi possível determinar a taxa de erro para cada etiqueta em cada experimento, dividindo-se o número de erros pelo número total de vezes que a etiqueta aparece. Dessa forma foi possível determinar quais são as etiquetas problemáticas, ou seja, quais etiquetas mais levam o etiquetador a cometer erros. A fórmula para medir o impacto do erro causado por uma etiqueta (t) na taxa global de erro

do etiquetador é dada pela Equação 2, onde PM e FM são, respectivamente, a precisão média e freqüência média de uma etiqueta em porcentagem. Tal fórmula corresponde ao percentual de erros na etiquetagem do corpus, relacionado com a etiqueta *t*. Considera-se, neste trabalho, como problemáticas as etiquetas para as quais o fator de impacto é superior a 0,4%.

$$impacto\_no\_erro_t = (1 - \frac{PM_t}{100})FM_t \quad (2)$$

A precisão média por etiquetas, juntamente com as suas freqüências médias, e o fator de impacto podem ser vistos na Tabela 7.

Em geral, quanto mais rara a etiqueta, maior a taxa de erro. Os etiquetadores normalmente não cometem erros quando a tarefa é etiquetar sinais de pontuação, pois em geral apresentam sempre a mesma representação. Entretanto, analisando a tabela nota-se que o etiquetador cometeu alguns erros com sinais de pontuação. Este fato ocorreu por dois motivos:

- o sinal de pontuação apareceu uma única vez no corpus;
- alguns sinais de pontuação não foram corretamente padronizados

É importante observar que as etiquetas que possuem uma pequena taxa de acerto, etiquetas com precisão inferior a 80%, apresentam uma soma de freqüências de apenas 7,7%, ou seja, a possibilidade de alguma dessas etiquetas aparecerem em alguma referência é pequena. Isso obviamente contribui para um grande índice geral de acerto. Após uma análise dos erros cometidos pelo etiquetador verificou-se que o mesmo cometeu mais erros substituindo a etiqueta BOOKTITLE por TITLE e vice versa, sendo que as duas possuem um alto fator de impacto e uma precisão alta. Acredita-se algumas etiquetas, tais como CROSSREF e INSTITUTION, apresentem uma pequena precisão, principalmente, por possuirem uma baixa freqüência, mas também por serem usadas em situações específicas.

4.5.2 Segundo Experimento

Para o segundo experimento, o número de palavras, etiquetas erradas e a porcentagem de etiquetas corretas e erradas, para o conjunto de treino e teste são apresentados na Tabela 8. Com comentado anteriormente, a divisão utilizada neste experimento foi de 70% para treinamento e 30% para teste. O corpus de teste, gerado aleatoriamente, apresenta referências bibliográficas de vários estilos contidos no corpus completo.

Com a disponibilidade de um corpus maior, o etiquetador neste experimento alcançou uma precisão global de 96,9%. Se comparado com o experimento anterior, isso corresponde a um acréscimo de 3,3% em sua precisão.

Na Tabela 9 são apresentados os tempos de processamento do etiquetador TBL para o treinamento e a etiquetagem do corpus de teste. Embora o tempo de treinamento de regras contextuais tenha levando 38 horas, este treinamento apenas será realizado quando for necessário à FIP conhecer novos padrões de referências.

Etiqueta	Precisão Média	Freqüência Média	Impacto no Erro
BOOKTITLE	88,659%	11,024%	1,2503%
TITLE	95,006%	20,803%	1,0389%
AUTHOR	95,650%	13,917%	0,6054%
JOURNAL	85,090%	3,138%	0,4679%
INITPAGE	79,668%	1,559%	0,3169%
FINALPAGE	79,748%	1,559%	0,3157%
ADDRESS	84,911%	2,069%	0,3121%
NOTE	64,902%	0,876%	0,3075%
NUMBER	70,987%	0,762%	0,2211%
PUBLISHER	86,598%	1,611%	0,2159%
VOLUME	74,884%	0,821%	0,2063%
EDITOR	75,961%	0,754%	0,1814%
SERIES	60,389%	0,443%	0,1755%
INSTITUTION	41,859%	0,289%	0,1680%
CROSSREF	46,738%	0,208%	0,1107%
SCHOOL	70,316%	0,370%	0,1099%
URL	89,859%	1,023%	0,1037%
TYPE	86,630%	0,470%	0,0629%
INDICATOR	97,403%	2,333%	0,0606%
MONTH	96,925%	0,580%	0,0178%
CHAPTER	60,000%	0,030%	0,0119%
PAGES	50,000%	0,018%	0,0089%
URLACCESSDATE	86,667%	0,063%	0,0083%
DAYS	70,000%	0,018%	0,0054%
EDITION	90,000%	0,036%	0,0036%
’	80,000%	0,006%	0,0011%
!	90,000%	0,003%	0,0003%
=	90,000%	0,003%	0,0003%
‘	90,000%	0,003%	0,0003%
´	100,000%	0,185%	0,0000%
-	100,000%	4,101%	0,0000%
”	100,000%	0,358%	0,0000%
&	100,000%	0,062%	0,0000%
(	100,000%	1,132%	0,0000%
)	100,000%	1,135%	0,0000%
,	100,000%	8,903%	0,0000%
.	100,000%	14,137%	0,0000%
:	100,000%	1,364%	0,0000%
;	100,000%	0,195%	0,0000%
?	100,000%	0,033%	0,0000%
[	100,000%	0,085%	0,0000%
\	100,000%	0,003%	0,0000%
]	100,000%	0,085%	0,0000%
˜	100,000%	0,013%	0,0000%
<	100,000%	0,021%	0,0000%
>	100,000%	0,021%	0,0000%
BARRA	100,000%	0,740%	0,0000%
YEAR	100,000%	2,639%	0,0000%

Tabela 7: Freqüência média e precisão por etiquetas do etiquetador

Corpus com 24928 referências (1134986 palavras)	
Número de palavras no corpus de treino	797652
Número de palavras no corpus de teste	337334
Número de etiquetas erradas	10362
Porcentagem de etiquetas erradas	3,1%
Porcentagem de etiquetas corretas	96,9%

Tabela 8: Número de palavras e a taxa de acerto para o conjunto de treino e de teste do corpus

	Tempo de processamento
Regras para palavras desconhecidas	5 horas, 39 minutos e 25 segundos
Regras contextuais	38 horas, 36 minutos e 10 segundos
Tempo de etiquetagem	5 segundos

Tabela 9: Tempo de treinamento e etiquetagem - TBL

4.6 Extração das Informações

Concluída a etapa de etiquetagem das referências, pode-se iniciar a extração de suas informações. A extração de elementos de referências bibliográficas, basicamente, consiste em concatenar informações de etiquetas correspondentes. Considera-se neste trabalho como etiquetas correspondentes, as etiquetas cujos os nomes são iguais ou sinais de pontuação que possuam algum significado para a informação a ser extraída, por exemplo o ponto que abrevia o nome de um autor ou o hífen que separa a página inicial da final de uma referência.

O produto final do processo de extração é um documento XML que contém as informações extraídas para cada referência. O código fonte do programa "extraction", escrito em Perl, que realiza a extração das informações é descrito no Apêndice B. Para uma melhor compreensão é apresentado a seguir, um exemplo da extração de informações de uma referência etiquetada pelo TBL. Caso o programa receba como entrada a referência abaixo

Achermann/AUTHOR ,/, F/AUTHOR ./. and/AUTHOR Nierstrasz/AUTHOR ,/, O/AUTHOR ./. (/(2000c/YEAR)/) ./. Explicit/TITLE Namespaces/TITLE ./. In/INDICATOR Gutknecht/EDITOR ,/, J/EDITOR ./. and/EDITOR Weck/EDITOR ,/, W/EDITOR ./. ,/, editors/INDICATOR ,/, Modular/BOOKTITLE Programming/BOOKTITLE Languages/BOOKTITLE ,/, volume/INDICATOR 1897/VOLUME of/SERIES LNCS/SERIES ,/, pages/INDICATOR 77/PAGES -/- 89/PAGES ,/, Zurich/ADDRESS ,/, Switzerland/ADDRESS ./. Springer/PUBLISHER -/- Verlag/PUBLISHER ./. URL/INDICATOR http/URL :/: $b/BARRA $b/BARRA www/URL ./. iam/URL ./. unibe/URL ./. ch/URL $b/BARRA ~/~ scg/URL $b/BARRA Archive/URL $b/BARRA Papers/URL $b/BARRA Ache00bExplicitNamespaces/URL ./. pdf/URL

irá combinar e extrair as suas informações, conforme descrito no seguinte código XML

<ref>

<author>F. Achermann</author>

<author>O. Nierstrasz</author>
<year>2000</year>
<title>Explicit Namespaces</title>
<editor>J. Gutknecht</editor>
<editor>W. Weck</editor>
<booktitle>Modular Programming Languages</booktitle>
<volume>1897</volume>
<series>of LNCS</series>
<pages>77-89</pages>
<address>Zurich, Switzerland</address>
<publisher>Springer-Verlag</publisher>
<url>http://www.iam.unibe.ch/~scg/Archive/Papers/Ache00bExplicitNamespaces.pdf</url>
</ref>

A partir do exemplo acima, é possível observar algumas características do processo de extração: quando a referência possuir dois ou mais autores (ou editores), estes são extraídos em separado; os sinais de pontuação que estiverem entre o final de uma informação e início de outra são removidos; os termos da etiqueta INDICATOR são removidos, pois tais termos apenas indicam a presença de informações.

A adoção da XML (*eXtensible Markup Language*) neste trabalho, se deve pelos seguintes motivos:

1. a linguagem é um padrão aberto, flexível e facilmente expandível;
2. possibilita a transferência e manipulação de dados através da internet de modo fácil, rápido e consistente, de tal forma que qualquer tipo de aplicação, independentemente da plataforma, sistema operacional, ou linguagem em que foi construída consiga manuseá-los;
3. representa os dados de maneira hierárquica e organizada;
4. separa o conteúdo da apresentação, o que possibilita integrar e manipular dados de diversas fontes;
5. permite realizar buscas eficientes, pois os dados em um documento XML podem ser unicamente "etiquetados";
6. disponibilidade de várias ferramentas na Web que realizam a edição e análise (formatação/gramática) de documentos XML.

5 Considerações Finais

A proposta para extrair automaticamente informações de referências bibliográficas de artigos científicos, baseia-se no mapeamento do problema de *part-of-speech* (POS) *tagging*

ao problema de extração de informação. Este mapeamento, consiste em etiquetar todos os termos das referências selecionando alguma etiqueta de um conjunto de possíveis etiquetas, por exemplo AUTOR, TÍTULO, REVISTA, EVENTO, PÁGINAS e ANO, e, finalmente, combinar e extrair as informações (etiquetas correspondentes) das referências.

Nos experimentos foi utilizado o etiquetador TBL, desenvolvido por Eric Brill, que possui uma alta precisão de etiquetagem morfossintática. O etiquetador apresenta dois módulos de etiquetagem: um primeiro módulo que gera regras, não contextuais, para determinar o conjunto de etiquetas mais prováveis de palavras (conhecidas e desconhecidas) e outro módulo que gera regras contextuais para melhorar a precisão da etiquetagem.

Para avaliar o etiquetador TBL frente à tarefa de etiquetagem de referências bibliográficas foi utilizado um corpus com mais de 1 milhão de palavras. O corpus possui referências da área de Computação e abrange os estilos bibliográficos Plain, Alpha, Abbrv, Apalike e Chicago. O corpus é composto por uma pequena parcela etiquetada manualmente e por outra semi-automaticamente etiquetada utilizando informações contidas em arquivos BibTEX. Os resultados obtidos são bastante promissores, pois o etiquetador treinado rotula com alta precisão referências bibliográficas de vários estilos, e, conseqüentemente, eleva a confiança no processo de extração das informações.

A FIP tagset

A seguir são apresentadas as etiquetas do FIP tagset e o que cada uma abrange de informação, no contexto das referências bibliográficas. Na tabela abaixo, *documento* representa todos os tipos de referências vistos no Apêndice **??**.

N	**Etiqueta**	**Informação abrangida**
1	ADDRESS	endereço completo, ou apenas a cidade, de uma editora
2	AUTHOR	autor(es) de um *documento*
3	BOOKTITLE	título de um livro ou evento
4	CHAPTER	número do capítulo de um livro
5	EDITION	edição de um livro
6	EDITOR	editor(es) de um livro ou evento
7	INSTITUTION	instituição no qual um relatório foi escrito
8	ISBN	numeração internacional para livros
9	ISSN	numeração internacional para publicações periódicas
10	JOURNAL	revista que publicou um artigo
11	MONTH	mês de publicação
12	NOTE	alguma informação adicional
13	NUMBER	número de uma revista, livro ou relatório
14	ORGANIZATION	órgão patrocinador de um evento
15	INDICATOR	pp, pages, In, Inc, Eds, Volume, Vol, No, editor(s), etc.
16	INITPAGE	página inicial do *documento*
17	FINALPAGE	página final do *documento*
18	PUBLISHER	editora
19	SCHOOL	universidade em que a monografia foi escrita
20	SERIES	série de um livro ou evento
21	TYPE	alguma informação que informe o tipo do *documento*
22	TITLE	título de um *documento* ou capítulo de um livro
23	URL	link para um *documento* (pdf, ps)
24	URLACCESSDATE	data em que a url foi acessada
25	VOLUME	volume de uma revista ou livro
26	YEAR	ano de publicação do *documento*
27	PAGES	número de páginas do *documento*
28	DAYS	dias em que aconteceu o evento
29	CROSSREF	referência cruzada
30	.	.
31	:	:
32	;	;
33	,	,
34	[	[
35	]	]
36	(	(
37	)	)
38	{	{
39	}	}
40	!	!
41	?	?
42	-	-
43	&	&
44	/	/
45	\	\
46	’	’
47	‘	‘
48	”	”
49	“	“

B Código Fonte dos Programas Desenvolvidos

```
#################################################################################################################
#               Information Extraction: Pré-processamento das Referências Bibliográficas
#                                     Autor: Alberto Cáceres Álvarez
# ---------fixrefs.prl---------          ICMC - USP - São Carlos          ---------fixrefs.prl---------
#                                        2o. Semestre de 2005
#
# Este programa assume que o conjunto de referências bibliográficas localiza-se dentro de um subdiretório
# chamado "refs", e que as referências estejam divididas em vários arquivos texto.
#################################################################################################################

main();

#################################################################################################################
# Identifica se a linha, que nao foi coberta pelos padrões anteriores, pode ser início de uma referência
#################################################################################################################
sub is_InitRef($)
{
    $_ = shift;
    my $char = substr($_, 0, 1);

    #se a string começar com um char minusculo
    if ( ($char =~ /\D/) && ($char =~ /\w/) && (lc $char eq $char) )
    { return 0; }

    if (/[\w\,]\s\d\d\d\d\.?$/)
    { return 0; }

    if ( (/\.?\s\(?19\d\d\D?\)?\.?\s?/)          ||       #existe um ano com formato 19XX
         (/\.?\s\(?20\d\d\D?\)?\.?\s?/)          ||       #existe um ano com formato 20XX
         (/\.?\s\d\d\d\d\D?\\d\d\d\d\D?\.?\s?/) ||     #existe um ano com formato XXXXYYYY.
         (/\w+\,\s(\w+\s)?\w\./ && /\D+\,\s(\w+\s)?\D\./) #algo, C.
       )
    { return 1;  }
    else
    { return 0;  }
}

#################################################################################################################
# Faz a 'limpeza' da string, removendo erros (espaço em branco, ponto) no início e no final
#################################################################################################################
sub alltrim($) {

    $_ = shift;

    s/^\s*//;
    s/\s*$//;
    s/^\.+//;
    s/^\]+//;
    s/^\,+//;
    s/\cM//ig;
    return $_;
}

#################################################################################################################
# Procedimento que identifica o modelo (padrão) da referência, limpa alguns erros (alltrim) e ao final
# retorna a referência completa
#################################################################################################################
sub get_Ref
{
```

```
my($InputLine, $string);
$InputLine = $_[0];
$string = "";

# Se a linha inicia com [num]. "alnum ou (" - 1º modelo
if ($InputLine =~ /^\[\d\d?\d?\]\.?\s?[\w\(]/)
{
   $InputLine = alltrim($InputLine);
   do{
          if($string =~ /(\D)[-||||-]$/){
             if($1 eq '-'|| $1 eq '' || $1 eq '-'){
                $string = $string . " " . $InputLine;
             }
             else{
                $string =~ s/(.)(-||||-)$/$1/;
                $string = $string . $InputLine;
             }
          }
          else{
             $string = $string . " " . $InputLine;}
          do{
              $InputLine = <INPUTFILE>;
              if(not $InputLine){$EndOfFile = 1;}
              $InputLine = alltrim($InputLine);
              if(length($InputLine) <= 4 && $InputLine =~ /\d\d?\d?\d?/ && (not $InputLine =~ /\./))
              {
                 $string =~ s/\s$//;
              }
          }while(length($InputLine) <= 4 && $InputLine =~ /\d\d?\d?\d?/ && (not $InputLine =~ /\./));

          if( $string =~ / of$/ || $string =~ / to$/ || $string =~ / in$/  ||
              $string =~ / at$/ || $string =~ / by$/ || $string =~ / for$/ ||
              $string =~ / from$/ || $string =~ / on$/ || $string =~ / a$/ ||
              $string =~ / an$/ || $string =~ / the$/ || $string =~ / and$/||
              $string =~ /,$/ || $string =~ /:$/ || $string =~ /;$/ ||
              $string =~ / pages$/ || $string =~ / vol\.$/ || $string =~ / pp\.$/ ||
              $string =~ / Eds\.$/ || $string =~ / In\.?$/)
          { }
          elsif ($InputLine =~ /^\[\d\d?\d?\]\.?\s?[\w\(]/)
          {
              return ($InputLine, $string);
          }
   } while (not $EndOfFile) ;
}
# Se a linha inicia com [XXXX, 1992] ...) - 2º modelo
elsif($InputLine =~ /^\[.+\w\w\d?\d?\D?\]\s/)
{
   $InputLine = alltrim($InputLine);
   do
   {
          if($string =~ /(\D)[-||||-]$/){
             if($1 eq '-'|| $1 eq '' || $1 eq '-'){
                $string = $string . " " . $InputLine;
             }
             else{
                $string =~ s/(.)(-||||-)$/$1/;
                $string = $string . $InputLine;
             }
          }
          else{
             $string = $string . " " . $InputLine;}
          do{
```

```
            $InputLine = <INPUTFILE>;
            if(not $InputLine){$EndOfFile = 1;}
            $InputLine = alltrim($InputLine);
            if(length($InputLine) <= 4 && $InputLine =~ /\d\d?\d?\d?/ && (not $InputLine =~ /\./))
            {
               $string =~ s/\s$//;
            }
         }while(length($InputLine) <= 4 && $InputLine =~ /\d\d?\d?\d?/ && (not $InputLine =~ /\./));

         if( $string =~ / of$/ || $string =~ / to$/ || $string =~ / in$/  ||
            $string =~ / at$/ || $string =~ / by$/ || $string =~ / for$/ ||
            $string =~ / from$/ || $string =~ / on$/ || $string =~ / a$/ ||
            $string =~ / an$/ || $string =~ / the$/ || $string =~ / and$/||
            $string =~ /,$/ || $string =~ /:$/ || $string =~ /;$/ ||
            $string =~ / pages$/ || $string =~ / vol\.$/ || $string =~ / pp\.$/ ||
            $string =~ / Eds\.$/ || $string =~ / In\.?$/)
         { }
         elsif ($InputLine =~ /^\[.+\w\w\d?\d?\D?\]\s/)
         {
            return ($InputLine, $string);
         }
   } while (not $EndOfFile) ;
}
# Se a linha inicia com (YYYYY, 1992) ... - 3º modelo
elsif($InputLine =~ /^\(\w+\,?\s?\d*\D?\)\s?/)
{
   $InputLine = alltrim($InputLine);
   do
   {
         if($string =~ /(\D)[-||||-]$/){
            if($1 eq '-'|| $1 eq '' || $1 eq '-'){
               $string = $string . " " . $InputLine;
            }
            else{
               $string =~ s/(.)(-||||-)$/$1/;
               $string = $string . $InputLine;
            }
         }
         else{
            $string = $string . " " . $InputLine;}
         do{
            $InputLine = <INPUTFILE>;
            if(not $InputLine){$EndOfFile = 1;}
            $InputLine = alltrim($InputLine);
            if(length($InputLine) <= 4 && $InputLine =~ /\d\d?\d?\d?/ && (not $InputLine =~ /\./))
            {
               $string =~ s/\s$//;
            }
         }while(length($InputLine) <= 4 && $InputLine =~ /\d\d?\d?\d?/ && (not $InputLine =~ /\./));

         if( $string =~ / of$/ || $string =~ / to$/ || $string =~ / in$/  ||
            $string =~ / at$/ || $string =~ / by$/ || $string =~ / for$/ ||
            $string =~ / from$/ || $string =~ / on$/ || $string =~ / a$/ ||
            $string =~ / an$/ || $string =~ / the$/ || $string =~ / and$/||
            $string =~ /,$/ || $string =~ /:$/ || $string =~ /;$/ ||
            $string =~ / pages$/ || $string =~ / vol\.$/ || $string =~ / pp\.$/ ||
            $string =~ / Eds\.$/ || $string =~ / In\.?$/)
         { }
         elsif ($InputLine =~ /^\(\w+\,?\s?\d*\D?\)\s?/)
         {
            return ($InputLine, $string);
         }
```

```
   } while (not $EndOfFile) ;
}
# Se a linha inicia com num. (Init) - 4º modelo
elsif($InputLine =~ /^\d+\s?\.?\s/)
{
   $InputLine = alltrim($InputLine);
   do
   {
          if($string =~ /(\D)[-||||-]$/){
             if($1 eq '-'|| $1 eq '' || $1 eq '-'){
                $string = $string . " " . $InputLine;
             }
             else{
                $string =~ s/(.)(-||||-)$/$1/;
                $string = $string . $InputLine;
             }
          }
          else{
             $string = $string . " " . $InputLine;}
          do{
              $InputLine = <INPUTFILE>;
              if(not $InputLine){$EndOfFile = 1;}
              $InputLine = alltrim($InputLine);
              if(length($InputLine) <= 4 && $InputLine =~ /\d\d?\d?\d?/ && (not $InputLine =~ /\./))
              {
                 $string =~ s/\s$//;
              }
          }while(length($InputLine) <= 4 && $InputLine =~ /\d\d?\d?\d?/ && (not $InputLine =~ /\./));

          if( $string =~ / of$/ || $string =~ / to$/ || $string =~ / in$/  ||
              $string =~ / at$/ || $string =~ / by$/ || $string =~ / for$/ ||
              $string =~ / from$/ || $string =~ / on$/ || $string =~ / a$/ ||
              $string =~ / an$/ || $string =~ / the$/ || $string =~ / and$/||
              $string =~ /,$/ || $string =~ /:$/ || $string =~ /;$/ ||
              $string =~ / pages$/ || $string =~ / vol\.$/ || $string =~ / pp\.$/ ||
              $string =~ / Eds\.$/ || $string =~ / In\.?$/)
          { }
          elsif ($InputLine =~ /^\d+\s?\.?\s/)
          {
              return ($InputLine, $string);
          }
   } while (not $EndOfFile) ;
}
else #a principio nao é possivel identificar o padrão da referência - heurísticas
{
   $InputLine = alltrim($InputLine);
   my $endchar;
   do
   {
          if (not $string){
           $string = $string . $InputLine;
          }
          elsif(length($string) <= 41 && not($string =~ /http:/) && not($string =~ /ftp:/))#tamanho da string
          {
            $string = $string . " " . $InputLine;
          }
          elsif ($string =~ / of$/ || $string =~ / to$/ || $string =~ / in$/  || #lista de preposições em ingles
                 $string =~ / at$/ || $string =~ / by$/ || $string =~ / for$/ ||
                 $string =~ / from$/ || $string =~ / on$/ || $string =~ / a$/ ||
                 $string =~ / an$/ || $string =~ / the$/ || $string =~ / and$/||
                 $string =~ /,$/ || $string =~ /:$/ || $string =~ /;$/)
          {
```

```
    $string = $string . " " . $InputLine;
}
elsif ($string =~ / Conference$/ || $string =~ / San$/ || $string =~ / pages$/ || #heuristicas
       $string =~ / &$/ || $string =~ / Technical$/  || $string =~ /\s\d\d?th$/ ||
       $string =~ / vol\.$/ || $string =~ / pp\.$/ || $string =~ / Eds\.$/ ||
       $string =~ / In\.?$/)
{
    $string = $string . " " . $InputLine;
}
elsif ($string =~ /(\D)[-||||-]$/){
   if($1 eq '-'|| $1 eq ''|| $1 eq '-'){
      $string = $string . $InputLine;
   }
   else{
      $string =~ s/(.)(-||||-)$/$1/;
      $string = $string . $InputLine;
   }
}
elsif ($InputLine =~ /^of / || $InputLine =~ /^to / || $InputLine =~ /^in /  ||
       $InputLine =~ /^at / || $InputLine =~ /^by / || $InputLine =~ /^for / ||
       $InputLine =~ /^from / || $InputLine =~ /^on / || $InputLine =~ /^a / ||
       $InputLine =~ /^an / || $InputLine =~ /^the / || $InputLine =~ /^and /||
       $InputLine =~ /^, / || $InputLine =~ /^: / || $InputLine =~ /^;/)
{
    $string = $string . " " . $InputLine;
}
elsif ($InputLine =~ /^Conference / || $InputLine =~ /^San / || $InputLine =~ /^pages / || #heuristicas
       $InputLine =~ /^& / || $InputLine =~ /^Technical /  || $InputLine =~ /^\d\d?th / ||
       $InputLine =~ /^vol\. / || $InputLine =~ /^pp\. / || $InputLine =~ /^Eds\. / ||
       $InputLine =~ /^In\.? / || $InputLine =~ /^System\.? / || $InputLine =~ /^Computer / ||
       $InputLine =~ /^Knowledge /)
{
    $string = $string . " " . $InputLine;
}
elsif(not($string =~ /\.\s.\.$/) && $string =~ /\s.\.$/) #capturando o estilo terminando: C.
{
  $endchar = substr($string, -2, 1);
  if ( ($endchar =~ /\D/) && ($endchar =~ /\w/) && (uc $endchar eq $endchar) )
  {
     $string = $string . " " . $InputLine;
  }
}
elsif($string =~ /and\s\w+$/)#terminando em: and Kotomami
{
 $string = $string . " " . $InputLine;
}
elsif(not($string =~ /\d/) && not($string =~ /http:/) &&   #se a string pequena nao tiver digito
   not($string =~ /ftp:/) && length($string) <= 60)
{
 $string = $string . " " . $InputLine;
}
elsif ($string =~ / Berlin\.$/ || $string =~ / CO\.$/ ||
       $string =~ / guide\.$/  || $string =~ /[\w\,]\s\d\d\d\d\.?$/) #heurísticas
{
    return ($InputLine, $string);
}
elsif ($string =~ /\,\s\d\d\d\d\D?\.$/ ||
       $string =~ /\d\d?\d?\d?\-\-\d\d?\d?\d?\.$/ || is_InitRef($InputLine))
{
    return ($InputLine, $string);
}
#heurísticas
```

```
            elsif ($string =~ /ese\)\.$/ || $string =~ /Guide\.$/ || $string =~ /Springer\.$/ ||
                   $string =~ /Denmark\.$/ || $string =~ /Summarization\.$/ || $string =~ /Humanities\.$/)
            {
             return ($InputLine, $string);
            }
            else{
                $string = $string . " " . $InputLine;
            }

            if($InputLine && (length($InputLine) <= 8))
            {
                do{
                    $InputLine = <INPUTFILE>;
                    $InputLine = alltrim($InputLine);
                    if(length($InputLine) <= 4 && $InputLine =~ /\d\d?\d?\d?/ && (not $InputLine =~ /\./))
                    {
                       $string =~ s/\s$//;
                    }
                }while(length($InputLine) <= 4 && $InputLine =~ /\d\d?\d?\d?/ && (not $InputLine =~ /\./));

                return ($InputLine, $string);
            }
            do{
                $InputLine = <INPUTFILE>;
                if(not $InputLine){$EndOfFile = 1;}
                $InputLine = alltrim($InputLine);
                if(length($InputLine) <= 4 && $InputLine =~ /\d\d?\d?\d?/ && (not $InputLine =~ /\./))
                {
                   $string =~ s/\s$//;
                }
            }while(length($InputLine) <= 4 && $InputLine =~ /\d\d?\d?\d?/ && (not $InputLine =~ /\./));

     }while(not $EndOfFile);
  }
  return ($InputLine, $string);
}

#################################################################################################################
#  Main
#################################################################################################################
sub main
{
  my($InputLine, $string, $ID, $erros, @files, $file);

  $erros=0;

  # Le o subdiretorio refs e armazena todos os arquivos .TXT na lista @files
  opendir(DIR, "./refs/");
  while ($name = readdir(DIR))
  {
      next if $name !~ /^(.*)\.txt/;
      push (@files, $name);
  }
  close(DIR);

  if ($#files < 0)
  {
      print "Nenhum arquivo .TXT encontrado na pasta ./refs";
  }
  else
  {
    open(ERROR, ">erro.txt");
```

```
# Para cada um dos arquivos contidos na lista, o algoritmo faz o seguinte:
# 1 - Abre o próximo arquivo de referências
# 2 - Lê uma linha
#   2.1 - Identifica seu modelo
#   2.2 - Lê o restante da referência
#  2.3 - Corrige alguns problemas na referência
# 4 - Grava a referência completa no arquivo ALL
# 5 - Caso o tamanho da referência for maior que 450 (provavelmente com problema)
#   5.1 - Grava a referência no arquivo ERROR

open(ALL, ">./refs/all.out");

foreach $file (@files)
{
    open(INPUTFILE, "./refs/".$file);
    $EndOfFile = 0;
    $InputLine = <INPUTFILE>;

    # Enquanto o arquivo possuir dados para serem lidos...
    while (defined($InputLine))
    {
      # Desconsidera linhas que apenas possuam Literature, Bibliography, ...
      if(((lc $InputLine) =~ /iterature/ || (lc $InputLine) =~ /eference/ ||
         (lc $InputLine) =~ /ibliography/) && length($InputLine) <= 20)
      {
        $InputLine = "";
        $string = "";
      }
      else #Adquire uma referência completa
      {
            ($InputLine, $string) = &get_Ref($InputLine);
      }

      if($string ne "")
      {
             $string = alltrim ($string); #limpeza

             #correção de erros
             $string =~ s/  / /ig; #remoção de espaço duplo
             $string =~ s/\s([.,;:?!\\\/])/$1/ig;
             $string =~ s/’’/"/ig;
             $string =~ s/‘‘/"/ig;
             $string =~ s/‘/’/ig;
             $string =~ s/[\-\]/\-/ig;
             $string =~ s/\Æ/ffi/ig;
             $string =~ s/(\d)\{/$1--/ig;
             $string =~ s/\{/-/ig;
             $string =~ s/\sth\s/th /ig;
             $string =~ s/\’\ss/\’s/ig;

             #ajustes finais
             $string =~ s/^\d+\.?\s//;

             $string =~ s/^\[\w+[’-]?\w*\s?\w*\s?\w*[’-]?\w*\.?\,?\s?\d*\D?\]\s?//;

             $string =~ s/^\(\w+\,?\s?\d*\D?\)\s?//;

             $string =~ s/\.\s\d+$/\./; #retira o numero no final das ref
             $string = alltrim ($string);

             #preparação das referências para utlizar o brill´s tagger
```

```
                $string =~ s/(\S)([\.|\;|\:|\?|\\|\/|\,|\"|\'|\´|\‘|\!|\@|\#|\%|\¨|\&|\*|\-|\=|\{|\}|\[|\]|\~|\<|\>|
                                \(|\)|\||Æ|\+|\-|\°|\°|\ø|\¸|\®|\¯|\■|\■])/$1 $2/ig;
                $string =~ s/(\S)([\.|\;|\:|\?|\\|\/|\,|\"|\'|\´|\‘|\!|\@|\#|\%|\¨|\&|\*|\-|\=|\{|\}|\[|\]|\~|\<|\>|
                                \(|\)|\||Æ|\+|\-|\°|\°|\ø|\¸|\®|\¯|\■|\■])/$1 $2/ig;

                $string =~ s/([\.|\;|\:|\?|\\|\/|\,|\"|\'|\´|\‘|\!|\@|\#|\%|\¨|\&|\*|\-|\=|\{|\}|\[|\]|\~|\<|\>|
                                \(|\)|\||Æ|\+|\-|\°|\°|\ø|\¸|\®|\¯|\■|\■])(\S)/$1 $2/ig;
                $string =~ s/([\.|\;|\:|\?|\\|\/|\,|\"|\'|\´|\‘|\!|\@|\#|\%|\¨|\&|\*|\-|\=|\{|\}|\[|\]|\~|\<|\>|
                                \(|\)|\||Æ|\+|\-|\°|\°|\ø|\¸|\®|\¯|\■|\■])(\S)/$1 $2/ig;

                $string =~ s/\//\$b/ig;
                $string =~ s/\cM//;

                if(length($string) >= 450)
                {
                   print ERROR $file." ".$string, "\n";
                }

               print ALL $string, "\n";
          }

          #Lê a próxima linha, pulando as linhas em branco
          while (length($InputLine) < 2 && defined($InputLine))
          {
              $InputLine = <INPUTFILE>;
          }

        } # Fim do while
        close(INPUTFILE);

    } #Fim FOREACH

    close(ERROR);
    close(ALL);

  } #Fim ELSE
}

##################################################################################################################

##################################################################################################################
#                         Divisão do Corpus Original em 10 Corpus de Treino e Teste
#                                    Autor: Alberto Cáceres Álvarez
# ---------crossv.prl---------          ICMC - USP - São Carlos          ---------crossv.prl---------
#                                     2o. Semestre de 2005
#
# Este programa realiza a divisão do corpus original considerando que o método de avaliação a ser
# utilizado é 10-fold Cross-Validation.
#
# usage: cat TAGGED-CORPUS-ENTIRE | perl crossv.prl
##################################################################################################################
for($fold = 1;$fold <= 10;$fold++)
{
open("TR"."$fold", ">cp_treino$fold");
open("TS"."$fold", ">cp_teste$fold");
}

srand;
while(<STDIN>) {
    $x = rand;
    if ($x <= 0.1) {
        print TS1 $_;
```

```
    print TR2 $_;
    print TR3 $_;
    print TR4 $_;
    print TR5 $_;
    print TR6 $_;
    print TR7 $_;
    print TR8 $_;
    print TR9 $_;
    print TR10 $_;
}
elsif ($x > 0.1 && $x <= 0.2)
{
    print TS2 $_;

    print TR1 $_;
    print TR3 $_;
    print TR4 $_;
    print TR5 $_;
    print TR6 $_;
    print TR7 $_;
    print TR8 $_;
    print TR9 $_;
    print TR10 $_;
}
elsif ($x > 0.2 && $x <= 0.3)
{
    print TS3 $_;

    print TR1 $_;
    print TR2 $_;
    print TR4 $_;
    print TR5 $_;
    print TR6 $_;
    print TR7 $_;
    print TR8 $_;
    print TR9 $_;
    print TR10 $_;
}
elsif ($x > 0.3 && $x <= 0.4)
{
    print TS4 $_;

    print TR1 $_;
    print TR2 $_;
    print TR3 $_;
    print TR5 $_;
    print TR6 $_;
    print TR7 $_;
    print TR8 $_;
    print TR9 $_;
    print TR10 $_;
}
elsif ($x > 0.4 && $x <= 0.5)
{
    print TS5 $_;

    print TR1 $_;
    print TR2 $_;
    print TR3 $_;
    print TR4 $_;
    print TR6 $_;
```

```
    print TR7 $_;
    print TR8 $_;
    print TR9 $_;
    print TR10 $_;
}
elsif ($x > 0.5 && $x <= 0.6)
{
    print TS6 $_;

    print TR1 $_;
    print TR2 $_;
    print TR3 $_;
    print TR4 $_;
    print TR5 $_;
    print TR7 $_;
    print TR8 $_;
    print TR9 $_;
    print TR10 $_;
}
elsif ($x > 0.6 && $x <= 0.7)
{
    print TS7 $_;

    print TR1 $_;
    print TR2 $_;
    print TR3 $_;
    print TR4 $_;
    print TR5 $_;
    print TR6 $_;
    print TR8 $_;
    print TR9 $_;
    print TR10 $_;
}
elsif ($x > 0.7 && $x <= 0.8)
{
    print TS8 $_;

    print TR1 $_;
    print TR2 $_;
    print TR3 $_;
    print TR4 $_;
    print TR5 $_;
    print TR6 $_;
    print TR7 $_;
    print TR9 $_;
    print TR10 $_;
}
elsif ($x > 0.8 && $x <= 0.9)
{
    print TS9 $_;

    print TR1 $_;
    print TR2 $_;
    print TR3 $_;
    print TR4 $_;
    print TR5 $_;
    print TR6 $_;
    print TR7 $_;
    print TR8 $_;
    print TR10 $_;
}
else
```

```
    {
        print TS10 $_;

        print TR1 $_;
        print TR2 $_;
        print TR3 $_;
        print TR4 $_;
        print TR5 $_;
        print TR6 $_;
        print TR7 $_;
        print TR8 $_;
        print TR9 $_;
    }
}

for($fold = 1;$fold <= 10;$fold++)
{
open("TR"."$fold");
open("TS"."$fold");
}
#############################################################################################################

/*############################################################################################################
#                            Comparação entre TEST-CORPUS e TAGGED-TEST-CORPUS
#                                     Autor: Alberto Cáceres Álvarez
# ---------evaluate.c---------          ICMC - USP - São Carlos          ---------evaluate.c---------
#                                       2o. Semestre de 2005
############################################################################################################*/
#include <stdio.h>
#include <string.h>

int main(int argc, char* argv[])
{
if (argc == 5)
{ FILE *in_res, *in_base, *out_err, *out_freq;
char buffer1[100],//Variaveis para armazenamento temporario durante a leitura
     buffer2[100],
     tag1[30],//Variaveis para armazenamento temporario das etiquetas
     tag2[30],
     *ptr,//Ponteiro utilizado para manipular strings
     nome_tag[100][30]; //Array para armazenar o nome das etiquetas

int  n_words,  //Numero total de palavras
     n_tags_err,//Numero de etiquetas diferentes das do corpus etiquetado manualmente
     cont_tag[100],//Numero de ocorrencias de cada etiqueta
     i;

if (!(in_res=fopen(argv[1],"rt")))
{ fprintf(stderr, "Erro ao abrir corpus gerado pelo etiquetador.\n");
return 1;}
if (!(in_base=fopen(argv[2],"rt")))
{ fprintf(stderr, "Erro ao abrir corpus etiquetado manualmente.\n");
return 1;}
if (!(out_err=fopen(argv[3],"wt")))
{ fprintf(stderr, "Erro ao abrir arquivo para exibir etiquetas erradas.\n");
return 1;}
if (!(out_freq=fopen(argv[4],"wt")))
{ fprintf(stderr, "Erro ao abrir arquivo para exibir frequencia das etiquetas.\n");
return 1;}
//inicializa variaveis de contagem de tags erradas e numero de palavras
n_words = 0;
n_tags_err = 0;
```

```
//inicializa variaveis para contagem do numero de cada tag
nome_tag[0][0] = '\0';
for(i = 0; i < 100; i++)
      cont_tag[i] = 0;

while (!feof(in_res) && !feof(in_base)){
      fscanf(in_res, "%s ", buffer1);
      ptr = strchr(buffer1, '/');
      strcpy (tag1, ptr+1);
      *ptr = '\0';

      fscanf(in_base, "%s ", buffer2);
      ptr = strchr(buffer2, '/');
      strcpy (tag2, ptr+1);
      *ptr = '\0';
      if (strcmp(buffer1, buffer2) == 0){
        n_words++;

        //Verifica a posicao no array da tag.
        for (i = 0;
            (nome_tag[i][0] != '\0')
            && (strcmp(tag2, nome_tag[i]) != 0)
            && (i < 99);
            i++);

        //Caso esta estiqueta seja nova na lista, o nome é gravado e
        //a marca '\0', indicando fim da lista, é gravado na posição seguinte.
        if(nome_tag[i][0] == '\0') {
          strcpy(nome_tag[i], tag2);
          nome_tag[i+1][0] = '\0';
        };
        cont_tag[i]++;

        //Verifica se as etiquetas dos dois arquivos sao coincidentes.
        if (strcmp(tag1, tag2) != 0){
          n_tags_err++;
          fprintf(out_err, "%s_%s\t%s\n", tag2, tag1, buffer1);
        }
      } else {
        fprintf(stderr, "Palavras não coincidentes: %s : %s\n",
                buffer1, buffer2);
      }
    }
    printf("***Numero_de_palavras: %d\n", n_words);
    printf("***Numero_de_etiquetas_erradas: %d\n", n_tags_err);

    for (i = 0; (nome_tag[i][0] != '\0') && (i < 100); i++) {
      fprintf(out_freq, "%s\t%d\n", nome_tag[i], cont_tag[i]);
    };
    fclose(in_res);
    fclose(in_base);
    fclose(out_err);
    fclose(out_freq);

  } else {
    printf("\navalia CORPUS-TESTE CORPUS-MANUAL ETIQUETAS-ERRADAS ETIQUETAS-FREQUENCIA\n\n");
    printf("\n    CORPUS-TESTE           Corpus de teste gerado pelo etiquetador.\n");
    printf("\n    CORPUS-MANUAL          Corpus etiquetado manualmente.\n");
    printf("\n    ETIQUETAS-ERRADAS      Arquivo de saída para exibir a etiqueta correta,\n");
    printf("                           a etiqueta errada, e a palavra etiquetada\n");
    printf("                           erradamente.\n");
```

```
    printf("\n    ETIQUETAS-FREQUENCIA   Arquivo de saída para exibir a frequência de\n");
    printf("                           cada etiqueta.\n");
  }
  return 0;
}
//#############################################################################################################

/*#############################################################################################################
#                        Número de erros entre TEST-CORPUS e TAGGED-TEST-CORPUS
#                                    Autor: Alberto Cáceres Álvarez
# --------count_errors.c--------       ICMC - USP - São Carlos          --------count_errors.c--------
#                                      2o. Semestre de 2005
#############################################################################################################*/
#include <stdio.h>
#include <string.h>

int main()
{
  char tags_1[60], tags_2[60], palavra[100];
  int cont = 1;

  fscanf(stdin, "%s ", tags_1);
  fscanf(stdin, "%s ", palavra);

  while (!feof(stdin)){
    fscanf(stdin, "%s ", tags_2);
    fscanf(stdin, "%s ", palavra);
    if (strcmp(tags_1, tags_2) == 0) {
      cont++;
    } else {
      fprintf(stdout, "%s\t%d\n", tags_1, cont);
      cont = 1;
      strcpy(tags_1, tags_2);
    }
  }
  fprintf(stdout, "%s\t%d\n", tags_1, cont);
  return 0;
}
//#############################################################################################################

/*#############################################################################################################
#                        Número de erros globais entre TEST-CORPUS e TAGGED-TEST-CORPUS
#                                    Autor: Alberto Cáceres Álvarez
# --------global_errors.c--------      ICMC - USP - São Carlos         --------global_errors.c--------
#                                      2o. Semestre de 2005
#############################################################################################################*/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main()
{
  char tags[60], aux[10], tag_atual[60], *ptr;

  int n_err, n_err_tag;
  tag_atual[0] = '\0';
  n_err_tag = 0;

  while (!feof(stdin)){
    fscanf(stdin, "%s ", tags);
    fscanf(stdin, "%s ", aux);

    if (strchr(tags, '*')) {
```

```
      if (tag_atual[0] != '\0') {
         fprintf(stdout, "%s\t%d\n", tag_atual, n_err_tag);
      }
      tag_atual[0] = '\0';
      n_err_tag = 0;

      fprintf(stdout, "%s %s\n", tags, aux);
    } else {
      ptr = strchr(tags, '_');
      *ptr = '\0';
      n_err = atoi(aux);

      if (strcmp(tags, tag_atual) == 0) {
        n_err_tag += n_err;
      } else {
        if (tag_atual[0] != '\0') {
          fprintf(stdout, "%s\t%d\n", tag_atual, n_err_tag);
        }
        strcpy(tag_atual, tags);
        n_err_tag = n_err;
      }
    }
  }
  fprintf(stdout, "%s\t%d\n", tag_atual, n_err_tag);
  return 0;
}
//#############################################################################################################

/*#############################################################################################################
#                        Cálculo do valor relativo que irá preencher alguma tabela
#                                     Autor: Alberto Cáceres Álvarez
# --------calc\_val\_rel.c--------      ICMC - USP - São Carlos        --------calc\_val\_rel.c--------
#                                       2o. Semestre de 2005
#############################################################################################################*/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main()
{
  char tags[60], aux[10];
  int n_err_tot, n_err, cont_treino = 0, i;

  while (!feof(stdin)){
    fscanf(stdin, "%s ", tags);
    fscanf(stdin, "%s ", aux);

    if (strchr(tags, '*')) {
      //Le o numero total para sequencia seguinte.
      fscanf(stdin, "%s ", tags);
      fscanf(stdin, "%s ", aux);
      n_err_tot = atoi(aux);
      cont_treino++;
    } else {
      n_err = atoi(aux);
      fprintf(stdout, "%s\t%d\t%.10f\n", tags, cont_treino, ((double)n_err)/n_err_tot);
    }
  }
  return 0;
}
//#############################################################################################################
```

```
/*#############################################################################################################
#                    Cálculo do valor relativo que irá preencher alguma tabela (modificado)
#                                      Autor: Alberto Cáceres Álvarez
# --------calc\_val\_mod.c--------      ICMC - USP - São Carlos        --------calc\_val\_mod.c--------
#                                        2o. Semestre de 2005
#############################################################################################################*/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main()
{
  char tags[60], aux[10];
  int n_err_tot, n_err, cont_treino = 0, i;

  while (!feof(stdin)){
    fscanf(stdin, "%s ", tags);
    fscanf(stdin, "%s ", aux);

    if (strchr(tags, '*')) {
      //Le o numero total para sequencia seguinte.
      fscanf(stdin, "%s ", tags);
      fscanf(stdin, "%s ", aux);
      n_err_tot = atoi(aux);
      cont_treino++;
    } else {
      n_err = atoi(aux);
      fprintf(stdout, "%s\t%d\t%.10f\n", tags, cont_treino, (double)n_err);
    }
  }
  return 0;
}
//#############################################################################################################

/*#############################################################################################################
#                       Preenche um arquivo com valores relativos em forma de tabela
#                                      Autor: Alberto Cáceres Álvarez
# --------table.c--------               ICMC - USP - São Carlos               --------table.c--------
#                                        2o. Semestre de 2005
#############################################################################################################*/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(int argc, char* argv[])
{
  char tags_1[60], tags_2[30], aux[15], aux2[4];
  int pos_tabela, n_treino, n_tot_treinos = 1;

  //Armazena na variavel o numero de colunas, se especificado.
  if (argc == 2){
    n_tot_treinos = atoi(argv[1]);
  }
  fscanf(stdin, "%s ", tags_1);
  fscanf(stdin, "%s ", aux2);
  fscanf(stdin, "%s ", aux);
  n_treino = atoi(aux2);
  fprintf(stdout, "%s", tags_1);

  pos_tabela = 1;
  for(; pos_tabela < n_treino; pos_tabela++){
     fprintf(stdout, "\t%f", 0);}
```

```
  fprintf(stdout, "\t%s", aux);
  pos_tabela++;

  while (!feof(stdin)){
    fscanf(stdin, "%s ", tags_2);
    fscanf(stdin, "%s ", aux2);
    fscanf(stdin, "%s ", aux);
    n_treino = atoi(aux2);

    if(strcmp(tags_1, tags_2) == 0){

      for(; pos_tabela < n_treino; pos_tabela++){
        fprintf(stdout, "\t%f", 0);}
      fprintf(stdout, "\t%s", aux);
      pos_tabela++;
    } else {
       strcpy(tags_1, tags_2);

      for(; pos_tabela <= n_tot_treinos; pos_tabela++){
        fprintf(stdout, "\t%f", 0);}
      fprintf(stdout, "\n%s", tags_1);
      pos_tabela = 1;
      for(; pos_tabela < n_treino; pos_tabela++){
        fprintf(stdout, "\t%f", 0);
      }
      fprintf(stdout, "\t%s", aux);
      pos_tabela++;
    }
  }
  for(; pos_tabela <= n_tot_treinos; pos_tabela++){
    fprintf(stdout, "\t%f", 0);}
  return 0;
}
//#############################################################################################################
```

```
#############################################################################################################
#               Information Extraction: Pré-processamento das Referências Bibliográficas
#                                     Autor: Alberto Cáceres Álvarez
# ---------bibtex.pl---------          ICMC - USP - São Carlos          ---------bibtex.pl---------
#                                       2o. Semestre de 2005
#
# Este programa realiza a etiquetagem das informações de cada campo para cada entrada de um arquivo .bib.
#############################################################################################################

open(READ, "C:/Mestrado/JabRef_bibtex/00.bib");
open(WRITE, ">C:/Mestrado/Bibliographic-styles/00.bib");

$nextline = <READ>;
while(defined($nextline))
{
$linha = $nextline;
do{
$nextline = <READ>;
}while($nextline eq "\t\n");
#concatenas varias linhas referentes a um mesmo campo
if(not $linha =~ /^[@}\n]/)
```

```
{
if(not $nextline =~ /^[@}\n]/)
{
$nextline =~ s/\s*\=\s*/=/;#ajustes
until($nextline =~ /^.*=/ || $nextline =~ /^}/)
{
$linha =~ s/\n//;
$nextline =~ s/^\s*//;
$linha = $linha." ".$nextline;
do{
$nextline = <READ>;
}while($nextline eq "\t\n");
$nextline =~ s/\s*\=\s*/=/;#ajustes
}
}
#resolve o problema das aspas dupla ‘‘......’’
$linha =~ s/\\\"/\\aspas/ig;
$linha =~ s/‘‘/\"/ig;
$linha =~ s/’’/\"/ig;
for($i=0;$i<3;$i++){
$linha =~ s/\"/‘‘/;
$linha =~ s/\"/’’/;
}
$linha =~ s/\\aspas/\\\"/g;
$linha =~ s/(\\ldots|\\-) //g;
$linha =~ s/\\({|})/$1/ig; #retirando \{footnotesize
$linha =~ s/\s+\n/\n/;
$linha =~ s/ \\ / /ig;
$linha =~ s/\\ / /ig;
$linha =~ s/~\\/ \\/ig;

$linha =~ s/^\s*//;#remove espaços no início
$linha =~ s/\s*\=\s*/=/;#ajustes
$linha =~ s/\//1barra1/ig;#brill’s tagger --->>>> 1barra1 representa a ’/’

#capturando a etiqueta e rotulando
if($linha =~ /^(\w*)=/)#get tag
{
$tag = uc $1;
if($linha =~ /(.+=)({.+})(.+)/)#get tag=|{conteúdo}|resto
{
            $init = $1;
$info = $2;
$final = $3;
$info =~ s/[[:blank:]]+/ /ig;
#tokenizando...
if ($tag eq "AUTHOR" || $tag eq "EDITOR" )
{
$info =~ s/(\.|,)(\S)/$1 $2/ig;
$info =~ s/ and and / and /g;
}
else
{
$info =~ s/([!-[]|[]-ÿ])([!-\/]|[:-@]|[\[\]^_‘\|~]|1barra1)/$1 $2/ig;
$info =~ s/([!-[]|[]-ÿ])([!-\/]|[:-@]|[\[\]^_‘\|~]|1barra1)/$1 $2/ig;
$info =~ s/([!-\/]|[:-@]|[\[\]^_‘\|~]|1barra1)(\S)/$1 $2/ig;
$info =~ s/([!-\/]|[:-@]|[\[\]^_‘\|~]|1barra1)(\S)/$1 $2/ig;
$info =~ s/ ,$/,/ig;
}
#etiquetando ...
if($tag eq "MONTH" && $linha =~ /#/)#arrumando o campo MONTH
{
```

```
$info =~ s/(\d\d?)(\D)/$1\/{DAYS}$2/ig;
$info =~ s/(\d\d\d\d)\/{\w+}/$1\/{YEAR}/ig;
$info =~ s/(\-|1barra1)/$1\/$1/ig;
}
elsif($tag eq "ADDRESS" && $linha =~ /{.*#.*{/)
{
        $linha =~ s/}/\/{$tag}}/ig;
        $linha =~ s/\s$//ig;
        $info = $linha;
        $init = $final = "";
}
else#se nao for campo MONTH com o caractere #
{
#ajustes no inicio da string
$info =~ s/{ /{/g;
#ajustes na parte final da string
$info =~ s/ }/}/g;
#rotula as palavras
$info =~ s/ /\/{$tag} /ig;
#rotula a ultima palavra da linha
$info =~ s/}$/\/{$tag}}/;
#$info =~ s/}\n/\/{$tag}}\n/;
}
#retirando e consertando etiquetas ...
$info =~ s/ and\/{AUTHOR}/ and/ig;#desmarca 'and'
$info =~ s/ and\/{EDITOR}/ and/ig;#desmarca 'and'
$info =~ s/(\S)(\.|,)\/{(AUTHOR|EDITOR)}/$1\/{$3}$2/ig;
$info =~ s/ (\.|,)\/{(AUTHOR|EDITOR)}/$1/ig;
#retirando o ponto do 'and'
$info =~ s/ and\. / and /g;

$info =~ s/(\\em|\\bf|\\it|\\rm|\\sl|\\sf|\\sc|\\tt|\\footnotesize|\\v)\/{\w+} /$1 /ig;
#$info =~ s/({|})\/{\w+} /$1/ig;
$info =~ s/\/{CROSSREF}//;
$info =~ s/\/{URL}/\/URL/g;
$info =~ s/-\/{PAGES} -\/{PAGES}/--\/-/g;
#ajustando ... \"/{TITLE}
$info =~ s/\\(.)\/{\w+} /\\$1/ig;
for($i=0;$i<7;$i++){
$info =~ s/(\\crossref{[!-||~-ÿ]+)\/{\w+} /$1/g;
}
$linha = $init.$info.$final."\n";
}
 }
}
print WRITE $linha;
}
close(READ);
close(WRITE);
#############################################################################################################

#############################################################################################################
#                          Extração de elementos de referências bibliográficas
#                                      Autor: Alberto Cáceres Álvarez
# ---------extraction.pl---------       ICMC - USP - São Carlos       ---------extraction.pl---------
#                                        1o. Semestre de 2006
#
# Este programa recebe como entrada um conjunto de referências etiquetadas (TAGGED-CORPUS) e produz como
# saída um arquivo XML que contém as informações extraídas para cada referência
#############################################################################################################
#unidades de informações (tags) a serem extraídas das referências
@tags = ("ADDRESS","AUTHOR","BOOKTITLE","CHAPTER","EDITION","EDITOR","HOWPUBLISHED",
```

```
        "INSTITUTION","ISBN","ISSN","JOURNAL","MONTH","NOTE","NUMBER","ORGANIZATION",
        "INITPAGE","FINALPAGE","PUBLISHER","SCHOOL", "SERIES","TYPE","TITLE","URL",
        "URLACCESSDATE","VOLUME","YEAR","PAGES","DAYS","CROSSREF","KEY");

open(READ, "TAGGED-CORPUS");
open(WRITE, ">extr.xml");

print WRITE "<?xml version=\"1.0\" encoding=\"ISO-8859-1\"\?>\n<references>\n";

$ref = <READ>;
while(defined($ref))
{
   #as informações da etiqueta INDICATOR não devem ser extraídas, simplesmente, por apenas serem indicadores
   $ref =~ s/([^\s]+)\/INDICATOR[[:space:]]+//g;
   $ref =~ s/[{}\[\]<>\(\)]\/[^\s]+[[:space:]]+//ig;
   #quando houver varios 'mês e dia' consecutivos, altera a etiqueta dos 'dias' para MONTH
   $ref =~ s/([[:alpha:]]+\/MONTH \d\d?)\/DAYS ((-\/- ){1,2}[[:alpha:]]+\/MONTH \d\d?)\/DAYS /
             $1\/MONTH $2\/MONTH /g;
   $ref =~ s/(\d\d?)\/DAYS ([[:alpha:]]+\/MONTH (-\/- ){1,2}\d\d?)\/DAYS ([[:alpha:]]+\/MONTH )/
             $1\/MONTH $2\/MONTH $4/g;
   #caso uma referência possuir dois anos, remove-se o ano mais próximo do final da referência
   if($ref =~ /\/YEAR.+\/YEAR/)
   {
     $ref =~ s/(.+)\d{4}[a-z]?\/YEAR[[:space:]]+/$1/;
   }
   $ref =~ s/(\d{4})[a-z]\/YEAR/$1\/YEAR/g;

   #marca com uma tag XML o início e fim de cada informação a ser extraída, ou seja, delimita dentro da
   #referência onde começa e termina uma informação
   for($i = 0;$i < @tags;$i++)
   {
     $ltag = lc $tags[$i];
     $ref =~ s/([^\s]+\/$tags[$i]) /<$ltag>$1 /;
     $ref =~ s/(.*)\/$tags[$i] /$1<1$ltag> /;
   }
   $ref =~ s/([^\s]+)\/[^\s]+[[:space:]]+/$1 /ig;#removendo etiquetas
   #remove espaços e sinais de pontuação que estiverem entre o fim (</info1>)
   #de uma informação e início (<info2>) de outra
   $ref =~ s/(<1[a-z]+>)([[:punct:][:space:]¿¡]+)(<[a-z]+>)/$1\n$3/g;
   $ref =~ s/<1([a-z]+)>/<\/$1>/g;
   #$b significa '/' para não confundir com a / que separa a palavra de sua etiqueta
   $ref =~ s/\$b/\//ig;
   $ref =~ s/ ([-'\/])/$1/ig;
   $ref =~ s/([-'\/]) /$1/ig;
   $ref =~ s/ ([.,:])/$1/ig;
   #remove os sinais de pontuação localizados após a última tag XML (final da ref)
   $ref =~ s/(.+<\/[a-z]+>).+/$1/;

   #captura a string que contém os autores da referência e
   #extrai separadamente cada autor (1º, 2º, 3º, ...) da referência
   if($ref =~ /((.|\n)*)(<author>.+<\/author>)((.|\n)+)/)
   {
     $init = $1;
     $autores = $3;
     $end = $4;
     $autores =~ s/;/,/ig;
     $autores =~ s/,? and/, and/;
     $autores =~ s/([[:alpha:]-.]+), ((([[:alpha:]-.]+\.\s?){1,2}),/$2 $1,/ig;
     $autores =~ s/(and |<author>)([[:alpha:]-.]+), ((([[:alpha:]-.]+\.?\s?){1,2})<\/author>/$1$3. $2<\/author>/;
     $autores =~ s/, and /, /;
     $autores =~ s/ ([[:alpha:]])<\/author>/ $1.<\/author>/;
     $autores =~ s/, /<\/author>\n<author>/g;
```

```
    $ref = $init.$autores.$end;
  }

  #captura a string que contém os editores da referência e extrai separadamente cada editor
  if($ref =~ /((.|\n)*)(<editor>.+<\/editor>)((.|\n)+)/)
  {
    $init = $1;
    $editores = $3;
    $end = $4;
    $editores =~ s/,? and/, and/;
    $editores =~ s/([[:alpha:]-.]+), (([[:alpha:]-.]+\.\s?){1,2}),/$2 $1,/ig;
    $editores =~ s/(and |<editor>)([[:alpha:]-.]+), (([[:alpha:]-.]+\.?\s?){1,2})<\/editor>/$1$3. $2<\/editor>/;
    $editores =~ s/, and /, /;
    $editores =~ s/ ([[:alpha:]])<\/editor>/ $1.<\/editor>/;
    $editores =~ s/, /<\/editor>\n<editor>/g;
    $ref = $init.$editores.$end;
  }

  #remove espaços na url da referência
  if($ref =~ /((.|\n)*)(<url>.+<\/url>)((.|\n)*)/)
  {
    $init = $1;
    $url = $3;
    $end = $4;
    $url =~ s/[[:blank:]]+//ig;
    $ref = $init.$url.$end;
  }

  $ref =~ s/[[:blank:]]+/ /ig;
  print WRITE "<ref>\n".$ref."\n</ref>\n\n"; #delimitadores de referências
  $ref = <READ>;
}

print WRITE "</references>";
close(READ);
close(WRITE);
#########################################################################################################
```

C Manual do Etiquetador TBL

Neste apêndice são apresentadas as ferramentas utilizadas e os arquivos utilizados/gerados no treinamento e etiquetagem do etiquetador TBL.

O treinamento com o TBL consiste de duas etapas: aprender regras para etiquetagem de palavras desconhecidas e aprender regras contextuais. No treinamento para aprendizado de regras para palavras desconhecidas utiliza-se o comando:

```
unknown-lexical-learn.prl BIGWORDLIST SMALLWORDTAGLIST BIGBIGRAMLIST N
LEXRULEOUTFILE
```

Em que:
`BIGWORDLIST` é um arquivo com todas as palavras/símbolos que estão presentes no corpus em ordem decrescente de freqüência.

and
:
pages
IEEE
;
Signal
for

`SMALLWORDTAGLIST` é um arquivo no formato - palavra etiqueta freqüência - que lista o número de vezes que uma palavra aparece com uma dada etiqueta no corpus.

. . 89530
, , 52419
and AUTHOR 7689
In INDICATOR 5385
Conference BOOKTITLE 1420

`BIGBIGRAMLIST` é um arquivo com os bigramas que aparecem no corpus de treinamento.

: 33
pages .
image processing
Efficient computation
machine learning

`N` é um número que indica que deverão ser utilizados apenas os bigramas onde pelo menos uma das palavras é uma das n mais freqüentes no corpus.

`LEXRULEOUTFILE` é o arquivo onde serão armazenadas as regras aprendidas.

No treinamento para aprendizado de regras contextuais utiliza-se o comando:

```
contextual-rule-learn TAGGED-CORPUS DUMMY-TAGGED-CORPUS CONTEXT-RULEFILE
TRAINING.LEXICON
```

Em que:
`TAGGED-CORPUS` é o arquivo que contém o texto etiquetado manualmente, tokenizado e no formato de uma sentença por linha.

`DUMMY-TAGGED-CORPUS` é o arquivo formado pelo mesmo texto do arquivo TAGGED-CORPUS só que etiquetado pelo etiquetador inicial.

`CONTEXT-RULEFILE` é o arquivo onde serão armazenadas as regras contextuais.

`TRAINING.LEXICON` é o arquivo do léxico, que é formado pelas palavras e suas possíveis etiquetas que tenham aparecido no corpus de treinamento.

collection TITLE PUBLISHER NOTE
Tasaki AUTHOR
Based BOOKTITLE TITLE JOURNAL
Electronics JOURNAL BOOKTITLE INSTITUTION ORGANIZATION PUBLISHER
2000a YEAR

O etiquetador TBL tem ferramentas auxiliares para gerar este arquivos que são utilizados no treinamento e estão descritas no arquivo README que acompanha o etiquetador.

Na etiquetagem é utilizado o comando:

```
tagger LEXICON CORPUS BIGBIGRAMLISTS LEXRULEOUTFILE CONTEXTUALRULEFILE
[opções]
```

Em que:
`CORPUS` é o nome do arquivo que será etiquetado.

Depois do nome de todos os arquivos, podem ser colocadas opções - Tabela 10.

Opções	**Descrição**
-h	Help
-w wordlist	Provê um conjunto extra de palavras além das que estão no léxico
-i filename	Grava o resultado intermediário do etiquetador inicial em um arquivo
-s number	Processa o corpus para ser etiquetado "number" linhas a cada iteração
-S	Utiliza apenas o etiquetador inicial
-F	Utiliza apenas o etiquetador final, ou seja o corpus deve estar etiquetado

Tabela 10: Opções do comando `tagger`

Existe ainda a possibilidade de aumentar a lista de bigramas e o léxico. Há também a possibilidade de se alterar manualmente as regras.

Referências

American Psychological Association (1994). *Publication Manual of the American Psychological Association* (4th ed.). Washington, DC: American Psychological Association.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (2002). *NBR 6023: Informação e Documentação - Referências - Elaboração*. Rio de Janeiro: ABNT.

Borko, H. (1967). *Automated Language Processing*. New York: John Wiley and Sons.

Brasil, C. & A. A. Lopes (2004, November). Mineração de artigos científicos usando aprendizado de máquina. In *Jornadas Chilenas de la Computación - V Workshop de Inteligência Artificial*, Arica-Chile, pp. 1–7.

Brill, E. (1994). Some advances in transformation-based part of speech tagging. In *AAAI '94: Proceedings of the twelfth national conference on Artificial intelligence (vol. 1)*, Seattle, Washington, United States, pp. 722–727. American Association for Artificial Intelligence.

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

Brill, E. (1997). Unsupervised learning of disambiguation rules for part of speech tagging. In *Natural Language Processing Using Very Large Corpora*. Kluwer Academic Press.

Chanod, J. P. & P. Tapanainen (1995). Creating a tagset, lexicon and guesser for a french tagger.

Chicago Editorial Staff (1993). *Chicago Manual of Style* (14th ed.). Chicago: University of Chicago Press.

Connan, J. & C. W. Omlin (2000, March 24). Bibliography extraction with hidden markov models. Technical report.

Cowie, J. & W. Lehnert (1996). Information extraction. *Communications of the ACM 39*(1), 80–91.

Daelemans, W., J. Zavrel, & S. Berck (1996). MBT: A memory based part of speech tagger-generator. In E. Ejerhed & I. Dagan (Eds.), *Proceedings of the Fourth Workshop on Very Large Corpora*, pp. 14–27.

Day, M.-Y., T.-H. Tsai, C.-L. Sung, C.-W. Lee, S.-H. Wu, C.-S. Ong, & W.-L. Hsu (2005). A knowledge-based approach to citation extraction. In D. Zhang, T. M. Khoshgoftaar, & M.-L. Shyu (Eds.), *Proceedings of the 2005 IEEE International Conference on Information Reuse and Integration, IRI - 2005, August 15-17, 2005, Las Vegas Hilton, Las Vegas, NV, USA*, pp. 50–55. IEEE Systems, Man, and Cybernetics Society.

Ding, Y., G. Chowdhury, & S. Foo (1999). Template mining for the extraction of citation from digital documents. *Proceedings of the Second Asian Digital Library Conference, Taiwan*, 47–62.

EAGLES - Expert Advisory Group on Language Engineering Standards (1996). Recommendations for the morphosyntactic annotation of corpora.

Gaizauskas, R. & Y. Wilks (1998). Information extraction: Beyond document retrieval. *Journal of Documentation 54*(1), 70–105.

Geng, J. & J. Yang (2004). Autobib: Automatic extraction of bibliographic information on the web. In *8th International Database Engineering and Applications Symposium (IDEAS 2004), 7-9 July 2004, Coimbra, Portugal*, pp. 193–204. IEEE Computer Society.

Giuffrida, G., E. C. Shek, & J. Yang (2000). Knowledge-based metadata extraction from postscript files. In *DL '00: Proceedings of the fifth ACM conference on Digital libraries*, New York, NY, USA, pp. 77–84. ACM Press.

Grishman, R. (1997). Information extraction: Techniques and challenges. In *SCIE '97: International Summer School on Information Extraction*, pp. 10–27. Springer-Verlag.

Han, H., C. L. Giles, E. Manavoglu, H. Zha, Z. Zhang, & E. A. Fox (2003). Automatic document metadata extraction using support vector machines. In *JCDL '03: Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries*, Washington, DC, USA, pp. 37–48. IEEE Computer Society.

Hobbs, J., D. Appelt, J. Bear, D. Israel, M. Kameyama, M. Stickel, & M. F. Tyson (1997). Fastus: A cascaded finite-state transducer for extracting information from natural-language text. In E. Roche & Y. Schabes (Eds.), *Finite-State Devices for Natural Language Processing*, pp. 383–406. Cambridge, MA: MIT Press.

Lawrence, S., C. L. Giles, & K. Bollacker (1999). Digital libraries and Autonomous Citation Indexing. *IEEE Computer 32*(6), 67–71.

Leech, G., R. Garside, & M. Bryant (2004). Claws4: The tagging of the british national corpus. In *Proceedings of the 15th International Conference on Computational Linguistics*, pp. 622–628.

Lin, C.-Y. & E. Hovy (2003). The potential and limitations of automatic sentence extraction for summarization. In *Proceedings of the HLT-NAACL 03 on Text summarization workshop*, Morristown, NJ, USA, pp. 73–80. Association for Computational Linguistics.

Lopes, A. A. & A. Jorge (2000). Combining rule-based and case-based learning for iterative part-of-speech tagging. In *EWCBR '00: Proceedings of the 5th European Workshop on Advances in Case-Based Reasoning*, London, UK, pp. 26–36. Springer-Verlag.

Ma, Q., K. Uchimoto, M. Murata, & H. Isahara (1999, July). Elastic neural networks for part of speech tagging. In *International Joint Conference on Neural Networks(IJCNN'99)*, Washington, DC.

Mao, S., J. W. Kim, & G. R. Thoma (2004). A dynamic feature generation system for automated metadata extraction in preservation of digital materials. In *1st International Workshop on Document Image Analysis for Libraries (DIAL 2004), 23-24 January 2004, Palo Alto, CA, USA*, pp. 225–232. IEEE Computer Society.

Marcus, M. P., B. Santorini, & M. A. Marcinkiewicz (1994). Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics 19*(2), 313–330.

McCallum, A., K. Nigam, J. Rennie, & K. Seymore (2000). Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval 3*(2), 127–163.

Melo, V. & A. A. Lopes (2004a). Identificação eficiente de referências bibliográficas duplicadas em um corpora de artigos científicos. In *II Workshop de Teses e Dissertações em Inteligência Artificial*, São Luis - Maranhão, pp. 71–80.

Melo, V. & A. A. Lopes (2004b). Usando as referências bibliográficas no clustering de artigos científicos. In *Jornadas Chilenas de la Computación - V Workshop de Inteligência Artificial*, Arica-Chile, pp. 1–7.

Melo, V. & A. A. Lopes (2005). Efficient identification of duplicate bibliographical references. In *Proceedings of the 5th Congress of Logic Applied to Technology - Laptec 2005*, Himeji - Japan, pp. 1–8.

Melo, V., M. Secato, & A. A. Lopes (2003). Extração e identificação automáticas de informações bibliográficas de artigos científicos. In *Jornadas Chilenas de la Computación - IV Workshop de Inteligência Artificial*, Chillán, pp. 1–7.

Muslea, I. (1999, July). Extraction patterns for information extraction tasks: A survey. In *Proceedings of AAAI Workshop on Machine Learning for Information Extraction*, Orlando, Florida.

Peng, F. & A. McCallum (2004). Accurate information extraction from research papers using conditional random fields. In *Proceedings of Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pp. 329–336.

Petinot, Y., P. B. Teregowda, H. Han, C. L. Giles, S. Lawrence, A. Rangaswamy, & N. Pal (2003). eBizSearch: an OAI-compliant digital library for eBusiness. In *JCDL '03: Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries*, Washington, DC, USA, pp. 199–209. IEEE Computer Society.

Rajman, M. & R. Besançon (1997). *Text Mining: Natural Language techniques and Text Mining applications*. Chapman and Hall.

Ratnaparkhi, A. A. (1996). A maximum entropy part-of-speech tagger. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*, Philadelphia, Pa.

Schmid, H. (1995). Probabilistic part–of–speech tagging using decision trees. In *Proceedings of the Conference on New Methods in Language Processing*, Manchester, UK.

Seymore, K., A. McCallum, & R. Rosenfeld (1999, July). Learning hidden markov model structure for information extraction. In *AAAI'99 Workshop on Machine Learning for Information Extraction*, Orlando, Florida, USA.

Soderland, S. (1999). *Learning Information Extraction Rules for Semi-structured and Free Text*. Kluwer Academic Publishers.

Takasu, A. (2003). Bibliographic attribute extraction from erroneous references based on a statistical model. In *JCDL '03: Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries*, Houston, Texas, pp. 49–60. IEEE Computer Society.

Voutilainen, A. (1995). A syntax-based part-of-speech analyser. In *Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics*, San Francisco, CA, USA, pp. 157–164. Morgan Kaufmann Publishers Inc.

Wiebe, J., G. Hirst, & D. Horton (1996). Language use in context. *Communications of the ACM 39*(1), 102–111.

Wilkens, M. & J. Kupiec (1996). Training hidden markov models for part of speech tagging. Revision 4.

Yin, P., M. Zhang, Z.-H. Deng, & D. Yang (2004). Metadata extraction from bibliographies using bigram HMM. In Z. Chen, H. Chen, Q. Miao, Y. Fu, E. A. Fox, & E.-P. Lim (Eds.), *Digital Libraries: International Collaboration and Cross-Fertilization, 7th International Conference on Asian Digital Libraries, ICADL 2004, Shanghai, China, December 13-17, 2004, Proceedings*, Volume 3334 of *Lecture Notes in Computer Science*, pp. 310–319. Springer.