Investigação do Processo de *Stemming* na Língua Portuguesa

Reinaldo Viana Alvares

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre.

Área de concentração: Otimização Combinatória e Inteligência Artificial.

Orientadora: Ana Cristina Bicharra Garcia

Niterói, março de 2005.

Investigação do Processo de *Stemming* na Língua Portuguesa

Reinaldo Viana Alvares

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre.

Área de concentração: Otimização Combinatória e Inteligência Artificial.

Aprovada por:

Profa. Ana Cristina Bicharra Garcia / IC-UFF (Presidente)

Profa. Solange Oliveira Rezende / ICMC-USP

Prof. Miguel Pari Soto / UFF

Niterói, março de 2005.

Dedico este trabalho aos meus Pais,

Amilcar e Terezinha.

Agradecimentos

A Deus, que forneceu todo o cenário necessário e suficiente para a realização completa deste curso. Aprendi que é comum e inevitável ter que enfrentar situações favoráveis e desfavoráveis. Estas últimas, transformadas em motivação, contribuiram para minha formação e amadurecimento.

Aos funcionários do CCBEU de Belém, por todo apoio e confiança prestados quando da minha decisão de arriscar tudo em função do mestrado. Aos professores da UNAMA e UFPA, pela confiança depositada. Ao amigo Silvio Andrade.

Ao Pedro, pela hospedagem e conselhos dispensados ao longo dos dez primeiros meses. À Equipe da República 2305: Heitor, Fabiano, Fábio, Leonardo, Marcelo, Marcio e Paulinho. Sem sombra de dúvidas, esta convivência foi uma experiência única.

Em especial, à Nathielly, minha namorada, pessoa com quem tive oportunidade de compartilhar momentos das mais diversas naturezas, dos fáceis aos adversos.

Aos amigos: André, Cristiane, Ilza, Idalmis, Melba, Adelaide, Geyza, Solange, José Carlos, José Viterbo, José Wilson, Cristiano, Áthila, Stênio, Walber, Fabrício e Emerson. Aos professores do mestrado e aos colegas da UNISUAM, pelo apoio e valiosas dicas. À equipe da Academia Brasileira de Letras, em especial ao Professor Claudio Sobrinho. À professora Maria Cristina Bessa Moreira, do Departamento de Estatística da UFF.

À minha Orientadora, Professora Ana Cristina, pela paciência e por toda con-

vivência. Aos membros da Banca, Professor Miguel e Professora Solange.

Resumo da Tese apresentada à UFF como parte dos requisitos necessários para a obtenção do grau de Mestre em Computação (M.Sc.)

Investigação do Processo de *Stemming* na Língua Portuguesa

Reinaldo Viana Alvares

março - 2005

Orientadora: Ana Cristina Bicharra Garcia

Programa: Pós-Graduação em Computação

O processo de busca e recuperação de informação é uma tarefa rotineira do ser humano, no entanto, de complexa automatização. Isto ocorre pois a qualidade dos resultados é muitas vezes relacionada com o grau de satisfação do usuário, um parâmetro de difícil mensuração. Em geral esta qualidade é avaliada levando-se em consideração um conjunto de consultas realizadas em uma coleção de textos, e as respostas relevantes obtidas.

Comumente, duas medidas de avaliação são utilizadas neste processo: *precision*, que representa a proporção de itens relevantes recuperados do total de itens recuperados; e *recall*, que representa a proporção de itens relevantes recuperados do total de itens relevantes da coleção.

Para isso, um dos desafios é encontrar formas eficientes para representar os documentos, de maneira a evitar ambigüidade. Uma alternativa para resolver este problema consiste em obter uma representação única para palavras que apontem para um mesmo conceito. Esta tarefa pode ser definida como *stemming*.

O processo de *stemming* muitas vezes é atrelado à estrutura morfológica do idioma onde é utilizado. Em se tratando da língua portuguesa, foram encontradas poucas soluções para atender a demanda por esses tipos de algoritmos.

A complexidade morfológica da língua portuguesa e as poucas soluções de *stemming* encontradas para este idioma, serviram como motivação para o desenvolvimento desta dissertação.

Este trabalho apresenta um modelo para algoritmos de *stemming*, aplicável à língua portuguesa, baseado num estudo estatístico realizado em uma coleção de palavras extraídas da *Web* brasileira.

Com objetivo de avaliar o modelo, um *stemmer* é implementado e comparado com uma solução encontrada na literatura, especialmente desenvolvida para este idioma. As principais contribuições deste trabalho são o modelo sistemático para o processo de *stemming*, além do *stemmer* concebido e implementado especialmente para a língua portuguesa.

Abstract of Thesis presented to UFF as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

Stemming Process Investigation for the Portuguese Language

Reinaldo Viana Alvares

março - 2005

Advisor: Ana Cristina Bicharra Garcia
Department: Computer Science

The information retrieval process is a usual task for the human. However, having a complex automation. This happens because the quality of the results is often related with the degree of the user's satisfaction, a difficult parameter to measure. In general this quality is evaluated being taking into account a group of queries in a text collection, and their relevant answers.

Commonly, two evaluation measures are used in this process: the first is the precision, wich represents the proportion of recovered relevant items from the total of recovered items; and the second is the recall, wich represents the proportion of recovered relevant items from the total of relevant items of the collection.

One of the challenges is to find efficient forms to represent the documents, in order to avoid ambiguity. An alternative to solve this problem consists of obtaining a unique representation for words that appear for a same concept. This task can be defined as stemming.

Many times, the stemming process is dependent to the morphologic structure of the target language. For the Portuguese language, there were found few solutions to assist the demand for these algorithms.

The morphologic complexity of Portuguese language, and the few stemming solutions found for this language, were the motivation for the research shown in this

work.

This work presents a new model for the stemming process, that is applicable to the Portuguese language, based on a statistical study accomplished in a collection of extracted words of the Brazilian Web.

With objective of evaluating the model, a stemmer is implemented and compared with a solution found in the literature, especially developed for Portuguese. The main contributions of this work are the systematical model for the stemming process, besides the stemmer conceived and implemented specially for the Portuguese language.

Palavras-chave

1. Algoritmos de *stemming*

2. Processamento de Linguagem Natural

3. Mineração de texto

Glossário

LexWeb	:	*Léxico do Português da Web*;
OI	:	*Overstemming Index* - Índice de erro de *overstemming*;
RI	:	*Recuperação de Informação*;
RSLP	:	*Removedor de Sufixos da Língua Portuguesa*;
SGBD	:	*Sistema Gerenciador de Banco de Dados*;
SQL	:	*Structured Query Language* - Linguagem de Consulta Estruturada;
SW	:	*Stemming Weight* - Força do *stemmer*;
UI	:	*Understemming Index* - Índice de erro de *understemming*;

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

A área de Recuperação de Informação (RI) trata de caracterizar soluções e desenvolver algoritmos para resolução de problemas que envolvem a manipulação de grandes quantidades de texto.

Em um documento sobre música é comum, por exemplo, que sejam encontradas palavras tais como **musicalização**, **músico** e **músicas**. Essas palavras representam de forma genérica o significado **música**. Além disso, elas são diferenciadas graficamente apenas por variações afixais (prefixos e/ou sufixos).

Há casos entretanto, onde palavras parecidas graficamente possuem significado totalmente diferentes. Por exemplo: e**minente**/i**minente** e **cole**ga/**cole**tar.

Dessa forma, o problema da RI aponta para a tarefa de identificar a subcadeia de uma palavra que sirva como uma representação única e não ambígüa da mesma, e a de suas diversas variações, provendo ao usuário um resultado de busca mais abrangente. Esta subcadeia é comumente denominada *stem*.

Algoritmos que manipulam palavras a fim de extrair o *stem* delas são conhecidos como algoritmos de *stemming*.

Assim, *stemming* pode ser definido como o processo de transformar formas variantes das palavras para uma representação concisa e precisa, denominada *stem*.

A idéia do processo de *stemming* segue o Princípio da Anotação [37], no qual o objetivo é mapear, para uma forma unívoca, objetos que possuem o mesmo significado. Neste caso, o *stem* deve possuir generalidade suficiente que permita capturar a essência das palavras e a de suas diversas variações.

Observa-se que não existe unanimidade em relação ao conceito de *stem*. Alguns autores consideram o próprio radical morfológico da palavra [3, 6, 8]. Outros, entretanto, diferenciam [15, 17, 21, 22, 35].

Em [15], o *stem* é definido como a parte essencial do item lexical, semelhante ao radical, obtida pelo processo de *stemming*. No trabalho realizado em [17], o *stem* deve ser capaz de capturar o significado da palavra, sem perder muito detalhe. Já em [21], o *stem* é considerado uma pseudo-raiz da palavra, resultante da aplicação de um algortimo de *stemming*. Em [22], o *stem* é visto como uma forma mais geral da palavra.

Finalmente, em [35], o *stem* é considerado como um termo mais genérico da palavra, resultante do processo de *stemming*.

No contexto deste trabalho, o *stem* não necessariamente é igual à raiz lingüística da palavra. Apesar disso, é possível que muitas vezes tanto um quanto outro possam ter a mesma forma gráfica. A idéia central é que o *stem* represente uma denotação mínima não ambígua do termo. Para isso em geral o *stemmer* - programa de computador que executa o processo de *stemming* - retira algumas letras no início e/ou no final das palavras, assumindo que a cadeia de caracteres restante ainda represente a essência delas.

Atualmente, há duas vertentes quando da implementação de *stemmers*: uma estatística, que não leva em conta as particularidades do idioma; e a outra, dependente das características morfológicas do idioma.

No processo de *stemming*, há dois tipos de erros que podem aparecer: *overstemming* e *understemming*. O primeiro representa o processo de retirar mais letra do que o devido, permitindo que palavras com significados diferentes apontem para o mesmo *stem*. Por exemplo, se for gerado o *stem* **cole** para as palavras **colegiado** e

coletivo. Já o *understemming* ocorre quando letras são deixadas a mais, permitindo que palavras com mesmo significado sejam mapeadas para *stems* diferentes. Por exemplo, se forem gerados os *stems* **planejam** e **planejav**, respectivamente, para as palavras **planejamento** e **planejavam**.

Algoritmos de *stemming* foram originalmente propostos para a língua inglesa [28, 33]. Com o passar do tempo, surgiram soluções para diversos idiomas, tais como: árabe [26], búlgaro [31], espanhol [11], holandês [24], persa [22] e sueco [5]. Para a língua portuguesa, foi encontrada uma solução adaptada da língua inglesa - avaliada em [20] - e um *stemmer* desenvolvido especialmente para esse idioma [17].

1.1 A Dissertação

A constatação de que as soluções de *stemming* atuais para a língua portuguesa não atendem bem a demanda por esses tipos de algoritmos e a própria complexidade desse idioma serviram como motivação para o desenvolvimento desta dissertação.

A hipótese deste trabalho é que a construção de um *stemmer* que realize um tratamento sufixal hierarquizado provê resultados mais precisos em relação aos *stems* gerados. Para isso, um estudo de caso é realizado, envolvendo três métodos de avaliação.

A principal contribuição desta dissertação consiste na apresentação de um novo *stemmer*, especialmente projetado para a língua portuguesa. Para isso, é definido um modelo, denominado STEMBR, o qual especifica três fases que resultam na geração dos *stems*.

1.2 Organização do Texto da Dissertação

A dissertação está organizada da seguinte forma:

O Capítulo 2 contextualiza o processo de *stemming* como uma fase da Mineração de Texto, apresenta os principais métodos utilizados para avaliação desses algorit-

mos, expõe as abordagens mais comuns usadas na construção de *stemmers*, além de caracterizar a solução que será comparada com o *stemmer* desenvolvido neste trabalho.

O Capítulo 3 descreve o STEMBR: um modelo para o processo de *stemming* na língua portuguesa.

O Capítulo 4 apresenta um estudo de caso, envolvendo o STEMBR e a solução encontrada na literatura.

Encerrando o trabalho, o Capítulo 5 traz conclusões e apresenta sugestões para trabalhos vindouros.

Capítulo 2

Algoritmos de *Stemming*

2.1 Preâmbulo

Este capítulo contextualiza o processo de *stemming* como uma etapa da Mineração de Texto, expõe e exemplifica os principais métodos utilizados para avaliação de algoritmos de *stemming* e apresenta as abordagens mais comuns usadas no processo, destacando as principais soluções existentes. Além disso, descreve as características do *stemmer* encontrado na literatura, desenvolvido especificamente para a língua portuguesa e utilizado no estudo de caso desta dissertação.

2.2 O Processo de *Stemming* no Contexto da Mineração de Texto

A Mineração de Texto refere-se ao processo de extração de padrões interessantes e não triviais a partir de textos. Atualmente, esta tecnologia está sendo empregada em diversas aplicações, por exemplo, para extração automática de resumos, categorização automática de mensagens de correio eletrônico, entre outras.

Costuma-se dividir o processo de Mineração de Texto três grandes etapas. Um esquema dessas etapas é mostrado na Figura 1.

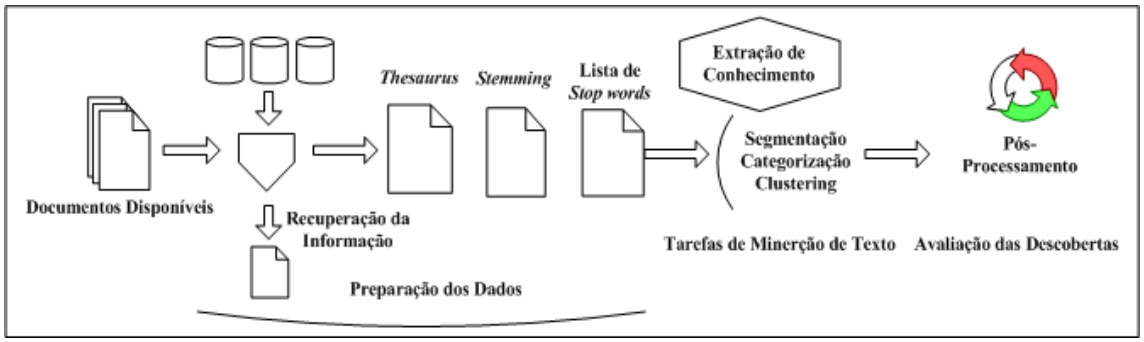

Figura 1: Etapas do Processo de Mineração de Texto (adaptada de [10])

As principais etapas deste processo são:

- Preparação dos dados;
- Extração de conhecimento, e;
- Pós-processamento.

O processo de Mineração de Texto é iniciado com a compreensão do domínio da aplicação e o estabelecimento dos objetivos a serem obtidos. Neste momento, as questões em potencial para a mineração são identificadas. Dependendo do problema a ser minerado e da massa de documentos disponível, haverá a escolha do tipo de tarefa a ser trabalhada.

A fase de preparação dos dados objetiva preparar o conjunto de dados textuais que servirá como entrada para a etapa de extração de conhecimento. Este conjunto de dados deve representar a maior quantidade possível de características relevantes dos documentos.

Em geral, a fase de pré-processamento dos dados é constituída por três tarefas:

1. uso de *thesaurus*;
2. aplicação de *stemming*, e;

3. eliminação de *stop words*.

Um *thesaurus* (dicionário) pode ser definido como um vocabulário controlado que representa hierarquias, sinônimos e relacionamentos associativos entre termos, com o objetivo de ajudar os usuários a encontrar a informação de que eles necessitam. O objetivo do *thesaurus* é substituir todos os termos que possuem o mesmo significado para um termo comum. Outro objetivo é substituir de hierarquias de conceitos, onde todos os termos são generalizados para termos de mais alto nível de acordo com a hierarquia de conceitos descrita no *thesaurus*.

A vantagem do processo de *stemming* como parte da etapa de pré-processamento dos dados, ocorre pelo fato do mesmo conseguir identificar similaridades em função da morfologia das palavras, dessa forma reduzindo o número de atributos de um texto, pois muitas vezes palavras com morfologias semelhantes representam de forma genérica o mesmo conceito.

Stop words são palavras que de maneira geral não constituem informação relevante para o contexto geral do texto. Na maioria das vezes essas palavras são removidas do texto quando da fase de pré-processamento dos dados. É comum que esta tarefa seja controlada pelos algoritmos de *stemming*. Para isto, eles gerenciam uma lista de palavras (*stoplist*), identificando se a palavra a ser processada é ou não uma *stop word*, realizando o procedimento de *stemming* apenas se ela não o for.

Apesar do consenso existente a respeito do fato de que essas palavras não apresentam grande importância para a formação do sentido geral do texto, em [34] é apresentada pesquisa, utilizando textos na língua inglesa, onde foi constatado que algumas formas de substantivos no singular e plural, preposições e algumas formas verbais são resposáveis por produzir resultados totalmente diferentes quando do uso dessa técnica na tarefa de classificação de textos.

Em geral, algumas classes de palavras, tais como por exemplo artigos, preposições, pronomes e interjeições, representam *stop words*. A lista a seguir apresenta algumas *stop words* para a língua portuguesa:

- a, à, adeus, agora, aí, ainda, além, algo, algumas, alguns, ali, ano, anos, antes,

ao, aos, apenas, apoio, após, aquela, aquelas, aquele, através;

- baixo, bastante, bem, boa, boas, bom, bons, breve;
- cá, cada, catorze, cedo, cento, certamente, certeza, cima, cinco, coisa, com, como, conselho, contra, custa;
- da, dá, dão, daquela, daquelas, daquele, daqueles, dar, das, de, debaixo, demais, dentro, depois, desde, dessa;
- e, é, ela, elas, ele, eles, em, embora, entre, era, és, essa, essas, esse, esses, esta, está, estão, estar, estas, estás.

Após o pré-processamento, surge a etapa de extração de conhecimento, através da execução das tarefas de mineração. Neste caso, as tarefas representam classes de problemas a serem solucionados. Uma tarefa típica existente é a classificação, que consiste em examinar características de um texto e atribuir a ele uma classe pré-definida.

Finalmente, após a extração de conhecimento, há a etapa de pós-processamento, que ajuda na avaliação das descobertas. Esta etapa objetiva a validação dos resultados obtidos, para que os mesmos possam ser efetivamente empregados.

2.3 Métodos de Avaliação

Há várias maneiras de avaliar o desempenho de algoritmos de *stemming*. No entanto, quando o processo de *stemming* é visto apenas como uma etapa da Mineração de Texto ou da RI, muitas vezes o que é avaliada é a qualidade dos resultados de cada tarefa, em geral com e sem a utilização desses algoritmos.

A investigação do uso de algoritmos *stemming* junto à tarefa de classificação de textos no idioma holandês é realizada em [13]. Verificou-se que neste caso o uso de *stemming* não contribui de forma consistente para o aumento da acurácia da tarefa.

Um trabalho comparando o uso de *stemming* e o de lematização na tarefa de

clusterização de textos em documentos no idioma finlandês é realizado em [23]. Lematização é a representação da palavra no masculino singular para substantivos e adjetivos, e infinitivo para os verbos. Verificou-se que para esta tarefa, o uso de lematização é mais apropriado que o de *stemming*.

Em geral, os sistemas de RI são avaliados através do uso de duas métricas: *precision* e *recall* [14]. Essas métricas são obtidas através das Fórmulas 1 e 2, respectivamente:

$$Precision = \frac{(Dr \cap Dq)}{Dr}, \text{ e} \tag{1}$$

$$Recall = \frac{(Dr \cap Dq)}{Dq}, \text{ onde} \tag{2}$$

- Dr representa o número de documentos recuperados pelo sistema para uma consulta $\mathbf{q}$;
- Dq representa o número de documentos relevantes a uma consulta $\mathbf{q}$; e
- $(Dr \cap Dq)$ equivale ao número de documentos relevantes recuperados.

Há diversos trabalhos comparando as métricas *precision* e *recall* junto ao uso de algoritmos de *stemming* [2, 5, 22, 25].

Em [2], é apresentado um *stemmer*, baseado em uma técnica chamada *light-stemming*, desenvolvida especialmente para o idioma árabe. Nesse idioma, diversas palavras com a mesma raiz possuem significados totalmente diferentes, ocasionando erros de *stemming*. Na abordagem proposta, são removidos apenas os prefixos e sufixos mais freqüentes.

A análise de um *stemmer* para o idioma sueco é encontrada em [5], onde foram realizados testes em um sistema de RI com e sem a utilização de *stemming*. Neste caso foi verificado aumento de 15% na métrica *precision* e 18% na métrica *recall*. O algoritmo obtém o *stem* da palavra por meio da execução de quatro passos, onde em cada um há a tentativa de retirar um sufixo da palavra.

Em [22], é apresentado um *stemmer* desenvolvido especialmente para o idioma persa. Esse *stemmer*, similar ao algoritmo de Porter [33], trabalha com um conjunto de regras para a extração do *stem* das palavras. Além disso, prefixos são tratados. Os testes envolveram a utilização de um sistema de RI com e sem *stemming*. Foi verificado aumento de 18% na métrica *precision*.

Já em [25], uma análise da métrica resposta em textos do idioma holandês é descrita. Neste caso, notou-se que o processo de *stemming* é benéfico em termos da métrica *recall* sem perda significante na métrica *precision*.

A avaliação da qualidade dessas tarefas quando do uso de algoritmos de *stemming* não provê, aos desenvolvedores, informação que possa trazer benefícios em termos do projeto desses algoritmos.

No entanto, os próximos métodos que serão apresentados ajudam o projetista a avaliar melhor as particularidades do *stemmer*, visto que eles estão relacionados com características inerentes aos *stems* gerados. Os métodos são:

- método manual;
- redução do vocabulário, e;
- método de Paice.

O mecanismo de funcionamento de cada um será ilustrado baseado no exemplo a seguir.

Considere uma amostra composta por 10 palavras e, **SI** e **SII** dois *stemmers* diferentes.

Na Tabela 1, os campos correspondentes à coluna **Palavra** representam a amostra; os referentes à coluna ***Stem*** representam o *stem* convencionado como correto para cada palavra; e os referentes às colunas **SI** e **SII** representam os *stems* gerados pelos *stemmers* **SI** e **SII**, respectivamente.

Palavra	***Stem***	**SI**	**SII**
bebe	beb	beb	bebe
bebezinhos	bebe	beb	beb
bebezinho	bebe	beb	be
bebida	beb	bebid	beb
comparação	compar	comp	comp
comparar	compar	comp	compara
computacionalmente	comput	comp	comput
computador	comput	comput	comput
computadores	comput	comput	comp
computar	comput	comput	comp

Tabela 1: Exemplo do Processo de *Stemming*

2.3.1 Método Manual

A Figura 2 ilustra o mecanismo de funcionamento do método manual.

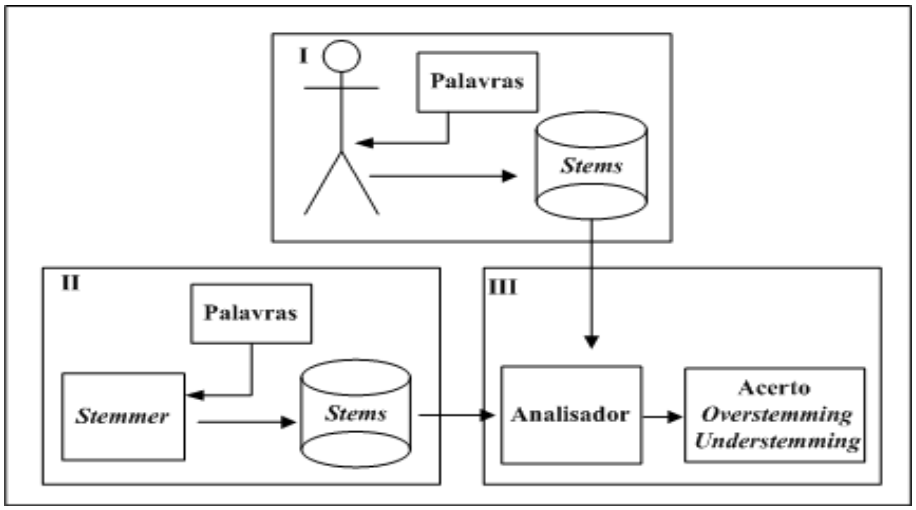

Figura 2: Etapas do método manual

Na etapa **I**, um ser humano realiza o processo de *stemming* de forma direta, convencionando o *stem* correto para cada palavra. Cada *stem* serve como parâmetro

de comparação com o *stem* respectivo, gerado pelo *stemmer* sendo analisado. Na etapa **II**, o *stemmer* é executado, sendo gerados os *stems* das palavras. Finalmente, na etapa **III** são obtidas três medidas para avaliação:

- número de acertos, contabilizado quando o *stem* gerado for igual ao *stem* convencionado;
- número de erros de *overstemming*, contabilizado quando o *stem* gerado for menor (número de caracteres) que o convencionado, e;
- número de erros de *understemming*, contabilizado quando o *stem* gerado for maior (número de caracteres) que o convencionado.

Os dados da Tabela 2, representam o resultado do método manual referente ao exemplo apresentado na Tabela 1.

Stemmer	**Acertos**	***Overstemming***	***Understemming***
SI	4	5	1
SII	3	5	2

Tabela 2: Resultado do método manual para o exemplo

Neste caso, o *stemmer* **SI** obteve um acerto a mais que o *stemmer* **SII**. Ambos obtiveram a mesma quantidade de erros de *overstemming* e o *stemmer* **SII** obteve um erro de *understemming* a mais que o *stemmer* **SI**.

2.3.2 Redução do Vocabulário

A redução do vocabulário, ou Taxa de Redução(Tr) a *stems*, é obtida por meio da Fórmula 3:

$$\mathrm{Tr} = \frac{Np}{Qtde}, \text{ no qual} \tag{3}$$

- Np representa o número de palavras do corpus, e;
- $Qtde$ representa a quantidade de *stems* gerados, excluindo-se as repetições.

Os dados da Tabela 3, representam a taxa de redução do vocabulário para o exemplo apresentado na Tabela 1:

Stemmer	***Stems***	**Redução**
SI	4	40%
SII	6	60%

Tabela 3: Resultado do método redução do vocabulário para o exemplo

De maneira geral, o melhor *stemmer* é o que consegue obter maior redução do vocabulário. Neste caso, o *stemmer* **SI** é melhor, pois o mesmo conseguiu gerar dois *stems* a menos que o *stemmer* **SII**.

2.3.3 Método de Paice

Descrição do método

Paice [32] propôs um método para avaliação de algoritmos de *stemming* no qual três medidas foram introduzidas: *overstemming index* (OI), *understemming index* (UI) e *stemming weight* (SW). O método requer uma amostra de palavras, sem repetição, particionada em grupos conceituais, onde as palavras de cada grupo são semântica e morfologicamente relacionadas. Dessa forma, desconsiderando homógrafos, são definidas duas classes de relacionamentos entre as palavras:

- **Classe 0**: Palavras diferentes em forma, mas semanticamente equivalentes, e;
- **Classe 1**: Palavras diferentes em forma e semanticamente distintas.

Um bom algoritmo de *stemming* é aquele que na medida do possível gera apenas um *stem* para palavras pertencentes à **classe 0**, e evita gerar o mesmo *stem* para palavras pertencentes à **classe 1**.

Neste método, para cada grupo conceitual, dois totais são calculados:

- *Desired Merge Total* (DMT), que representa o número possível de pares diferentes formados por palavras do mesmo grupo, obtido através da Fórmula 4:

$$DMTg = 0,5ng(ng - 1), \tag{4}$$

onde ng representa o número de palavras do grupo. Para grupos que possuem uma única palavra, o valor DMT é zero.

- *Desired Non-merge Total* (DNT), que representa o número de pares formados por uma palavra membro com outra não-membro do grupo, obtido através da Fórmula 5:

$$DNTg = 0,5ng(W - ng), \tag{5}$$

onde W representa a quantidade de palavras da amostra.

Somando-se os valores DMT de cada grupo, é obtido o GDMT (*Global Desired Merge Total*). Da mesma forma, somando-se os valores DNT de cada grupo, é obtido o GDNT (*Global Desired Non-merge Total*).

Após a execução do *stemmer* na amostra, um *stem* é gerado para cada palavra. Entretanto, há possibilidade de existirem *stems* diferentes em um mesmo grupo conceitual. A inabilidade do *stemmer* produzir o mesmo *stem* para determinado grupo é quantificada através do parâmetro UMT (*Unachieved Merge Total*), obtido através da Fórmula 6:

$$UMTg = 0,5 \sum_{i=1..s} ui(ng - ui), \tag{6}$$

onde s é o número de *stems* distintos no grupo e ui é o número de instâncias de cada *stem* no grupo. Novamente, somando-se os valores UMT de cada grupo, é obtido o GUMT (*Global Unachieved Merge Total*). O UI é calculado através da divisão GUMT/GDMT.

Podem existir casos em que o mesmo *stem* tenha sido gerado para palavras pertencentes a dois ou mais grupos conceituais diferentes, resultando em erros de *overstemming*. Para contabilizar esses erros, a amostra é reorganizada em grupos que compartilham o mesmo *stem*, e calculado o WMT (*Wrongly Merged Total*),

obtido através da Fórmula 7:

$$WMTg = 0,5 \sum_{i=1..t} vi(ns - vi), \tag{7}$$

onde t é o número de grupos conceituais que compartilham o mesmo *stem*, ns é o número de instâncias do *stem* e vi é o número de *stems* de cada grupo t. Somando-se os valores WMT, é obtido o GWMT (*Global Wrongly Merged Total*). O valor de OI é calculado através de: GWMT/GDNT. Finalmente, o valor SW é calculado através da divisão OI/UI.

Para um melhor entendimento do método, o mesmo será aplicado aos dados da Tabela 1, sendo comparados os resultados dos dois *stemmers* hipotéticos. Neste caso, o tamanho da amostra, sendo pequeno, serve para enfatizar melhor o mecanismo de funcionamento do método.

Analisando as palavras da Tabela 1, estas foram classificadas em 4 grupos semânticos, presentes na Tabela 4:

Grupo	**Palavra**
1 bebê	bebezinho, bebezinhos
2 bebida	bebe, bebida
3 comparar	comparação, comparar
4 computador	computacionalmente, computador, computadores, computar

Tabela 4: Grupos semânticos da amostra

Os dados da Tabela 5 representam os valores GDMT e GDNT:

Grupo	**DMT**	**DNT**
1	1	8
2	1	8
3	1	8
4	6	12
Total	9	36

Tabela 5: Resultado de GDMT e GDNT para a amostra

Após a execução do *stemmer* **SI**, tem-se a configuração mostrada na Tabela 6:

Grupo	***Stems***
bebê	beb, beb
bebida	beb, bebid
comparar	comp, comp
computador	comp, comput, comput, comput

Tabela 6: Execução do *stemmer* **SI** na amostra

Os dados da Tabela 7 representam o valor GUMT para o *stemmer* **SI**:

Grupo	**UMT**
1	0
2	1
3	0
4	3
GUMT	4

Tabela 7: Resultado de GUMT para o *stemmer* **SI**

Reorganizando a amostra em grupos que compartilham o mesmo *stem*, para o *stemmer* **SI**, tem-se a configuração da Tabela 8:

Grupo
beb(1), beb(1), beb(2)
bebid(2)
comp(3), comp(3), comp(4)
comput(4), comput(4), comp(4)

Tabela 8: Amostra reorganizada

Os dados da Tabela 9 representam o valor GWMT para o *stemmer* **SI**:

Grupo	**WMT**
1	2
2	0
3	2
4	0
GWMT	4

Tabela 9: Resultado de GWMT para o *stemmer* **SI**

Após a execução do *stemmer* **SII**, tem-se a configuração mostrada na Tabela 10:

Grupo	*Stems*
bebê	be, bebe
bebida	bebe, beb
comparar	comp, compara
computador	comput, comput, comp, comp

Tabela 10: Execução do *stemmer* **SII** na amostra

Os dados da Tabela 11 representam o valor GUMT para o *stemmer* **SII**:

Grupo	**UMT**
1	1
2	1
3	1
4	4
GUMT	7

Tabela 11: Resultado de GUMT para o *stemmer* **SII**

Reorganizando a amostra em grupos que compartilham o mesmo *stem*, para o *stemmer* **SII**, tem-se a configuração mostrada na Tabela 12:

Grupo
be(1)
beb(2)
bebe(1), bebe(2)
comp(3), comp(4), comp(4)
compara(3)
comput(4), comput(4)

Tabela 12: Amostra reorganizada

Os dados da Tabela 13 representam o valor GWMT para o *stemmer* **SII**:

Grupo	**WMT**
1	0
2	0
3	1
4	2
5	0
6	0
GWMT	3

Tabela 13: Resultado de GWMT para o *stemmer* **SII**

Finalmente, na Tabela 14 são mostrados os valores OI, IU e SW para os *stemmers*.

Stemmer	**OI**	**UI**	**SW**
SI	0,1111	0,4444	0,25
SII	0,0833	0,7777	0,11

Tabela 14: Resultado do Método de Paice para os *stemmers* **SI** e **SII**

Neste caso, verifica-se que:

- OI(SI)>OI(SII);
- UI(SI)<UI(SII), e;
- SW(SI)>SW(SII).

Observe que o índice de *overstemming* do *stemmer* **SI** é maior que o do *stemmer* **SII**. Já em relação ao índice de *understemming*, este apresenta valor maior para o *stemmer* **SII**.

Fazendo uma análise comparativa, pode-se concluir que:

- É sempre interessante avaliar o desempenho dos algoritmos de *stemming* utilizando mais de um método;

- O *stemmer* **SII** obteve mais erros de *understemming* que o *stemmer* **SI**. Este fato foi comprovado pelos métodos manual e de Paice;

- Pelo método de Paice, o *stemmer* **SI** obteve índice de *overstemming* maior que o *stemmer* **SII**. Já em relação ao método manual, eles permaneceram empatados. Na prática, isto ocorre pela própria natureza do cálculo desses erros. No método manual tal cálculo é realizado comparando o *stem* de cada palavra com o previamente convencionado. Já no método de Paice, esse erro é calculado baseado no grupamento das palavras que possuem o mesmo *stem*, resultando em erro de *overstemming* quando verificado que palavras com mesmo *stem* pertencem a grupos conceituais diferentes.

2.4 Algoritmos e Abordagens

Nesta seção, serão apresentadas as abordagens mais utilizadas na construção de algoritmos de *stemming*, a saber: dicionário, abordagens estatísticas e removedores de afixos.

2.4.1 Dicionário

Esta abordagem, exemplificada na Tabela 15, consiste em armazenar em alguma estrutura a palavra e o *stem* correspondente. Aplicações que utilizam esta solução são encontradas em [12, 13].

Palavra	***Stem***
computadores	comput
computador	comput
computar	comput

Tabela 15: Abordagem Dicionário no Processo de *Stemming*

Neste tipo de solução em geral os *stems* são gerados da forma mais precisa

possível, dado que muitas vezes um ser humano convenciona o *stem* correto para cada palavra.

No entanto, a abordagem dicionário é limitada à recuperação apenas das palavras que estejam previamente armazenadas. Além disso, o espaço de armazenamento tende a crescer com o aumento do corpus, o que pode tornar o processo de busca ineficiente.

2.4.2 Abordagens Estatísticas

O processo de *stemming* muitas vezes é atrelado a um considerável conhecimento linguístico quando do projeto dos *stemmers*. Atualmente, a maioria dos *stemmers* é dependente das características morfológicas do idioma utilizado. No entanto, já existem pesquisas que empregam uma metodologia diferente, evitando ao máximo a dependência das características morfológicas do idioma: essas pesquisas são baseadas em métodos estatísticos [4, 29].

Em [4] é apresentado um *stemmer* estatístico baseado em análise de *links*. O método considera que cada palavra é composta por duas partes: um prefixo (que será designado *stem*) e um sufixo. A partir de um conjunto de palavras o algoritmo obtém, para cada uma, as possíveis subcadeias, que serão candidatas a serem o *stem*. Os *stems* extraídos são as subcadeias que:

- possuem freqüência alta, e;
- formam palavras com os sufixos que possuem freqüência alta.

Na prática, prefixos que possuem freqüência alta são candidatos a serem *stems*, mas os mesmos são desconsiderados caso não apareçam junto a sufixos de freqüência alta.

Em [29] é apresentado um *stemmer* baseado no método n-grama. Um n-grama corresponde a uma seqüência de n caracteres de determinada palavra. Por exemplo, as palavras **computadores**, **computação**, **computadorizado** e **computacional**

compartilham o 6-grama **comput**. A idéia central neste caso parte do princípio de que família de palavras que representam o mesmo conceito em geral compartilham n-gramas.

A desvantagem deste método está no fato de que em cada palavra todo caracter inicia um n-grama. Assim, o número de n-gramas de um texto é muito superior ao número de palavras ou *stems*.

2.4.3 Removedores de Afixos

A abordagem removedores de afixos emprega meios para extrair o *stem* analisando cada palavra isoladamente, através da adição, remoção ou substituição de caracteres. Essas manipulações podem ser realizadas no início (prefixo) ou, mais freqüêntemente, no final (sufixo) das palavras.

Normalmente, *stemmers* que adotam a abordagem removedores de afixos para a extração de sufixos trabalham comparando a terminação da palavra a um conjunto de regras, com o objetivo de remover determinada seqüência de caracteres que é considerada como sendo o sufixo. No entanto, tendo em vista as particularidades morfológicas de cada idioma e a própria simplicidade desses algoritmos, é sempre esperado que haja uma margem de erro em relação aos *stems* gerados.

Exemplos de *stemmers* que utilizam essa técnica são encontrados em [5, 16, 22, 33].

Em [5], é apresentado um *stemmer* para o idioma sueco, que utiliza aproximadamente 150 regras. O mecanismo de funcionamento é baseado em quatro passos onde em cada um apenas uma regra é aplicada à palavra. Os passos desse *stemmer* são resumidos:

- Passo 1: são removidos sufixos terminados pela letra **s**;
- Passo 2: são removidos alguns sufixos de adjetivos e substantivos, além de formas verbais no pretérito e no particípio passado;

- Passo 3: são retirados os principais sufixos formadores de plural de substantivos e adjetivos;
- Passo 4: são tratadas palavras que são exceção às regras sufixais.

Em [16], é apresentado um método simples de *stemming* para a língua inglesa, pelo qual apenas uns poucos finais de palavras são removidos: **ies**, **es** e **s**, existindo exceções.

Um *stemmer* especialmente desenvolvido para o idioma persa é apresentado em [22], realizando tratamento de prefixos e sufixos. Na primeira etapa do algoritmo, é retirado o maior sufixo da palavra, através da consulta a uma lista de sufixos. Nesta solução, os sufixos estão divididos nos seguintes grupamentos:

- Sufixos designadores de formas verbais;
- Sufixos designadores de formas no plural;
- Sufixos designadores de possessivos;
- Sufixos designadores de outros substantivos, e;
- Outros sufixos;

Adicionalmente, nesse *stemmer* há tratamento de sufixos que fogem às regras gerais apresentadas.

O algoritmo clássico para extração de sufixos, denominado algoritmo de Porter, é encontrado em [33]. Esse *stemmer*, desenvolvido originalmente para a língua inglesa, é composto por cinco passos nos quais são aplicadas regras às palavras, onde são removidos os sufixos mais comuns. Baseado em uma métrica específica relacionada com o número de vogais-consoantes presentes em uma palavra, esse *stemmer* tenta evitar a remoção de letras caso o *stem* seja muito pequeno. Esse algoritmo foi adaptado para diversos idiomas.

2.5 Algoritmos de *Stemming* para a Língua Portuguesa

O processo de *stemming* para a língua portuguesa começou a ser desenvolvido recentemente, destacando-se a adaptação do algoritmo de Porter para esse idioma, e um *stemmer*, denominado RSLP (Removedor de Sufixos da Língua Portuguesa) [17], desenvolvido especialmente para o português. Esses *stemmers* são basedos na técnica removedores de afixos.

Em [20], é apresentada uma avaliação do algoritmo de Porter adaptado para a língua portuguesa, utilizando três bases de documentos de domínios e tamanhos distintos, onde dois tipos de experimentos são realizados: redução do vocabulário e avaliação dos erros de *overstemming* e *understemming*. Ainda neste trabalho, foram propostas alterações no algoritmo original, objetivando a diminuição de erros de *understemming*. As modificações realizadas foram:

- Inclusão de algumas formas verbais irregulares, e;
- Inclusão de algumas formas sufixais de verbos e nomes.

A utilidade das mudanças realizadas foi comprovada, pois em uma das bases utilizadas a taxa de *understemming* diminuiu cerca de 2%. Observou-se também que a diminuição de erros de *understemming* e *overstemming* em conjunto tende a ser um trabalho conflitante, pois em geral modificações realizadas para diminuição de um erro tendem a aumentar o outro tipo.

Já em relação ao *stemmer* desenvolvido especialmente para a língua portuguesa, este mostrou-se mais eficiente que a versão brasileira do algoritmo de Porter [17]. O *stemmer* RSLP (Removedor de Sufixos da Língua Portuguesa) tem sido empregado em diversas aplicações neste idioma [7, 9, 27].

Em [7], é apresentada uma ferramenta - assistente digital - de auxílio ao usuário no processo de busca de informações em portais corporativos. Este assistente interage com o usuário humano através de um navegador *Web*, recebendo perguntas

textuais e respondendo automaticamente (sem que haja intervenção humana) a essas perguntas com base em conhecimento previamente armazenado. Neste caso, a aplicação do algoritmo de *stemming* ocorre antes das perguntas digitadas pelos usuários serem armazenadas.

Em [9], é implementado um sistema automatizado de perguntas e respostas para ambientes educacionais. O sistema possui um repositório de perguntas e respostas, podendo responder de forma automática a consultas formuladas por seus usuários. Dessa forma, a cada consulta nova formulada, o sistema consulta o repositório, devolvendo ao usuário uma resposta que julga ser adequada àquela consulta.

Em [27], é realizada pesquisa envolvendo o desenvolvimento de uma solução que realiza clustering de documentos cujos conteúdos se apresentam no idioma português. O processo de clustering serve para realizar o agrupamento de documentos similares. Neste trabalho, é implementada uma ferramenta que realiza as três principais etapas do processo de Mineração de Texto:

- pré-processamento dos dados, fazendo os procedimentos necessários para tratamento de dados textuais em português;
- clusering de dados, onde são disponibilizados diferentes métodos para esta tarefa, e;
- visualização dos resultados, no intuito de fornecer auxílio para a interpretação e utilização dos resultados fornecidos por um processo de clustering.

Esta solução de clustering foi testada sem o uso de *stemming*, com o *stemmer* de Porter para a língua portuguesa e com o *stemmer* RSLP (Removedor de Sufixos da Língua Portuguesa).

A próxima subseção irá abordar o modo de funcionamento do *stemmer* RSLP (Removedor de Sufixos da Língua Portuguesa).

Mecanismo de funcionamento do *Stemmer* RSLP

O *stemmer* RSLP é composto por 8 passos, que são executados em uma ordem predefinida, mostrada na Figura 3.

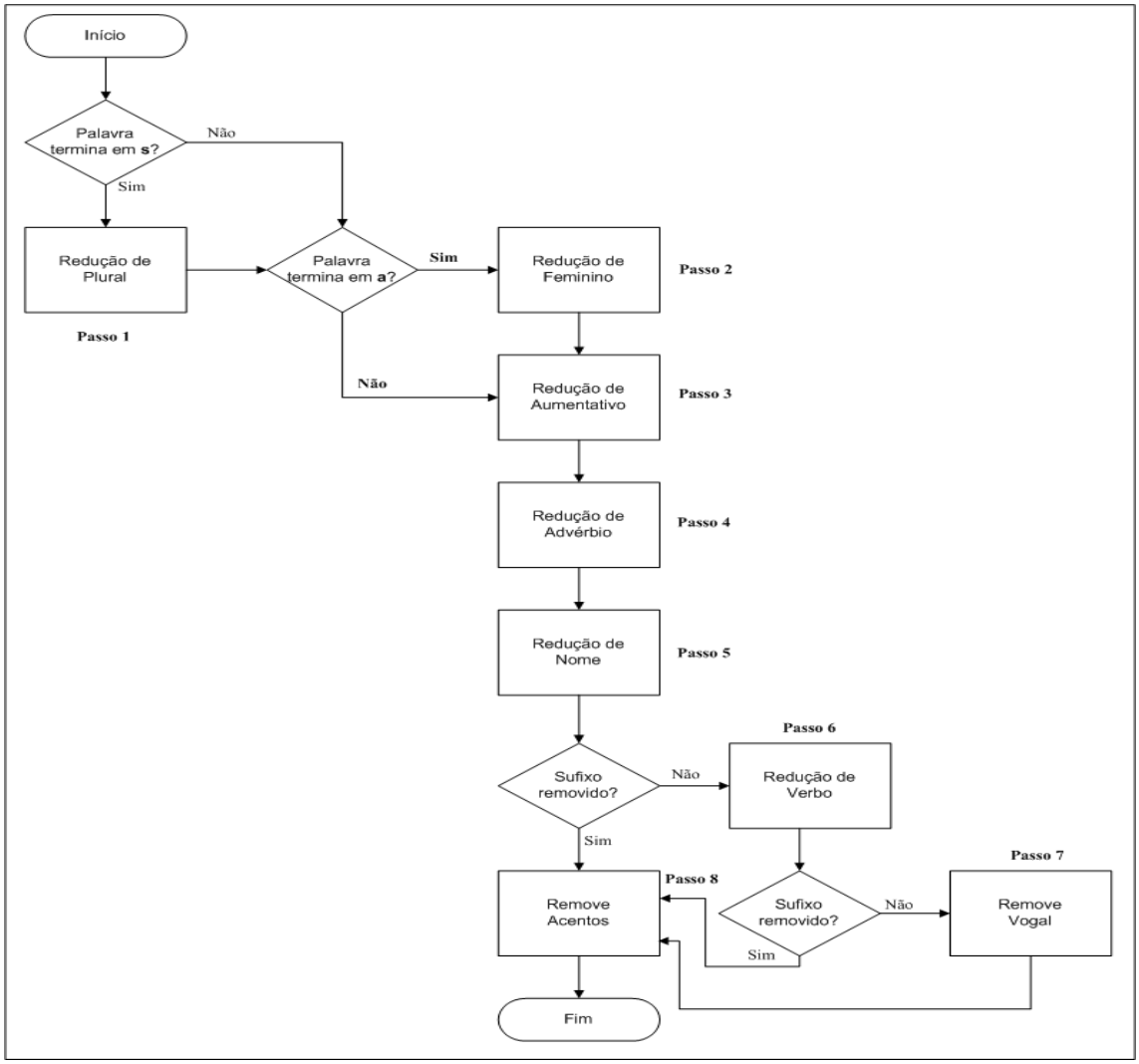

Figura 3: Fluxo de controle do algoritmo RSLP (adaptada de [17])

Cada passo do RSLP é formado por um conjunto de regras examinadas em seqüência em que apenas uma é aplicada a cada momento e cada regra possui quatro componentes:

- O sufixo a ser removido;
- O tamanho mínimo do *stem*;

- Um sufixo de substituição, se necessário, e;
- Uma lista de exceção.

Um exemplo de regra é {“ona”,3,“ão”,{“carona”}}, onde **ona** é o sufixo a ser removido, **3** é o tamanho mínimo do *stem*, **ão** representa o sufixo de substituição, e a palavra **carona** representa a lista de exceção. Por exemplo, a palavra **chefona**, depois de processada por esta regra, é tranformada em **chefão**.

O *stemmer* RSLP retira aos poucos as terminações das palavras, através da execução de diversos módulos, cada um possuindo uma lista de sufixos inerente ao tipo de redução. Os tipos de redução existentes são: plural para singular, feminino para masculino, aumentativo, advérbio, nome, verbo, e vogal.

Os passos executados pelo *stemmer* RSLP são resumidos:

- Passo 1 - redução de plural: este passo é executado caso a palavra termine em **s**. Em geral, na língua portuguesa, essa terminação representa plural. O sufixo dessas palavras é removido, caso elas não façam parte da lista de exceção.
- Passo 2 - redução de feminino: este passo é executado caso a palavra ou a cadeia de caracteres resultante do passo anterior termine em **a**. Aqui, as formas femininas são transformadas para o correspondente masculino, através da retirada da terminação designadora de feminino, e em alguns casos adicionando sufixo representativo da forma masculina.
- Passo 3 - redução de aumentativo e diminutivo: neste passo, são removidos os sufixos que representam aumentativo ou diminutivo.
- Passo 4 - redução adverbial: neste passo, caso a palavra termine em **mente**, este sufixo é retirado, havendo também exceções.
- Passo 5 - redução de substantivo e adjetivo: aqui, são tratados os sufixos de nomes e adjetivos. Caso o sufixo seja retirado aqui, os passos **6** e **7** não são executados.

- Passo 6 - redução verbal: são retirados os sufixos que representam terminações verbais.
- Passo 7 - remoção de vogal: são retiradas as vogais **a**, **e** e **o**.
- Passo 8 - remoção de acentos: são removidos os acentos das palavras.

A estratégia do RSLP é realizar um tratamento sufixal por subconjunto de palavras que possuem determinada particularidade (tipos de reduções realizadas em cada passo do algoritmo). A seqüência do algoritmo reduz aos poucos as terminações das palavras, de modo a obter o *stem*. Esse *stemmer* possui 199 regras.

2.6 Considerações Finais sobre Algoritmos de *Stemming*

Este capítulo apresentou uma revisão bibliográfica sobre processo de *stemming*, destacando-o como parte do processo de Mineração de Texto.

Diversas formas de avaliar algoritmos de *stemming* foram apresentadas, e descritas as principais abordagens utilizadas para sua construção. Notou-se que a maioria das pesquisas nesta área está voltada para a língua inglesa, no entanto havendo demanda por soluções para outros idiomas.

Em se tratando da língua portuguesa, foi encontrada na literatura apenas uma solução desenvolvida especialmente para esse idioma, denominada *stemmer* RSLP, mostrando-se com desempenho superior em relação à versão brasileira do algoritmo de Porter e servindo como *benchmark* para o estudo de caso do Capítulo 4.

Foi constatado que as soluções de *stemming* para a língua portuguesa não atendem suficientemente a demanda por esses tipos de algoritmos, servindo como motivação para o desenvolvimento de um modelo para o processo, que sirva para este idioma.

O próximo capítulo apresenta o STEMBR: um modelo para geração de *stems*

para a língua portuguesa. O STEMBR é construído baseado na abordagem removedores de afixos, também apresentada aqui.

Capítulo 3

O Modelo STEMBR

3.1 Preâmbulo

Neste capítulo é apresentado o STEMBR, um modelo para geração de *stems* composto por três etapas: tratador de exceção, tratador prefixal e tratador sufixal, sendo este último a etapa principal. Tal etapa é baseada num estudo estatístico realizado em um léxico. Neste caso, para este trabalho, foi escolhido o LexWeb (Léxico do Português da *Web*) [19], que implementa um léxico estatístico da língua portuguesa a partir de textos encontrados na *Web* brasileira. A seção 3.2 descreve o modelo STEMBR e a seção 3.3 apresenta o algoritmo principal do *stemmer*.

3.2 O Modelo STEMBR

O objetivo do modelo STEMBR é prover um mecanismo que dê suporte à criação de um *stemmer* que seja eficaz e que possa ser utilizado por aplicações que necessitem desse tipo de solução.

O modelo STEMBR é mostrado na Figura 4:

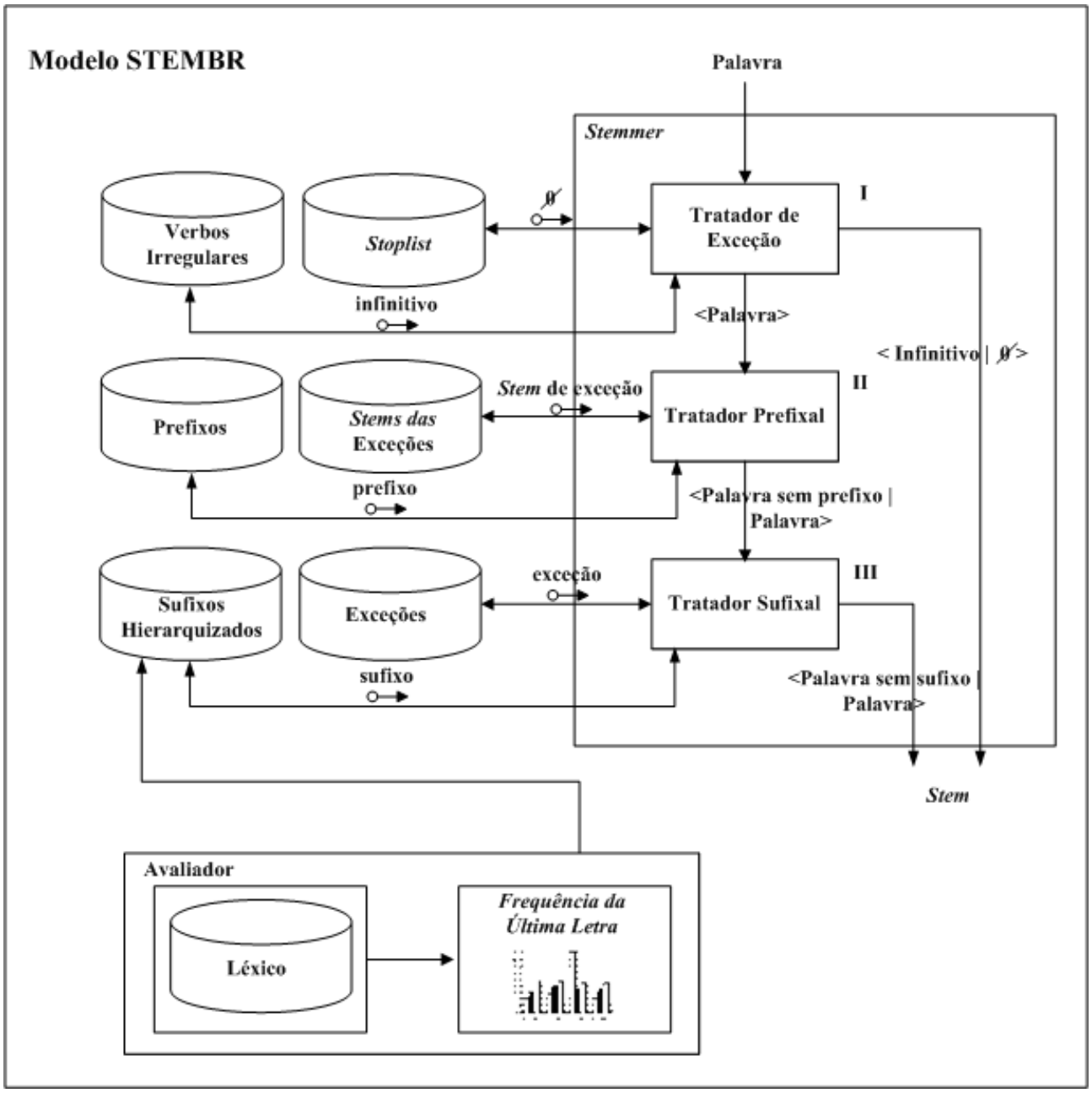

Figura 4: O Modelo STEMBR

No modelo STEMBR, a geração do *stem* ocorre em três etapas que seguem a seguinte ordem de execução:

1. Tratador de Exceção;
2. Tratador Prefixal, e;
3. Tratador Sufixal.

Na etapa Tratador de Exceção, ocorre o mapeamento de conjugações verbais irregulares para o infinitivo, além da identificação e eliminação das *stop words*.

Na etapa Tratador Prefixal, uma análise da subcadeia inicial da palavra é realizada, com o objetivo de identificar se a mesma é ou não prefixo, sendo este retirado, caso identificado.

Finalmente, na etapa Tratador Sufixal, é realizada remoção do sufixo da palavra. Esta é a etapa mais importante do processo, cujo mecanismo de funcionamento é baseado num estudo estatístico realizado em um léxico.

Stemmer

O componente ***Stemmer*** corresponde aos passos que devem ser executados para obter os *stems* das palavras. O *stemmer* concebido realiza esta tarefa por meio da execução seqüêncial de três etapas: tratador de exceção, tratador prefixal e tratador sufixal.

Tratador de Exceção

O componente Tratador de Exceção corresponde à primeira etapa que deve ser executada no processo de obtenção do *stem* das palavras. O objetivo aqui é verificar se a palavra pertence a algum tipo de caso particular, que é tratado como exceção. As exceções são classificadas em dois tipos:

- *stop words*, e;
- conjugações de verbos irregulares;

O componente Tratador de Exceção recebe a palavra, comparando-a com uma lista de *stop words* e/ou com uma lista de verbos irregulares. Ao final do processo, três situações podem ocorrer:

- a palavra, caso não seja exceção, é passada para o componente Tratador Prefixal;

- a palavra é identificada como *stop word*, neste caso não sendo processada, e;
- a palavra é identificada como conjugação verbal irregular, sendo mapeada para a forma infinitiva.

Tratador Prefixal

O componente tratador prefixal corresponde à segunda etapa que deve ser executada no processo de obtenção do *stem* das palavras. O objetivo desta etapa é analisar cada palavra, na tentativa de retirar o possível prefixo da mesma.

Diferentemente do RSLP (Removedor de Sufixos da Língua Portuguesa) - que não possui tratamento para prefixos - o modelo STEMBR possui um componente que objetiva identificar prefixos nas palavras, retirando-os.

Para executar esta tarefa, o tratador compara a subcadeia inicial de cada palavra com uma lista de prefixos, retirando-a, em caso positivo.

O tratamento prefixal não apresenta complexidade. A única particularidade ocorre quando a subcadeia não representa prefixo. Por exemplo, na palavra **impossibilidade**, a subcadeia **im** exerce função de prefixo, sendo perfeitamente retirável, sem perda da essência da palavra. Já na palavra **imagem** esta não exerce tal função, ao passo que sua remoção causa total perda do significado da palavra.

Para minimizar este problema, o tratador possui uma lista de exceções, onde são armazenados os *stems* de algumas palavras cuja subcadeia não representa prefixo. Prefixos que geram muitas exceções não são tratados.

A Tabela 16 mostra alguns prefixos, os *stems* das exceções, e algumas palavras cuja subcadeia inicial é mantida:

Prefixo	***Stems* das exceções**	**Exemplo**
ante	antecip	antecipar, antecipação
auto	autom, autor	automático, autoridade
ciclo	ciclon	ciclone, ciclones
contra	contrari, contrast	contrariar, contraste
inter	interpre, interess	interpretação, interpretar, interessado, interessante
super	superior	superioridade, supere
trans	transa, transp	transado, transpirar

Tabela 16: Prefixos, *stems* das exceções e exemplos

Tratador Sufixal

O componente tratador sufixal corresponde à última etapa que deve ser executada no processo de obtenção do *stem* das palavras. O objetivo desta etapa é analisar cada palavra, na tentativa de retirar o sufixo da mesma. Nesta etapa ocorre o tratamento dos verbos regulares. O tratamento sufixal é realizado adotando uma estratégia que consiste basicamente em:

- realizar o tratamento sufixal por subconjuntos de palavras que possuem a mesma letra final, e;
- retirar o maior sufixo possível da palavra.

O tratamento de sufixos é baseado em um conjunto de regras, cuja estrutura foi adaptada do RSLP, retirando o componente sufixo de substituição. Alguns sufixos também foram aproveitados do RSLP, e outros foram inseridos, totalizando 394 regras. Um exemplo de regra é: {“adíssima”,4,{“ ”}}, onde **adíssima** representa o sufixo a ser retirado, **4** o tamanho mínimo do *stem*, neste caso não havendo exceção cadastrada ({“ ”}).

A Tabela 17 ilustra algumas regras utilizadas pelo *stemmer*.

Sufixo	**Tamanho mínimo do *stem***	**Exemplo**	**Exceções**
entes	4	comoventes	acidentes, contentes
ados	4	torturados	consulados
ais	1	cantais	casais, mais
eis	2	comereis	acheis, alopreis
inho	3	cantinho	carinho, cominho
ado	2	cansado	consulado
este	4	comeste	faroeste, agreste

Tabela 17: Sufixos, tamanho mínimo do *stem*, exemplo e exceções

Para ilustrar, considere a palavra **irritadíssima** e **irrit** o *stem* convencionado. Considere também os sufixos **ima**, **íssima** e **adíssima**. Tais sufixos possuem a mesma letra final, e são candidatos a serem retirados da palavra. A melhor opção é a retirada do maior sufixo (**adíssima**).

Avaliador

O componente Avaliador representa o estudo realizado que provê a fundamentação teórica para o mecanismo de funcionamento do Tratador Sufixal. Este estudo corresponde a uma análise estatística realizada em um léxico, que resulta em uma lista de freqüência decrescente das letras no final das palavras. Cada elemento da lista forma um subconjunto de palavras, o que sugere que o tratamento sufixal seja realizado para cada um deles.

Neste ponto, surgem duas questões cruciais:

1. de que forma deve ser realizado o tratamento sufixal em cada subconjunto?
2. em qual ordem deve ser realizado o tratamento sufixal nos subconjuntos, de forma a obter melhores resultados?

Sobre o Tratamento Sufixal

No tratamento sufixal, a estratégia de retirar o maior sufixo da palavra é a que gera menos erros. Para isso, foi criada uma lista de sufixos para cada subconjunto, ordenada por tamanho, de forma decrescente. Assim, a remoção do sufixo ocorre percorrendo-se a lista de forma seqüêncial, verificando se o sufixo é subcadeia final da palavra. A Figura 5 ilustra esse mecanismo:

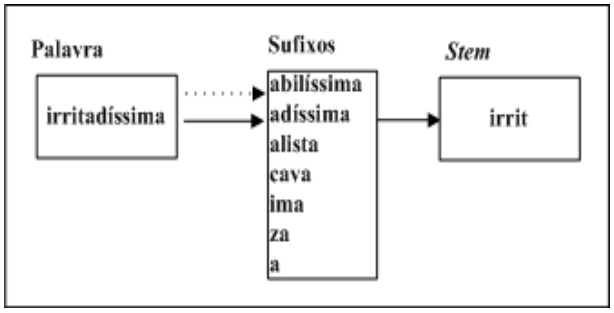

Figura 5: Retirada do maior sufixo da palavra

3.3 A Implementação

Para validação do modelo proposto, foi implementado um *stemmer*, na linguagem de programação C. O *stemmer* requer como parâmetro de entrada um arquivo texto contendo as palavras a serem processadas. Como saída, é gerado um arquivo texto, contendo os *stems* de cada palavra.

O LexWeb (Léxico do Português da Web)

Para a construção do *stemmer*, o léxico escolhido foi o LexWeb [19]. Este léxico foi motivado pela própria característica dinâmica da *Web*, como rica fonte de informações das mais diversas origens. Ele é composto por um conjunto de ferramentas que visitam a *Web* brasileira, coletando as palavras em uma base de dados,

e realizando um tratamento estatístico que consiste na contínua atualização de suas freqüências absolutas, com base nos textos baixados.

Como característica de projeto, o LexWeb armazena as palavras primeiramente em uma base de dados temporária, pois algumas palavras de baixa freqüência necessitam de uma análise individual para que possam realmente compor o léxico definitivo do projeto. Palavras com baixa freqüência em geral representam:

- palavras raras, geralmente encontradas apenas em dicionários especializados;
- palavras com ortografia errada;
- palavras que caíram em desuso, e;
- nomes de pessoas ou locais;

Freqüência da Última Letra

Como resultado do estudo estatístico realizado no LexWeb [19], foi obtida uma lista de freqüência decrescente das letras no final das palavras. Os nove primeiros elementos dessa lista, mostrados na Figura 6, representam 85% da amostra.

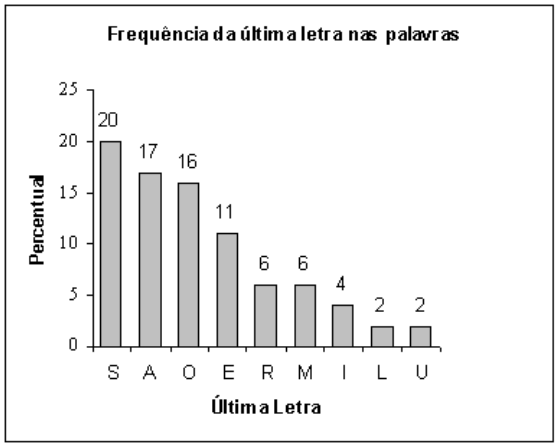

Figura 6: Freqüência das letras finais no corpus LexWeb

Ordem de execução do *stemmer*

A ordem de execução do *stemmer* foi obtida através da ajuda empírica de um especialista em Lexografia da Academia Brasileira de Letras, e depois de realizados exaustivos testes chegou-se a uma configuração que gerou os melhores resultados: "S", "R", "M", "L", "O", "A", "U", "E" e "I". Algumas justificativas são:

- Na língua portuguesa, salvo algumas exceções, palavras com sufixos terminados em **s** representam plural, ficando em geral com tamanho (número de letras) maior que na forma singular;
- Depois de retirados os sufixos que terminam em **r**,**m** ou **l**, existem casos onde as subcadeias geradas ainda não representam o *stem* correto da palavra. Alguns desses são resolvidos após a execução dos tratamentos para as vogais e semivogais. Alterando a ordem de execução do tratamento de sufixo entre esses subconjuntos, não foram notadas diferenças significativas no resultado do processamento;
- Alterando a ordem de execução entre os subconjuntos representantes das vogais **o** e **a**, não foram notadas diferenças significativas no resultado do processamento.

3.3.1 O Algoritmo

O pseudocódigo da Figura 7 apresenta o mecanismo de funcionamento do STEMBR. Nele, **Entrada** representa o arquivo texto contendo as palavras a serem processadas, **Stem** representa a seqüência de caracteres obtida após o processo de *stemming*, e **Saída** corresponde ao arquivo texto com os *stems* gerados.

```
1. Início STEMBR
2.   Para cada palavra do arquivo Entrada faça
3.    Se a palavra é stop word, então
4.     Stem é a própria palavra;
5.     Insere o Stem no arquivo Saída;
6.    Senão
7.     Se a palavra é verbo irregular, então
8.      Stem é a forma verbal no infinitivo;
9.      Insere o Stem no arquivo Saída;
10.    Senão
11.     Se a subcadeia inicial da palavra for prefixo, então
12.      Se o stem da palavra não estiver na lista de exceção, então
13.       Retira o prefixo da palavra;
14.      Fim Se
15.     Fim Se
16.     Se a subcadeia final da palavra for sufixo, então
17.      Retira o sufixo da palavra;
18.     Fim Se
19.     Insere o Stem no arquivo Saída;
20.    Fim Se
21.   Fim Se
22.  Fim Para
23. Fim STEMBR
```

Figura 7: Pseudocódigo principal do STEMBR

Foi visto que no modelo STEMBR a geração dos *stems* das palavras ocorre em três fases bem definidas. Tais fases fazem parte do componente *Stemmer* do modelo. O pseudocódigo da Figura 7 será explicado fazendo referência a cada uma delas.

Detalhamento do Algoritmo

No *stemmer*, cada palavra do arquivo de entrada pode passar por quatro testes principais:

1. onde é verificado se a palavra é *stop word* (linha 3);
2. onde é verificado se a palavra é verbo irregular (linha 7);
3. onde é verificado se a palavra possui prefixo (linha 11), e;
4. onde é verificado se a palavra possui sufixo (linha 16).

Os dois primeiros testes correspondem à implementação do tratador de exceções. No primeiro, caso a palavra seja uma *stop word*, o algoritmo convenciona que o *stem* é a própria palavra, neste caso inserindo o *stem* correspondente no arquivo **Saída**. Já no segundo, caso a palavra seja uma forma conjugada de verbo irregular, o algoritmo convenciona que o *stem* é a forma infinitiva desse verbo, inserindo o *stem* correspondente no arquivo **Saída**.

Há algoritmos de *stemming* que não realizam tratamento para *stop words*. Isto ocorre pois muitas vezes esta tarefa é realizada antes da execução deles. Optou-se por prever este tratamento no modelo, desta forma capturando casos onde *stop words* não são eliminadas antes do processo de *stemming*.

O terceiro teste corresponde à implementação do tratador prefixal. Neste caso, é examinado se a subcadeia inicial da palavra corresponde a algum prefixo cadastrado na lista de prefixos. Caso verdadeiro, ainda é necessário outro teste (linha 12), para verificar se a palavra representa exceção.

A situação de exceção no tratamento de prefixos ocorre porque há possibilidade de existirem palavras cuja subcadeia inicial corresponda graficamente a algum prefixo, quando na prática esta não exerce tal função. Por exemplo, a subcadeia **des** pode ser retirada da palavra **desmontar** sem que haja perda da essência da palavra. No entanto esta mesma subcadeia não deve ser retirada da palavra **destino** e até mesmo das suas diversas variações (destinado, destinava, destinou, etc.). Para isso,

o tratador perfixal gerencia uma lista de *stems* que correspondem às exceções. Neste exemplo, o *stem* armazenado corresponde à subcadeia **destin**.

O último teste corresponde à implementação do tratador sufixal. O teste verifica se a subcadeia final da palavra corresponde a algum sufixo. Caso positivo, este é retirado (linha 17), e sendo inserido o *stem* correspondente no arquivo **Saída** (linha 19).

3.4 Considerações Finais sobre o Modelo STEMBR

Este capítulo apresentou o STEMBR, um modelo para geração de *stems* desenvolvido especificamente para a língua portuguesa, utilizando a técnica removedores de afixos.

O *stemmer* realiza tratamentos prefixal e sufixal nas palavras, sendo este último o mais importante, cujo mencanismo de funcionamento é baseado em um estudo estatístico realizado no LexWeb [19].

Os sufixos são removidos por subconjunto de palavras que possuem a mesma letra final, sendo que a ordem de execução desse tratamento foi cuidadosamente testada, objetivando obter os melhores resultados. Além disso, o tratamento sufixal retira a maior subcadeia possível de cada palavra.

As principais contribuições que o modelo STEMBR provê são:

- tratamento prefixal, e;
- tratamento sufixal eficaz.

Através do modelo apresentado, cada fase de tratamento da palavra para a obtenção do *stem* fica delineada e com função bem definida. A estrutura do modelo facilita a implementação do *stemmer*, através do uso da técnica removedores de afixos.

O próximo capítulo apresenta um estudo de caso, objetivando avaliar o *stemmer*.

Capítulo 4

Estudo de caso

4.1 Preâmbulo

Três tipos de experimentos computacionais são realizados para comparar resultados e avaliar o desempenho do STEMBR, formulado, desenvolvido e implementado nesta dissertação: método manual, redução do vocabulário e o método de Paice. Esses métodos foram introduzidos e exemplificados na seção 2.2 deste trabalho. A próxima seção versa sobre as características das amostras utilizadas. A seção 4.3 apresenta detalhes sobre a implementação dos procedimentos para a realização dos testes e as seções restantes mostram os resultados do estudo de caso.

4.2 As Amostras Utilizadas

No estudo de caso, foram utilizadas duas amostras, de origens e tamanhos diferentes, discriminadas na Tabela 18:

Fonte	**Tamanho**
Dicionário Aurélio Eletrônico	1.000
LexWeb	5.000

Tabela 18: Origem e tamanho das amostras

A partir desse ponto, a amostra retirada do Dicionário Aurélio Eletrônico será referenciada por Amostra I e a obtida do LexWeb [19] será referenciada por Amostra II.

4.3 Metodologia Utilizada para Avaliação

Como visto no Capítulo 2, há duas tarefas manuais que devem ser realizadas para avaliação dos resultados: no método manual, um ser humano convenciona o *stem* correto para cada palavra. Já no método de Paice, as palavras devem ser organizadas em grupos semânticos.

Para essas tarefas, foi obtida a ajuda do professor Cláudio Sobrinho, especialista em Lexografia da Academia Brasileira de Letras.

O trabalho do professor foi realizado através do uso do computador, pelo preenchimento de uma planilha eletrônica para cada amostra. Nesta fase, o professor:

- convencionou um *stem* para cada palavra, e;
- classificou cada palavra em grupos semânticos.

Após a conclusão desta tarefa, optou-se pela implementação de um procedimento de exportação dos dados para um Sistema Gerenciador de Banco de Dados (SGBD), com o objetivo de realizar as consultas necessárias à obtenção das informações inerentes a cada método de avaliação.

Com o banco de dados criado e alimentado, foi implementado um procedimento de geração automática do arquivo texto, que é parâmetro de entrada para os *stem-*

mers. Além disso, também foram implementados procedimentos de exportação para o SGBD, dos arquivos textos contendo os *stems* gerados por cada *stemmer*.

Como o *stemmer* RSLP não possui tratamento especial para *stop words* e para verbos irregulares, essas palavras não foram incluídas nas amostras.

A partir desse ponto, os resultados de cada método de avaliação foram obtidos de forma automática, por meio de consultas em linguagem SQL, submetidas ao SGBD. As próximas seções mostram os resultados obtidos por meio da execução dos procedimentos ora descritos.

4.4 Resultados do Método Manual

4.4.1 Método Manual aplicado à Amostra I

A Tabela 19 mostra o resultado do método manual aplicado à Amostra I:

Stemmer	**Acerto**	***Overstemming***	***Understemming***
RSLP	55,30%	4,70%	37,70%
STEMBR	62,20%	8,90%	27,10%

Tabela 19: Resultado do método manual aplicado à Amostra I

A partir desta tabela, foi gerado o gráfico da Figura 8, que representa o comportamento dos *stemmers* em relação aos acertos, erros de *overstemming* e aos erros de *understemming*.

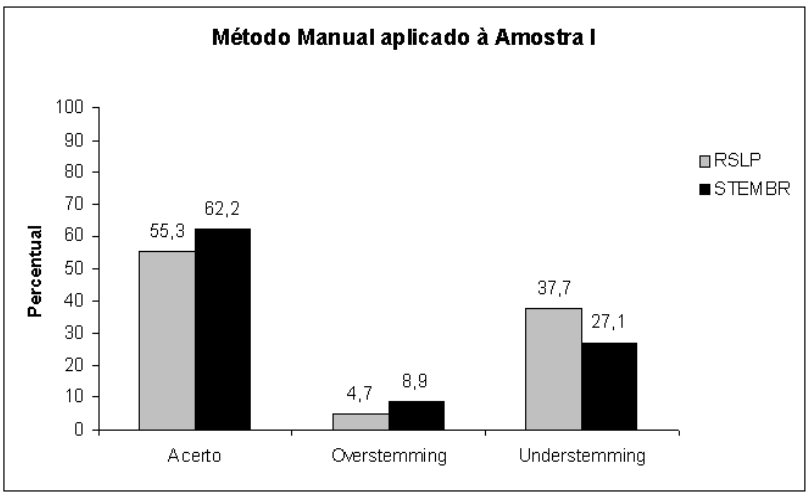

Figura 8: Percentual de Acertos dos *stemmers* na Amostra I

Analisando a Figura 8, percebe-se que o STEMBR acertou o *stem* de 622 palavras, e o RSLP, 553. O percentual de acertos do STEMBR foi 6,9% superior ao do RSLP. Os *stemmers* obtiveram acertos comuns em 485 palavras, sendo que o STEMBR acertou sozinho 137 *stems*, e o RSLP, 68.

Foi constatado que o STEMBR foi 4,2% pior que o RSLP, em relação aos erros de *overstemming*. Observou-se que muitas vezes os esforços realizados para minimizar erros de *overstemming* acabam contribuindo para o aumento dos erros de *understemming* e vice-versa. O RSLP teve taxa de erro 10,6% maior que a do STEMBR.

A próxima subseção apresenta os resultados desse mesmo teste, aplicado à amostra composta de 5000 palavras.

4.4.2 Método Manual aplicado à Amostra II

A Tabela 20 mostra o resultado do método manual aplicado à Amostra II:

Stemmer	Acerto	*Overstemming*	*Understemming*
RSLP	67,60%	8,96%	22,58%
STEMBR	69,20%	12,50%	17,96%

Tabela 20: Resultado do método manual aplicado à Amostra II

A partir desta tabela, foi gerado o gráfico da Figura 9, que representa o comportamento dos *stemmers* em relação aos acertos, erros de *overstemming* e aos erros de *understemming*.

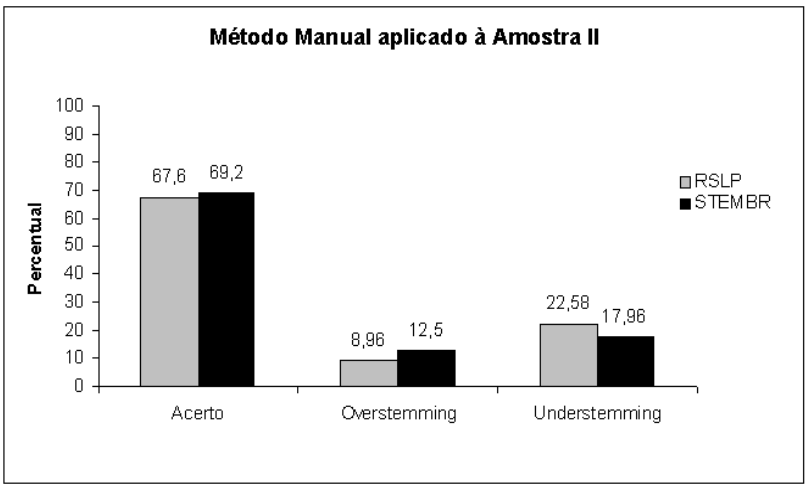

Figura 9: Percentual de Acertos dos *stemmers* na Amostra II

Analisando a Figura 9, observa-se que o STEMBR acertou o *stem* de 3451 palavras, e o RSLP, 3380. O percentual de acertos do STEMBR foi 1,6% superior ao do RSLP. Os *stemmers* obtiveram acertos comuns em 3.088 palavras, sendo que o STEMBR acertou sozinho 363 *stems*, e o RSLP, 292.

Foi constatado que o STEMBR foi 3,5% pior que o RSLP, em relação aos erros de *overstemming*. Já em relação aos erros de *understemming*, o RSLP teve taxa de erro 4,6% maior que a do STEMBR.

4.4.3 Validação Estatística

Testes estatísticos são úteis quando há interesse em avaliar suposições a respeito de algum processo. Para isso, há necessidade de que sejam formuladas duas proposições mutuamente excludentes [30]. Neste caso, uma proposição afirma uma dada característica do que é estudado; a outra, nega-a.

Por convenção, a característica imaginada como verdadeira em relação ao que vai ser avaliado é denominada hipótese nula (indicada por H_0). Já a sua negação (proposição tida como verdadeira caso a hipótese nula seja rejeitada) é chamada hipótese alternativa (indicada por H_1) [1].

A hipótese nula afirma a igualdade em que se presume a característica real como sendo igual ao valor alegado. Já a hipótese alternativa estabelece a desigualdade que um difere de forma significativa do outro [36]. O objetivo do teste é rejeitar H_0, dessa forma fornecendo indícios que a hipótese H_1 é aceitável.

Considere dois *stemmers* a serem comparados. Chamemos de P_1 e P_2 para duas proposições:

- P_1: proporção de acertos do *stemmer* RSLP, e;
- P_2: proporção de acertos do *stemmer* STEMBR.

Neste caso, tem-se:

- H_0: $P_1 = P_2$
- H_1: $P_1 \neq P_2$

Para validar estatisticamente as hipóteses H_0 ou H_1, será necessára a aplicação de um teste qui-quadrado (X^2). Este teste é adequado para comparações de proporções em problemas nos quais o mesmo conjunto de dados serve de base à comparação, neste caso sendo dependente apenas do número de acertos exclusivos do *stemmer* em cada amostra.

O teste é dado pela estatísica:

$$X^2 = \frac{(|x - y| - 1)^2}{x + y} \tag{8}$$

Para expressar a conclusão final de um teste de hipótese, pode-se quantificar a chance do que foi observado a respeito dos resultados mais extremos, sob a hipótese de igualdade dos grupos. Dessa forma, há necessidade de se calcular a probabilidade de ocorrência de valores iguais ou superiores ao assumido pela estatística do teste, admitindo a hipótese de que H_0 é verdadeira.

Este número é denominado *probabilidade de significância* ou *valor-p*. Já que o *valor-p* é calculado supondo-se que H_0 é verdadeira, é possível que sejam feitas duas inferências quando este for muito pequeno:

1. um evento extremamente raro pode ter ocorrido, ou;
2. a hipótese H_0 não deve ser verdadeira.

Na prática, quanto menor o *valor-p*, maiores são as evidências para rejeitar H_0. Neste trabalho, o *valor-p* será utilizado assumindo um risco de 5%.

Teste de hipótese para a Amostra I

A Tabela 21 representa o número de acertos exclusivos de cada *stemmer* para a Amostra I.

Stemmer	**Acertos exclusivos**
STEMBR	137
RSLP	68

Tabela 21: Número de acertos exclusivos dos *stemmers* na Amostra I

Calculando-se o valor de X^2 para a Amostra I, tem-se:

$X^2 = (|137 - 68| - 1)^2 / (137 + 68) = 22.56$

Para que seja obtido o *valor-p*, é necessário calcular a probabilidade de encontrar valores maiores que 22.56, ou seja, $Pr(\mathrm{X^2}_1 \geq 22.56)$. Após o cálculo dessa probablilidade, foi obtido o valor 2.0368_*10^{-6}.

Dado que $Pr(\mathrm{X^2}_1 \geq 22.56)$ é muito pequeno, pode-se inferir que a hipótese H_0 é falsa, dessa forma aceitando-se a hipótese H_1 como verdadeira.

Teste de hipótese para a Amostra II

A Tabela 22 representa o número de acertos exlusivos de cada *stemmer* para a Amostra II.

Stemmer	**Acertos exclusivos**
STEMBR	363
RSLP	292

Tabela 22: Número de acertos exclusivos dos *stemmers* na Amostra II

Calculando-se o valor de $\mathrm{X^2}$ para a Amostra II, tem-se:

$\mathrm{X^2}$= (|363 - 292| - 1)^2 / (363 + 292) = 7.48

Para que seja obtido o *valor-p*, é necessário calcular a probabilidade de encontrar valores maiores que 7.48, ou seja, $Pr(\mathrm{X^2}_1 \geq 7.48)$, sendo verdadeira a hipótese H_0. Após o cálculo dessa probablilidade, foi obtido o valor 6.239_*10^{-5}

Dado que $Pr(\mathrm{X^2}_1 \geq 7.48)$ é muito pequeno, pode-se inferir que a hipótese H_0 é falsa, dessa forma aceitando-se a hipótese H_1 como verdadeira.

Outro teste

Para verificar se o STEMBR é melhor que o RSLP, torna-se necessária a aplicação de outro teste de hipótese. Neste caso, tem-se:

- H_0: $P_1 >= P_2$
- H_1: $P_1 < P_2$

Aqui, o teste é dado pela estatística:

$$\frac{(x - y - 1)}{\sqrt{x + y}} \tag{9}$$

A Tabela 23 mostra os valores obtidos para as Amostras I e II.

Amostra	**Resultado da estatísica**
Amostra I	-4.88
Amostra II	-2.81

Tabela 23: Resultado do segundo teste de hipótese

Para que seja obtido o *valor-p*, é necessário calcular a probabilidade de encontrar, para a Amostra I, valores menores que -4.88 e, para a Amostra II, valores menores que -2.81. Ao realizar o cálculo desta probabilidade, foram obtidos os valores 5.07_*10^{-7} e 2.45_*10^{-3}, relativos às Amostras I e II, respectivamente.

Com base nos valores calculados para o *valor-p*, não há evidência de que H_0 seja verdadeira.

4.5 Resultado do Método Redução do Vocabulário

Para este teste, além das duas amostras, foram incluídas todas as 130.000 palavras do LexWeb [19]. A Tabela 24 mostra os resultados obtidos:

A partir da Tabela 24, foi gerado o gráfico da Figura 10, representando o comportamento dos *stemmers* em relação à redução do vocabulário.

Amostra	RSLP	STEMBR
AmostraI	32,70%	29,20%
AmostraII	53,90%	53,92%
LexWeb	46,62%	44,30%

Tabela 24: Redução do Vocabulário

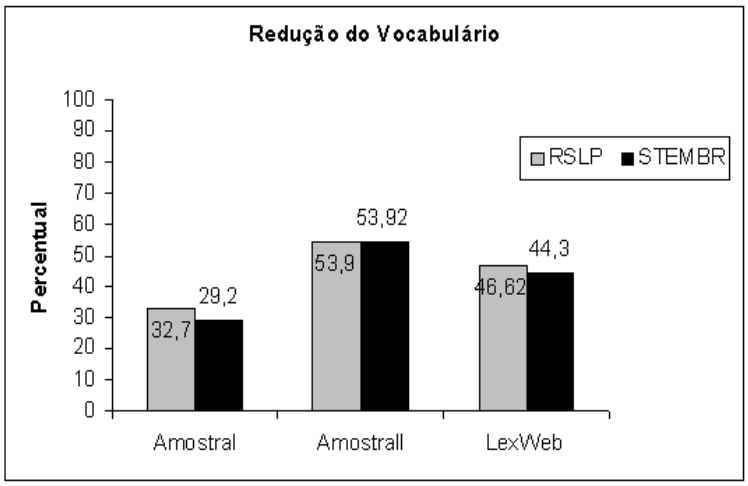

Figura 10: Redução do Vocabulário para a Amostra I

Verificou-se que o STEMBR gerou 292 *stems* excluindo-se as repetições, e o RSLP, 327. A taxa de redução do STEMBR foi 3,5% menor que a do RSLP considerando a Amostra I. Em relação à Amostra II, o STEMBR gerou 2.696 *stems* excluindo-se as repetições, e o RSLP, 2.695. Neste caso os *stemmers* ficaram praticamente empatados. Finalmente, em relação ao LexWeb, o STEMBR gerou 57.591 *stems* excluindo-se as repetições, e o RSLP, 60.615. A taxa de redução do STEMBR, neste caso, foi cerca de 2,3% menor que a do RSLP.

4.6 Resultado do Método de Paice

Como visto no Capítulo 2, o método de Paice requer que a amostra de palavras esteja dividida em grupos semânticos, e os erros de *overstemming* e *understemming* são contabilizados primeiramente para cada grupo, para posteriormente serem cal-

culados os índices globais OI e IU. Neste processo, todas as palavras das amostras foram cuidadosamente classificadas e revisadas, no intuito aumentar a acurácia do teste.

4.6.1 Resultado do Método de Paice Aplicado à Amostra I

A Tabela 25 apresenta o resultado do método de Paice aplicado à Amostra I.

	RSLP	**STEMBR**
OI	0,0000709	0,0000730
UI	0,492751	0,447116
SW	0,0001440	0,0001632

Tabela 25: Resultado do Método de Paice para a Amostra I

Neste caso, o STEMBR obteve índice de *overstemming* maior que o RSLP, confirmando os resultados obtidos através do método manual em relação aos erros de *overstemming*. Já em relação aos erros de *understemming*, o STEMBR obteve índice menor que o RSLP, também confirmando os resultados obtidos através do método manual em relação aos erros de *understemming*. O valor SW representa a **força** do *stemmer* (**forte** ou **fraco**). Neste teste foi constatado que SW(STEMBR)>SW(RSLP), demonstrando que o *stemmer* STEMBR é mais **forte** que o *stemmer* RSLP.

4.6.2 Resultado do Método de Paice Aplicado à Amostra II

A Tabela 26 mostra o resultado do método de Paice para a amostra Amostra II.

	RSLP	**STEMBR**
OI	0,0000098	0,00001010
UI	0,2959070	0,2885780
SW	0,000033	0,000035

Tabela 26: Resultado do Método de Paice para a Amostra II

O STEMBR obteve índice de *overstemming* maior que o RSLP, confirmando também os resultados obtidos através do método manual em relação aos erros de *overstemming*. Em relação ao índice de *understemming*, o STEMBR obteve valor menor que o RSLP, confirmando os resultados obtidos através do método manual em relação aos erros de *understemming*. Note que SW(STEMBR)>SW(RSLP), indicando que o *stemmer* STEMBR é mais **forte** que o *stemmer* RSLP.

Ponderações sobre o Métoto de Paice

No artigo proposto por Paice [32], os significados das medidas de índice de *overstemming*(OI) e de *understemming* (UI) estão bem definidos. Tais índices são computados para cada grupo semântico de palavras, baseado no princípio de que palavras pertencentes a um mesmo grupo devem possuir uma única representação (*stem*) e que esse *stem* não deve representar outro grupo semântico. Na prática esses índices são calculados sem uma preocupação individual com o *stem* de cada palavra sendo comparado a um *stem* convencionado, a exemplo do que ocorre com o método manual.

No entanto, o significado da medida SW não é claro. É possível inclusive que o parâmetro SW seja indeterminado, por exemplo, caso o *stemmer* não apresente erros de *understemming*. Segundo Paice (contatado por *email* em novembro de 2004), para um *stemmer* típico da língua inglesa, valores de OI ou UI sendo zero tendem a ocorrer apenas se o grupo de palavras for muito pequeno. O próprio autor concorda que a tarefa de realizar o grupamento semântico das palavras é uma atividade tediosa.

4.7 Conclusões Finais sobre os Testes

Em uma análise geral dos resultados dos testes utilizando o método manual, pode-se concluir que, para as duas amostras:

- o STEMBR obteve média de acerto de 67.8%, e o RSLP, 65.5%;

- o STEMBR obteve média de erro de *overstemming* igual a 5.9%, e o RSLP, 4.1%;
- o STEMBR obteve média de erro de *understemming* igual a 9.7%, e o RSLP, 12.5%;

Os resultados dos testes de hipótese realizados nas amostras evidenciam que os *stemmers* possuem capacidades diferentes quando do acerto dos *stems*. Neste caso, foi verificado que o STEMBR possui maior capacidade de acerto que o RSLP, em ambas as amostras.

Em relação ao teste de redução do vocabulário, observou-se que:

- o STEMBR foi melhor 2.3% que o RSLP, quando da utilização das 130.000 palavras do LexWeb [19].

No método de Paice, pode-se concluir que, para as duas amostras:

- UI(STEMBR)<UI(RSLP);
- OI(STEMBR)>OI(RSLP);
- SW(STEMBR)>SW(RSLP);

Os testes realizados tinham o objetivo de avaliar empíricamente o *stemmer* implementado a partir do modelo STEMBR. Para isso, escolheu-se como parâmetro de comparação o *stemmer* RSLP, que se mostrou superior à versão brasileira do algoritmo de Porter.

4.8 Considerações sobre o Estudo de Caso no Contexto da Dissertação

Este capítulo apresentou um estudo de caso envolvendo o STEMBR, implementado a partir do modelo apresentado no Capítulo 3, e o *stemmer* RSLP. Para a

avaliação dos *stemmers* foram utilizados três métodos: método manual, redução do vocabulário, e o método de Paice.

Os testes foram realizados em duas amostras de origens e tamanhos diferentes, cuidadosamente preparadas no intuito de aumentar a acurácia deles, tendo em vista as tarefas manuais de grupamento semântico e o procedimento de *stemming* manual.

Os bons resultados da análise de acurácia obtida pelo método manual permitem concluir que o STEMBR pode ser incluído em estudos para escolha do melhor *stemmer* para um determinado problema de mineração de texto. Os resultados da análise de erro do método de Paice e do método manual mostraram-se compatíveis quanto aos valores obtidos de cada amostra.

Capítulo 5

Conclusão

5.1 Contribuições

A primeira contribuição desta dissertação é a apresentação de um modelo (STEMBR) sistemático para o processo de *stemming* na língua portuguesa.

No modelo STEMBR, cada palavra é analisada com o objetivo de serem retirados o prefixo e o sufixo, quando necessário. Além disso, verbos com conjugações irregulares são mapeados para a forma infinitiva, e palavras consideradas não relevantes para o conteúdo geral dos textos (*stop words*) são identificadas.

Outra contribuição é o *stemmer*, concebido e avaliado neste trabalho. A avaliação foi realizada utilizando-se três métodos: método manual, redução do vocabulário e o método de Paice. Analisando-se os resultados dos testes, verifica-se que o mesmo pode ser utilizado em aplicações que necessitem do uso de *stemmers* desenvolvidos especificamente para a língua portuguesa.

O sucesso da experiência de implementação do STEMBR serve de incentivo para a implementação de outros *stemmers*. De acordo com o que foi apresentado, foi dada uma contribuição substancial ao desenvolvimento de ferramentas de PLN, especial-

mente em relação ao desenvolvimento e implementação de *stemmers* específicos para a língua portuguesa.

5.2 Trabalhos Correlatos

O processo de *stemming* tem sido bastante explorado em aplicações que utilizam a língua inglesa [18, 33].

O trabalho de Porter [33] tornou-se o algoritmo clássico para extração de sufixos. No entanto, observa-se que *stemmers* que não utilizam uma abordagem estatística em sua construção são dependentes da morfologia do idioma o qual são projetados.

Aos poucos, começou a aumentar o interesse da comunidade científica em utilizar o processo de *stemming* em outros idiomas. As primeiras tentativas estão relacionadas com a adaptação do algoritmo de Porter para outras línguas [22, 24]. No entanto, esta tarefa tende a ser questionada, tendo em vista as características diversificadas e particulares que cada idioma apresenta.

No cuidadoso trabalho realizado em [20], por exemplo, são mostrados pontos de melhoria para o algoritmo de Porter adaptado para a língua portuguesa.

Uma boa alternativa para melhoria do processo de *stemming* está no desenvolvimento de *stemmers* específicos para o idioma.

Em se tratando da língua portuguesa, foi encontrado em [17] um *stemmer* especificamente projetado para esse idioma, denominado *stemmer* RSLP (Removedor de Sufixos da Língua Portuguesa), neste caso mostrando-se consideravelmente melhor que a adaptação do algoritmo de Porter. Esse *stemmer* foi escolhido como *benchmark* para a avaliação do *stemmer* STEMBR, desenvolvido nesta dissertação.

Após o desenvolvimento do *stemmer* STEMBR, foram realizados três experimentos computacionais, utilizando duas amostras de tamanhos e origens diferentes, para avaliação. Os experimentos envolveram o método manual, a taxa de redução do vocabulário e o método de Paice.

Os resultados computacionais demonstraram que o STEMBR obteve em média uma taxa de acerto 2,6% maior que o RSLP, além de uma redução do vocabulário cerca de 2% melhor. Também foi constatado, tanto no método manual quanto no método de Paice, que o STEMBR gera maior quantidade de erros de *overstemming*, errando menos em termos de *understemming*, quando comparado ao RSLP.

5.3 Limitações

Uma limitação ocorre em função do tamanho da amostra. É interessante a realização dos testes em um conjunto de palavras maior, e com origem diversificada. Além disso, é desejável a avaliação do *stemmer* desenvolvido através do uso desse algoritmo como parte da etapa de pré-processamento de dados, sendo observada a qualidade das tarefas de Mineração de Texto.

5.4 Trabalhos Futuros

Em termos de implementação, como trabalho futuro pode-se citar identificação das *stopwords* e o mapeamento das conjugações verbais irregulares para a forma infinitiva.

Um fato interessante surge ao se analisar a quantidade de erros (*overstemming* e *understemming*) gerada pelos *stemmers*. A análise quantitativa pode ocultar detalhes importantes sobre a qualidade dos *stemmers*. Como erros no processo de *stemming* são inevitáveis, o objetivo é que se chegue o mais próximo (número de caracteres) da situação ideal, que é o *stem* correto.

Para determinada palavra, em relação aos erros de *overstemming*, se um *stemmer* remove mais caracteres que outro, o que removeu menos tende a recuperar erroneamente uma quantidade menor de palavras semanticamente diferentes. Já em relação aos erros de *understemming*, se um *stemmer* deixa de remover mais letras que outro, este tende a deixar de recuperar uma quantidade maior de palavras semanticamente

semelhantes.

Diante disso, uma idéia para trabalho futuro é avaliar a qualidade desses erros, analisando o quão distante ou próximo cada *stem* gerado ficou do *stem* correto.

Outra idéia é a realização dos experimentos com um número maior de palavras, além de aplicar os algoritmos a um sistema de RI e avaliar tal sistema com a utilização de cada *stemmer*.

Como o processo de *stemming* é considerado uma fase do pré-processamento de texto, a qualidade de tarefas como classificação e clusterização de textos é dependente desses *stemmers*. Avaliar os resultados dessas aplicações utilizando cada *stemmer* é uma tarefa desejável e promissora.

Referências Bibliográficas

[1] Agresti, A. *Categorical Data Analysis.* Wiley, 2002.

[2] Aljlayl, M., e Frieder, O. On arabic search: Improving the retrieval effectiveness via a light stemming approach. Em *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management* (2002), ACM Press, pp. 340–347.

[3] Allan, J., e Kumaran, G. Stemming in the language modeling framework. Em *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval* (2003), ACM Press, pp. 455–456.

[4] Bacchin, M., Ferro, N., e Melucci, M. University of padua at clef 2002: Experiments to evaluate a statistical stemming algorithm. Em *Working Notes for CLEF 2002: Cross-Language Evaluation Forum Workshop* (2002), pp. 161–168.

[5] Carlberger, J., Dalianis, H., Hassel, M., e Knutsson, O. Improving precision in information retrieval for swedish using stemming. Relatório técnico, NADA-KTH, 2001.

[6] Chen, H., e Ho, T. K. Evaluation of decision forests on text categorization. Em *Proceedings of the SPIE Conference on Document Recognition and Retrieval VII* (San Jose, CA, 2000), vol. 3967, pp. 191–199.

[7] da Silva, L. K., Brahm, D. R., Tagliassuchi, G., e Loh, S. Um assistente digital para responder automaticamente perguntas de usuários humanos em

portais corporativos. Em *6º Simpósio Internacional de Gestão do Conhecimento - ISKM2003* (Curitiba, 2003), Editora Universitária Champagnat.

[8] de Lourdes da Silveira, M. *Recuperação Vertical de Informação: Um Estudo de Caso na Área Jurídica.* PhD thesis, UFMG, 2003.

[9] de Souza Cardoso, A. L. M., e de Menezes, C. S. Processamento eficiente de consultas em linguagem natural em ambientes educacionais. *1º Workshop em Tecnologia da Informação e da Linguagem Humana - TIL'2003* (2003).

[10] Ebecken, N. F. F., Lopes, M. C. S., e de Aragão Costa, M. C. Mineração de textos. Em *Sistemas Inteligentes: Fundamentos e Aplicações*, S. O. Rezende, Ed. Editora Manole, 2003, ch. 13, pp. 337–370.

[11] Figuerola, C. G., Gomez, R., Rodriguez, A. F. Z., e Berrocal, J. L. A. Stemming in spanish: A first approach to its impact on information retrieval. Em *CLEF 2001: Revised Papers from the Workshop of Cross-Language Evaluation Forum on Cross-Language Information Retrieval and Evaluation* (2001), pp. 301–310.

[12] Fuller, M., e Zobel, J. Conflation-based comparison of stemming algorithms. Em *Proc. of the Third Australian Document Computing Symposium, Sydney, Australia* (1998), pp. 8–13.

[13] Gaustad, T., e Bouma, G. Accurate stemming of Dutch for text classification. Em *Computational Linguistics in the Netherlands 2001* (Amsterdam, 2001), A. Nijholt e H. Hondorp, Eds., Rodopi, pp. 104–117.

[14] Gonzales, M., e Lima, V. L. S. Recuperação de informação e processamento da linguagem natural. *XXIII Congresso da Sociedade Brasileira de Computação III* (2003), 347–395.

[15] Gonzales, M., Martins, A., Barão, F., Leite, M., e de Lima, V. L. S. Construção de hierarquia de temas e subtemas de texto. *1º Workshop em Tecnologia da Informação e da Linguagem Humana - TIL'2003* (2003).

[16] Harman, D. How effective is suffixing? *Journal of the American Society for Information Science - JASIS 42*, 1 (1991), 7–15.

[17] Huyck, V. M. O. C. A stemming algorithm for the portuguese language. Em *SPIRE '01: Proceedings of Eigth Symposium on String Processing and Information Retrieval* (2001), pp. 186–193.

[18] Jones, K. S., e Willett, P. *Readings in information retrieval.* Morgan Kaufmann Publishers Inc., 1997.

[19] Junior, A. M. Lexweb: um léxico da língua portuguesa extraído automaticamente da internet. Dissertação de Mestrado, UFPA, 2004.

[20] Junior, J. C., Imamura, C. Y.-M., e Rezende, S. O. Avaliação de um algoritmo de *Stemming* para a língua portuguesa. *II Congresso de Lógica Aplicada à Tecnologia - LAPTEC 2001* (2001), 267–274.

[21] Kantrowitz, M., Mohit, B., e Mittal, V. Stemming and its effects on tfidf ranking (poster session). Em *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval* (2000), ACM Press, pp. 357–359.

[22] Kazem Taghva, R. B., e Sadeh, M. A stemming algorithm for the farsi language. Relatório Técnico 2003-02, Information Science Research Institute, University of Nevada, Las Vegas, 2003.

[23] Korenius, T., Laurikkala, J., J;rvelin, K., e Juhola, M. Stemming and lemmatization in the clustering of finnish text documents. Em *CIKM '04: Proceedings of the Thirteenth ACM conference on Information and knowledge management* (2004), ACM Press, pp. 625–633.

[24] Kraaij, W., e Pohlmann, R. Evaluation of a Dutch stemming algorithm. Em *OTS yearbook 1994*, J. Don, B. Schouten, e W. Zonneveld, Eds. Research Institute for Language and Speech (OTS), 1995, pp. 63–84.

[25] Kraaij, W., e Pohlmann, R. Viewing stemming as recall enhancement. Em *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR confer-*

ence on Research and development in information retrieval (1996), ACM Press, pp. 40–48.

[26] Larkey, L. S., Ballesteros, L., e Connell, M. E. Improving stemming for arabic information retrieval: light stemming and co-occurrence analysis. Em *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval* (2002), ACM Press, pp. 275–282.

[27] Lopes, M. C. S. *Mineração de Dados Textuais Utilizando Técnicas de Clustering Para o Idioma Português*. PhD thesis, UFRJ, 2004.

[28] Lovins, J. B. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics* (1968), 11–31.

[29] Mayfield, J., e McNamee, P. Single n-gram stemming. Em *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval* (2003), ACM Press, pp. 415–416.

[30] Milone, G. *Estatística Geral e Aplicada*. Editora Thomson, 2004.

[31] Nakov, P. Building an inflectional stemmer for bulgarian. Em *CompSysTech '03: Proceedings of the 4th international conference conference on Computer systems and technologies* (2003), ACM Press, pp. 419–424.

[32] Paice, C. D. An evaluation method for stemming algorithms. Em *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval* (1994), Springer-Verlag New York, Inc., pp. 42–50.

[33] Porter, M. F. *An algorithm for suffix stripping*. Morgan Kaufmann Publishers Inc., 1997.

[34] Riloff, E. Little words can make a big difference for text classification. Em *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval* (1995), ACM Press, pp. 130–136.

[35] Singh, B. S. Search algorithms. *Workshop on Digital Libraries: Theory and Practice* (2003).

[36] Soares, J. F., e Siqueira, A. L. *Introdução à Estatística Médica.* COOPMED, 1992.

[37] Stefik, M. *Introduction to knowledge systems.* Morgan Kaufmann Publishers Inc., 1995.