Segmentação Textual Automática em Sentenças de Documentos em Português

Carlos N. Silla Jr.*, Celso A. A. Kaestner

Pontifícia Universidade Católica do Paraná
Rua Imaculada Conceição 1155 – 80215-901 Curitiba - PR

{silla,kaestner}@ppgia.pucpr.br

Resumo. *Este artigo apresenta um estudo comparando as dificuldades encontradas para adaptar os diferentes sistemas encontrados na literatura para realizar a tarefa de segmentação textual automática em sentenças de documentos em português, e os resultados obtidos após as adaptações dos mesmos. Foram analisados dois sistemas que utilizam aprendizagem de máquina: MxTerminator e Satz, além de um sistema baseado em regras fixas expressas na forma de Expressões Regulares.*

1. Introdução

A segmentação textual automática é o processo de segmentar automaticamente um texto em unidades menores, que podem ser cláusulas, orações, sentenças, parágrafos e até mesmo tópicos [10]. Neste trabalho é analisada a performance de diferentes métodos encontrados na literatura para realizar a tarefa de segmentação textual automática em sentenças para documentos em português. São analisadas também as facilidades e dificuldades encontradas para adaptar cada um desses sistemas à língua portuguesa.

A tarefa de segmentação textual automática em sentenças, é importante, visto que ela é a base de qualquer aplicação prática que lide com documentos pois na prática os documentos dificilmente são encontrados rotulados. Essa aplicação pode ser um tradutor, um sumarizador, dentre outros.

Dos sistemas encontrados na literatura, por exemplo os que trabalham com sumarização, normalmente utilizam textos com as sentenças já identificadas [4] ou então utilizam algum critério simples como um ponto seguindo de espaço em branco ou caractere de um nova linha [5]. Existem ainda trabalhos onde a tarefa de identificação de sentenças é feita sem nenhum indicativo de como foi realizada [3]. Porém, em um trabalho recente [14], foi verificado que essa etapa merece uma maior atenção, visto os problemas ocasionados com métodos extrativos. Porém esse problema afeta a performance não somente dos métodos extrativos mas também de rotuladores de função sintática, tradutores, dentre outros. Se as sentenças do documento sendo trabalhado não forem identificadas corretamente a performance de todas essas aplicações será aquém do desejado.

*Bolsista PIBIC - CNPq

Este artigo está organizado da seguinte forma: a seção 2 apresenta algumas indicações de como se efetua o tratamento do problema na literatura, e apresenta com maiores detalhes os métodos analisados neste trabalho; os resultados computacionais obtidos com a aplicação dos sistemas propostos são descritos na seção 3; e na seção 4 as conclusões e perspectivas deste trabalho são apresentadas.

2. A Tarefa de Segmentação Textual Automática em Sentenças

Inicialmente é possível pensar que para identificar as sentenças que compõem um documento basta se localizar o caractere correspondente ao "ponto": o que vier antes dele desde o último ponto detectado constitui uma sentença. Porém numa análise um pouco mais detalhada existem vários casos onde essa regra é violada, como endereços de páginas internet (www.sbc.org.br), abreviações (Dr.), números (R$ 1.000,00), dentre outros; por essa razão métodos mais elaborados são necessários.

Os trabalhos existentes na área podem ser classificados em duas categorias: os que utilizam regras fixas e os que utilizam algoritmos de aprendizagem de máquina, para resolver o problema. Dos trabalhos que utilizam regras fixas, como por exemplo, um trabalho recente realizado por Walker et. al [16], onde as regras são descritas na forma de expressões regulares, porém neste trabalho não se encontrou nenhum indicativo de como se tratou o problema ocasionado pelas abreviações. Dos trabalhos que utilizam algoritmos de aprendizagem de máquina existem aqueles que utilizam Classificadores de Árvore de Decisão[13], Modelo de Entropia Máxima, com o sistema MxTerminator [12] e o uso de Redes Neurais, com o sistema Satz [9]. As características utilizadas por esses trabalhos geralmente são os sufixos, prefixos, a existência de maiúsculas, classe gramatical das palavras, dentre outros. Essas características são utilizadas para treinar os algoritmos para poder predizer se um possível final de sentença é uma sentença ou não. O único inconveniente desses métodos é a necessidade de um conjunto de treinamento.

Nas sub-seções seguintes são descritos três sistemas presentes na literatura que realizam a tarefa de segmentação textual automática em sentenças.

2.1. O Sistema baseado em Expressões Regulares

Como representante dos sistemas baseados em regras fixas, foi utilizado um sistema que utiliza regras fixas na forma de expressões regulares e considera o contexto aonde a possível sentença ocorre [15]. Este sistema foi escolhido por ter obtido resultados próximos ao sistema MxTerminator em uma coleção de documentos da base TIPSTER [6] em experimentos realizados recentemente.

O Sistema utiliza uma base de expressões regulares que denotam cadeias que contém o ponto mas não indicam final de sentença, como abreviações e outras seqüências. Por exemplo o endereço eletrônico http://www.sbc.org.br pode ser considerado como um elemento da linguagem denotada pela expressão regular: http://www.$([\text{A-Za-z}]^{+}[.])^{+}$. A base de expressões é indicada por meio de um arquivo texto, permitindo assim que sejam feitas inclusões e/ou alterações nas expressões regulares consideradas de uma forma extremamente simples. Parte do conjunto de expressões regulares utilizadas é indicado na Tabela 1.

Tabela 1: Exemplos de expressões regulares utilizadas pelo sistema

Expressão Regular	Objetivo
[a-zA-Z]$^+$.[@].([a-zA-Z]$^+$.[.])$^+$	Reconhecer E-mails
[http://].[www].[.].([a-zA-Z]$^+$.[.])$^+$	Reconhecer Páginas Internet
[Mr.]	Reconhecer a abreviação Mr.

Para realizar a tarefa de identificação de sentenças o sistema realiza uma varredura do texto desde o seu início até localizar o primeiro ponto (.); em seguida são obtidas cadeias de caracteres antecedentes ao ponto que podem ser denotadas pelas expressões da base: se a palavra que antecede o ponto for denotada por uma expressão da base, então o sistema sabe que aquele ponto não indica "final de uma sentença" e avança até o próximo ponto. Se a palavra que anteceder o ponto não estiver na base de expressões regulares, o sistema verifica a ocorrência de expressões que ocorrem após o ponto, que são casos especiais e precisam de um tratamento diferenciado. Caso o sistema também não encontre nenhuma denotação relativa a estas expressões, o final de uma sentença foi encontrado e deve ser , marcado adequadamente pelos rótulos padrões <S> e </S>. O procedimento é repetido a partir deste ponto até que todo o documento tenha sido analisado. Destacam-se a seguir alguns casos especiais tratados pelo sistema:

- Números decimais: o sistema verifica se o que vem antes do ponto é um número, e se o que vem após o ponto também é um número. Dessa forma ele consegue distinguir entre: ".... 1990." e ".... R$ 23.50", casos que correspondem a uma data antes do final de uma sentença e a um número decimal, respectivamente.
- Parênteses no final de sentenças: como uma característica do idioma inglês, são corretas sentenças como: "(... that night.)" ao invés de: "(... that night).", onde o ponto final antecede o fechamento de parênteses.
- Reticências: O último caso especial tratado pelo sistema está relacionado à ocorrência de reticências; neste caso se verificam as ocorrências sucessivas de pontos, até a localização do último deles, indicativo do final de sentença.

Para adaptar este sistema para o português foi necessário adicionar 240 novas abreviações do idioma na forma de expressões regulares. O arquivo texto que contém as expressões regulares permite a fácil manipulação do arquivo para adicionar ou alterar as expressões regulares.

2.2. O Sistema MxTerminator

O sistema MxTerminator[1] foi desenvolvido por Jeffrey Reynar e Adwait Ratnaparkhi [12] da University of Pennsylvania, e utiliza uma abordagem independente de idioma ou gênero de texto, utilizando um algoritmo de aprendizado de máquina denominado Modelo de Entropia Máxima para identificar as sentenças de um documento.

A partir de um Corpus com as sentenças rotuladas, o modelo aprende a classificar cada ocorrência de .,?,! como sendo um elemento que indica o final ou não de uma sentença. A etapa de treinamento é robusta e não precisa de nenhum tipo de regra fixa ou informações de freqüência de *part-of-speech*, e nem mesmo informações específicas sobre o domínio ou gênero dos textos pois, durante a etapa de treinamento, o sistema cria

[1]Disponível em: ftp://ftp.cis.upenn.edu/pub/adwait/jmx/jmx.tar.gz

uma lista de abreviações induzidas. Essa lista é obtida da seguinte forma: é considerado como sendo uma abreviação qualquer token do conjunto de treinamento que possui um espaço em branco antes e depois da sua ocorrência, e contém um símbolo de possível final de sentença (.,?,!), mas não indica o final de uma sentença.

As possíveis sentenças do documento são identificadas percorrendo o texto em busca de seqüências de caracteres separados por um espaço em branco (tokens) contendo um dos símbolos que indicam o final de uma possível sentença (.,?,!). O token que contém o símbolo que denota uma possível sentença é chamado de Candidato. O sistema então utiliza as seguintes informações, que representam o contexto onde o Candidato ocorre: o prefixo, o sufixo, se o prefixo ou o sufixo estão na lista de abreviações criada, a palavra a esquerda do Candidato, a palavra a direita do candidato, e se à palavra à esquerda ou à direita do candidato estão na lista de abreviações criada.

A idéia principal do Modelo de Entropia Máxima é que a probabilidade de uma certa classe, no caso, os limites da sentença em um certo contexto, podem ser estimadas pela distribuição de probabilidade com entropia máxima sujeita a certas restrições. São consideradas apenas evidências específicas dos limites das sentenças, na forma de conhecimento lingüístico a priori sobre as características contextuais que indicam estes limites, que são determinadas experimentalmente.

O modelo de entropia máxima utilizado no MxTerminator é baseado no modelo utilizado para *part-of-speech tagging* (processo de rotular cada palavra com sua respectiva função sintática) em [11]. Para cada símbolo que indica um possível final de sentença (.,?,!), é estimada a distribuição de probabilidade conjunta p do token e o contexto onde ele ocorre, denotados por c, ocorrendo como um final de sentença. A distribuição de probabilidade é dada por:

$$p(b, c) = \pi \prod_{j=1}^{k} a_j^{f_i(b,c)}$$

onde: $b \in \{\text{sim, não}\}$; π é uma constante de normalização; a_j's são os parâmetros desconhecidos do modelo e cada a_j corresponde a um f_i, que representa uma característica. Logo a probabilidade de um determinado contexto ser uma sentença é dado por p(sim,c). As informações referentes ao contexto utilizadas pelo sistema devem ser codificadas usando características. Por exemplo, uma característica interessante seria:

$$f_i(b, c) = \begin{cases} 1 & \text{Se prefixo}(c) = \text{Mr \& } b = \text{não} \\ 0 & \text{Caso contrário.} \end{cases}$$

Essa característica vai permitir o modelo a descobrir que um ponto no final da palavra Mr. raramente ocorre como uma sentença. Portanto o parâmetro correspondente a essa característica vai com sorte aumentar a probabilidade $p(\textit{não}, c)$ se o prefixo for Mr.

O sistema usa uma regra simples para classificar cada possível sentença: uma possível sentença é uma sentença se e somente se $p(\textit{sim}|c) > 0.5$ onde:

$$p(\textit{sim}|c) = \frac{p(\textit{sim}, c)}{p(\textit{sim}, c) + p(\textit{não}, c)}$$

e c é o contexto onde ocorre a possível sentença.

Para adaptar o MxTerminator para o português o procedimento foi muito simples, visto que o sistema utiliza para o conjunto de treinamento um texto de qualquer tamanho que deve conter uma sentença por linha. Outro fator interessante é que além do texto de treinamento, realmente não foi necessário compilar qualquer outro tipo de informação.

2.3. O Sistema Satz

O sistema Satz[2] foi desenvolvido por David D. Palmer e Marti A. Heart e utiliza redes neurais[8] ou algoritmos de árvore de indução [9]. A arquitetura geral do sistema pode ser vista na figura 1.

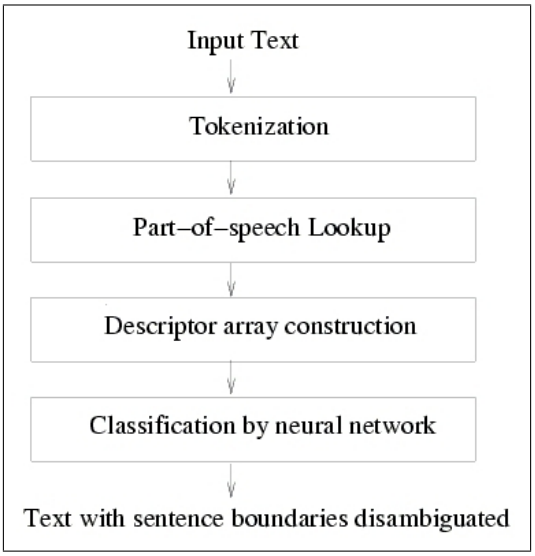

Figura 1: Arquitetura do Sistema Satz [8]

O Satz representa o contexto em volta de um símbolo que pode representar uma sentença, onde o vetor construído para cada palavra do contexto representa uma estimativa da distribuição de probabilidade de *part-of-speech* da palavra. O uso de estimativas de *part-of-speech* consideram o contexto onde a palavra ocorre mais que a palavra em si; esse é um aspecto único do sistema Satz e segundo seus autores é o fator responsável em grande parte pela eficiência e bons resultados do sistema.

Os vetores de contexto são a entrada para um algoritmo de aprendizado de máquina que é treinado para identificar as sentenças. A saída do algoritmo é então usada para determinar se um símbolo que indica um possível final de sentença como o .(ponto) indica ou não uma sentença.

Na etapa de tokenização são extraídos os tokens do texto, os tokens retornados são uma seqüência de caracteres alfabéticos, seqüência de números (este caso já trata o possível problema encontrado quando são identificados números na forma: 23.000,00), e seqüências de um ou mais caracteres não alfabéticos, como pontos e outros caracteres.

O contexto em volta de um símbolo que indica uma possível sentença pode ser representado utilizando as informações de *part-of-speech* de cada palavra. Por exemplo a sentença:

[2]Disponível em: http://elib.cs.berkeley.edu/src/satz/

```
"at the plant.  He had thought"
```

utilizando um *part-of-speech tagger* seria representada por:

```
preposition article noun.  pronoun verb verb
```

Porém, utilizar o *part-of-speech* de cada palavra com o uso de um *tagger* causa um problema de processamento circular pois, como a maioria dos *part-of-speech taggers* necessitam das sentenças já separadas [2]; dessa forma a detecção das sentenças precisa ser realizada antes de utilizar o *tagger*. Por isso durante a análise de *part-of-speech* aos tokens individuais são atribuídas uma série de possíveis *part-of-speech*, baseados em um léxico contendo freqüências de *part-of-speech* e heurísticas simples.

Para resolver este problema, o sistema utiliza uma aproximação do *part-of-speech* de cada palavra em uma das duas formas: 1. Pela probabilidade a priori de todos as freqüências de *part-of-speech* da palavra; 2. Um valor binário para cada valor possível da palavra.

No primeiro caso, o sistema representa o contexto da cada palavra pela probabilidade da palavra ocorrer como cada *part-of-speech*, com todas as probabilidades do vetor somando 1. No caso do exemplo (e para simplificar, omitindo as freqüências com o valor 0.0):

```
preposition(1.0) article(1.0) noun(0.8)/verb(0.2) .
     pronoun(1.0) verb(1.0) noun(0.1)/verb(0.9)
```

Essa probabilidades são obtidas a partir do léxico, no segundo caso, para cada possível *part-of-speech*, o vetor teria o valor de 1 para a palavra caso ela pudesse assumir aquela classe gramatical, e o valor 0 se não puder. Neste caso a soma de todos os itens no vetor não é pré-definida, como é o caso das probabilidades a priori, considerando o exemplo:

```
preposition(1) article(1) noun(1)/verb(1) .  pronoun(1) verb(1)
                        noun(1)/verb(1)
```

No sistema Satz a distribuição das palavras e suas respectivas classes gramaticais devem ser conhecidas a priori. Quando o token sendo analisado é uma palavra desconhecida, ou seja que não está presente no léxico, para atribuir sua freqüência de *part-of-speech* uma das heurísticas abaixo é utilizada:

- Uma palavra desconhecida contendo números é assumida como sendo um número.
- Qualquer *token* começando com um ponto, ponto de exclamação ou interrogação é atribuído a classe de *possible end-of-sentence punctuation tag*. Isso obtém seqüências como “?!” e “...”.
- Palavras que contem um hífen são atribuídas uma série de *tags* e freqüências distribuídas igualmente entre adjetivo, substantivo e nome próprio.
- Palavras contendo um ponto interno são assumidas como sendo abreviações.
- Uma palavra começando com maiúscula nem sempre é um nome próprio, mesmo quando aparece em outra posição que não o começo da sentença. Essas palavras não estão presentes no léxico e são atribuídas uma probabilidade dependente da linguagem (0.9) no caso do inglês de serem um nome próprio e o restante é distribuído uniformemente entre adjetivo, substantivo , verbo e abreviação. No caso de vetores binários todos os atributos teriam o valor 1.

noun	verb
article	modifier
conjunction	pronoun
preposition	proper noun
number	comma or semicolon
left parentheses	right parentheses
noun-punctuation char	possessive
colon or dash	abbreviation
sentence punctuation	others

Figura 2: As 18 Classes Gramaticais Utilizadas Pelo Sistema Satz

- Palavra em maiúsculas que aparecem no léxico mas não são registradas como nomes próprios ainda. Podem ser nomes próprios, em adição as freqüência de *part-of-speech* no léxico, essas palavras recebem uma probabilidade de serem um nome próprio (0.5) para o inglês em conjunto com as probabilidade que já foram atribuídas para a palavra, redistribuídas proporcionalmente. A proporção de palavras caindo nessa categoria varia muito dependendo do estilo do texto.
- E como último recurso a palavra recebe uma serie de *tags* como substantivo, verbo, adjetivo, e abreviação com uma freqüência distribuída uniformemente.

A próxima etapa é a construção dos vetores de contexto, para cada *token* do texto de entrada um veto de contexto é construído. O léxico pode conter várias *tags* específicas, mas primeiro é necessário mapear todas elas em categorias mais gerais.

Por exemplo, as funções sintáticas de Advérbio Conectivo Subordinativo e Advérbio Relativo Subordinativo seriam mapeadas para a categoria mais geral "advérbio". As freqüências de *part-of-speech* retornadas pelo *lookup module* são então mapeadas nas 18 categorias gerais mostradas na figura 2, e as freqüências de cada categoria são somadas, no caso do vetor de probabilidades, cada uma das 18 categorias da palavra são então convertidas para probabilidade dividindo as freqüências de cada uma pelo total. Para um vetor binário, todas as categoria não zero são atribuídas o valor de 1 e todas as outros são zero. Em adição ao 18 categorias, o vetor de contexto também contem duas flags adicionais que indicam se a palavra começa com uma letra maiúscula e se ela é seguida de um possível final de sentença. No total cada vetor de contexto é constituído de 20 características.

Cada uma das categorias é mapeada em uma das 18 categorias mais gerais como pode ser visto na Figura 2. Cada vetor de contexto contém 20 itens: sendo 18 itens referentes as freqüências de *part-of-speech* das palavras, se a palavra começa com maiúscula, e se o símbolo ocorre apos um sinal de possível final de sentença.

Para adaptar o Satz para o português, o primeiro passo foi construir o léxico com base no corpus que seria utilizado, para isso foi necessário alterar um dos arquivos do sistema mapeando as tags do Corpus para as 18 classes gramaticais apresentadas na figura 2. Para criar o léxico contendo as informações de *part-of-speech* de cada palavra, foi necessário criar um programa que lesse o corpus e automaticamente gerasse o arquivo utilizado pelo Satz.

Feitas essas mudanças foi encontrando um grande problema para realizar os ex-

perimentos. Na etapa de Tokenização o sistema gera os tokens, porém ele não reconhece palavras que possuem acentos, por exemplo a palavra agrícola seria dividida pelo tokenizador em duas palavras: agr e cola.

A solução para esse problema seria re-implementar o tokenizador do método de Palmer para que seja possível adaptar o Satz para documentos em português.

3. Resultados Computacionais

Para realizar os experimentos foi utilizada uma versão do Corpus Lacio-WEB [1] que contem 21.822 sentenças. No intuito de descobrir qual foi o real ganho com uso dos métodos analisados na seção 2, foi realizado um experimento para verificar qual seria o valor de referência mínima (*baseline*) do Corpus. Ou seja, como proposto por Palmer [8], foi realizado um experimento considerando todos os pontos naturais do documento, ou seja, todas as sentenças foram identificadas usando o único critério simples, que seria o ponto (.). O resultado encontrado é que o valor de referência mínima do texto é de 88.82%.

Para verificar quão robusto é cada um dos sistemas, isto é, seu desempenho usando sua configuração original, sem realizar alterações nas expressões regulares ou treinamento utilizando documentos em português, foi realizado um experimento dando como entrada para os sistemas o Corpus Lacio-Web com as sentenças não identificadas. Os resultados obtidos por cada sistema podem ser vistos na tabela 2, onde:

- Precisão (*Precision*) indica a porcentagem de finais de sentenças que foram corretamente identificados em relação aos que foram identificados pelo sistema (número de finais de sentenças corretamente identificadas / número de finais de sentenças identificadas);
- Cobertura (*Recall*) indica a porcentagem de finais de sentença que foram corretamente identificados em relação aos finais de sentença reais presentes na base (número de finais de sentenças corretamente identificadas / número de finais de sentenças reais na base);
- F-measure é uma medida de eficiência do sistema, combinando precisão e cobertura em uma medida única.

Finalmente, de forma à avaliar a performance de cada sistema, eles foram adaptados para o português fornecendo as informações necessárias sobre o idioma. O sistema MxTerminator foi treinado utilizando o Corpus Lacio-Web com validação-cruzada fator 10 (*ten fold cross-validation* [7]). Ao sistema baseado em expressões regulares foram adicionadas 240 novas expressões regulares contendo abreviações em português. Os resultados obtidos por cada sistema podem ser vistos na tabela 3.

Tabela 2: Robustez dos métodos de Separação de Sentenças

Método Utilizado	Precisão	Cobertura	F-measure
BaseLine	85,40%	92,25%	88,82%
Expressões Regulares	91,80%	88,02%	89,91%
MxTerminator	94,29%	95,84%	95,07%

Tabela 3: Performance dos métodos de Separação de Sentenças

Método Utilizado	Precisão	Cobertura	F-measure
BaseLine	85,40%	92,25%	88,82%
Expressões Regulares	91,80%	88,02%	89,91%
MxTerminator	96,31%	96,93%	96,62%

Apesar de terem sido adicionadas 240 novas expressões regulares ao sistema de expressões regulares a performance do sistema não melhorou, o sistema MxTerminator ao ser treinado com os documentos em português apresentou uma melhora significativa, visto que 1% de acerto médio na base significa que 200 sentenças foram classificadas corretamente.

4. Conclusões e Direções Futuras

Neste trabalho foi realizado um experimento analisando os diferentes métodos para realizar a tarefa de segmentação textual automática em sentenças de documentos em português. Foram comparadas a performance e a robustez de dois sistemas diferentes, um baseado em regras fixas na forma de expressões regulares e um sistema baseado em algoritmo de aprendizagem de máquina (MxTerminator). Infelizmente devido a forma como o parser do sistema Satz foi implementado não foi possível utiliza-lo neste trabalho para realizar as comparações.

Como foi visto na seção anterior os resultados obtidos pelo sistema MxTerminator são surpreendentes, pois ele é ao mesmo tempo robusto, fácil de ser treinado, e obtém bons resultados. Já o método baseado em expressões regulares, mesmo considerando o contexto aonde ocorre cada ponto, pode ser considerado inadequado para a tarefa proposta, no caso de documentos em português, visto que ele obteve resultados piores que o MxTerminator e deu um trabalho muito maior para ser adaptado para o novo idioma e mesmo após ser adaptado para o novo idioma, não melhorou sua performance.

Como trabalho futuro os autores deste trabalho estão re-implementando o método de construção de vetores de contexto do método de Palmer, para ser possível realizar uma comparação entre este método e os demais métodos apresentados neste trabalho. Assim seria possível verificar qual dos métodos seria o mais adequado para realizar a tarefa de segmentação textual automática para documentos em português, pois apesar do Satz precisar do léxico com a freqüência de *part-of-speech* das palavras, dependendo dos resultados, ele pode ser considerado apto a tarefa alvo, e a decisão de utilizar o Satz ou o MxTerminator pode ser baseada em possuir ou não um léxico, mas para isso mais experimentos são necessários.

Referências

[1] S. M. Aluisio, G. M. Pinheiro, Finger, M. G. V. Nunes, and S. E. Tagnin. The lacio-web project: overview and issues in brazilian portuguese corpora creation. In *Proceedings of the Corpus Linguistics 2003*, volume 16, pages 14–21, 2003.

[2] E. Brill. A simple rule-based part-of-speech tagger. In *Proceedings of ANLP-92, 3rd Conference on Applied Natural Language Processing*, pages 152–155, Trento, IT, 1992.

[3] E. Brill. Some advances in transformation-based part-of-speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, volume 1, pages 722–727, 1994.

[4] J. Larocca Neto, A. D. Santos, C. A. A. Kaestner, and A. Freitas. Generating text summaries through the relative importance of topics. In *Proc. Int. Joint Conf.: IBERAMIA-2000 (7th Ibero-American Conf. on Artif. Intel.) & SBIA-2000 (15th Brazilian Symp. on Artif. Intel.)*, pages 301–309, Sao Paulo, SP, Brazil, November 2000.

[5] J. Larocca Neto, A. D. Santos, C. A. A. Kaestner, and A. A. Freitas. Document clustering and text summarization. In *Proc. 4th Int. Conf. Practical Applications of Knowledge Discovery and Data Mining (PADD-2000)*, pages 41–55, London: The Practical Application Company, 2000.

[6] I. Mani, D. House, G. Klein, L. Hirschman, L. Obrsl, T. Firmin, M. Chrzanowski, and B. Sundheim. The tipster summac text summarization evaluation. MITRE Technical Report MTR 98W0000138, The MITRE Corporation, October 1998.

[7] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[8] D. D. Palmer. SATZ - an adaptive sentence segmentation system. Master's thesis, 1994.

[9] D. D. Palmer and M. A. Hearst. Adaptive multilingual sentence boundary disambiguation. *Computational Linguistics*, 23(2):241–267, 1997.

[10] T. A. S. Pardo and M. G. V. Nunes. Segmentação textual automática: Uma revisão bibliográfica. Série de Relatórios Técnicos 185, Instituto de Ciências Matemáticas e de Computação - ICMC, Universidade de São Paulo, 2003.

[11] A. Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Conference on Empiracal Methods in Natural Language Processing*, pages 133–142, 1996.

[12] J. Reynar and A. Ratnaparkhi. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 16–19, 1997.

[13] M. D. Riley. Some applications of tree-based modelling to speech and language. In *Proceedings of the Speech and Natural Language Workshop*, pages 339–352, Cape Cod, MA, 1989.

[14] C. N. Silla Jr. and C. A. A. Kaestner. Estudo de métodos automáticos para sumarização de textos. In K. Weber and C. Kaestner, editors, *Simpósio de Tecnologias de Documentos*, pages 45–49, São Paulo, SP, Brazil, Setembro 2002.

[15] C. N. Silla Jr., J. D. Valle Jr., and C. A. A. Kaestner. Detecção automática de sentenças com o uso de expressões regulares. In *III Congresso Brasileiro de Computação*, Itajaí, SC, Brazil, Agosto 2003.

[16] D. J. Walker, D. E. Clements, M. Darwin, and J. W. Amtrup. Sentence boundary detection: A comparison of paradigms for improving mt quality. In *Proceedings of the 8th Machine Translation Summit*, Santiago de Compostela, Spain, 2001.