Um Estudo e Apreciação sobre Algoritmos de Stemming para a Língua Portuguesa

Marcirio Silveira Chaves¹

Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Informática
Programa de Pós-Graduação em Ciência da Computação – PPGCC
Av. Ipiranga, 6681 - Partenon - Porto Alegre – RS
CEP 90619-900
mchaves@inf.pucrs.br

Resumo. Para auxiliar o gerenciamento do grande volume de informações que tem sido gerado nos últimos 50 anos, foram desenvolvidos sistemas de recuperação automática de informação. Desde então, os métodos de recuperação de informação (RI) começaram a ganhar importância, dentre eles a técnica de conflação e os algoritmos de *stemming*. Os algoritmos de *stemming* são tradicionalmente utilizados em RI com o objetivo de melhorar a abrangência dos resultados da busca. Neste artigo é apresentado um estudo e apreciação, por meio de um experimento, de dois algoritmos de *stemming* elaborados especificamente para o tratamento da língua portuguesa.

1 Introdução

Os sistemas de recuperação automática de informação foram originalmente desenvolvidos para ajudar a gerenciar o grande volume de informações que tem sido gerado desde os anos 50. Para isso, existem diversos métodos de Recuperação de Informação (RI), conforme apresentados em [5]. Infelizmente, as palavras que aparecem nos documentos e consultas frequentemente têm muitas variantes morfológicas, como, por exemplo, os pares de termos “caixa” e “caixote” que não serão reconhecidos como equivalentes sem alguma forma de Processamento da Língua Natural (PLN).

Neste contexto, o objetivo deste artigo é realizar um estudo e apreciação, por meio de um experimento, sobre dois algoritmos de *stemming* desenvolvidos especificamente para a língua portuguesa.

O restante deste artigo está organizado como segue: A seção 2 apresenta os conceitos de conflação, *stemming* e forma canônica. A seção 3 introduz os dois algoritmos de *stemming* e apresenta o experimento realizado para verificar a precisão desses algoritmos. Além disso, os resultados são incluídos e a apreciação é descrita. Na seção 4 são apresentadas as considerações finais e idéias para trabalho futuro.

¹ Financiado pelo projeto de pesquisa HP-CPAD (Centro de Processamento de Alto Desempenho), convênio PUC-RS/HP.

2 Conflação

Conflação é o ato de fusão ou combinação para o processo de igualar variantes morfológicas de termos [5]. A conflação pode ser manual, usando algum tipo de expressão regular, ou automática, via programas chamados *stemmers*. Para reduzir as variações de uma palavra para uma forma única utilizam-se algoritmos de conflação. Segundo [4], existem dois métodos principais de conflação para se obter tal redução: *stemming* e redução à forma canônica.

2.1 Stemming

Freqüentemente, o usuário especifica uma palavra em uma consulta, mas somente uma variante desta palavra é apresentada em um documento relevante. Plurais, formas de gerúndio e sufixos de tempo passado são exemplos de variações sintáticas que impedem uma perfeita combinação entre uma palavra da consulta e uma palavra do documento respectivo. Este problema pode ser parcialmente solucionado com a substituição de palavras pelos respectivos *stems* delas [7].

Stem é o conjunto de caracteres resultante de um procedimento de *stemming* [6]. Ele não necessariamente é igual à raiz lingüística, mas servirá como uma denotação mínima não ambígua do termo. De acordo com [4], *stemming* consiste em reduzir todas as palavras ao mesmo *stem*, por meio da retirada dos afixos² da palavra, permanecendo apenas a raiz dela. O propósito, segundo [1], é chegar a um *stem* que captura uma palavra com generalidade suficiente para permitir um sucesso na combinação de caracteres, mas sem perder muito detalhe e precisão. Um exemplo típico de um *stem* é “conect” que é o *stem* de “conectar”, “conectado” e “conectando”.

2.1.1 Erros de Stemming

Dois erros típicos que costumam ocorrer durante o processo de *stemming* são *overstemming* e *understemming*. *Overstemming* se dá quando a cadeia de caracteres removida não é um sufixo, mas parte do *stem*. Por exemplo, a palavra gramática, após ser processada por um *stemmer*, é transformada no *stem* grama. Neste caso, a cadeia de caracteres removida eliminou parte do *stem* correto, a saber “gramát”. Já *understemming* ocorre quando um sufixo não é removido completamente. Por exemplo, quando a palavra “referência” é transformada no *stem* “referênc”, ao invés do *stem* considerado correto “refer”.

2.2 Forma Canônica

A noção de que sentenças podem assumir múltiplos significados conduz ao fenômeno das entradas distintas que devem assumir a mesma representação de significado. Por exemplo, as sentenças a seguir devem ter a mesma representação:

1. McDonalds tem comida vegetariana?
2. Eles têm refeição vegetariana no McDonalds?

² Sufixos e prefixos.

3. São vegetarianos os pratos servidos no McDonalds?
4. McDonalds serve pratos vegetarianos?

Dado que as alternativas utilizam diferentes palavras e têm uma ampla variedade de análises sintáticas, seria razoável esperar que elas tivessem substancialmente representações diferentes de significado.

Segundo [2], redução à forma canônica consiste em reduzir os verbos para o infinitivo, e os substantivos e adjetivos para a forma masculina singular. Essa redução faz com que entradas que significam a mesma coisa tenham a mesma representação de significado.

Tanto os algoritmos de *stemming* quanto a técnica de redução à forma canônica podem auxiliar a tarefa de RI, uma vez que permitem ao sistema de busca encontrar um número maior de documentos que podem ser relevantes para o usuário.

Para auxiliar a tarefa de RI, freqüentemente são utilizados *stems*, pois eles reduzem as variantes de uma mesma raiz da palavra para um conceito comum [7]. Além disso, o processo de *stemming* tem o efeito secundário de reduzir o tamanho da estrutura de índice porque o número de termos distintos indexados é reduzido.

3 Experimento com Algoritmos de Stemming para a Língua Portuguesa

Nesta seção é feita uma análise de precisão de dois *stemmers* desenvolvidos para o tratamento da língua portuguesa, a saber:

- Pegastemming³ (que será tratado, deste ponto em diante, como S_1), é de autoria de Marco Antonio Insaurriaga Gonzalez.
- PortugueseStemmer (que será tratado, deste ponto em diante, como S_2), foi desenvolvido em Viviane Orengo e Christian Huyck [8].

Cabe destacar que os algoritmos foram desenvolvidos para aplicações específicas conforme a necessidade de cada autor.

3.1 Descrição do Experimento

No início desta seção é importante esclarecer o critério utilizado para estabelecer o *stem* correto de uma palavra. De acordo com Bechara [3], chama-se raiz ao radical primário⁴ a que se chega dentro da língua portuguesa e comum a todas as palavras de uma mesma família. Bechara ainda apresenta exemplos de radical secundário e terciário.

Neste trabalho não foi considerado como *stem* correto somente o radical primário da palavra resultante do processo de *stemming*. Pois, conforme descrito na seção 2.1, o *stem* não é necessariamente igual à raiz da palavra. Por exemplo, a palavra “regularizar” possui como radical primário a cadeia de caracteres “reg”. Entretanto,

³ Disponível em <http://www.inf.pucrs.br/~gonzalez/ri/pesqdiss/analise.htm>.

⁴ Também chamado radical irreduzível. Por exemplo, o radical “bel” extraído da palavra “beleza”.

neste caso consideramos como *stem* correto o radical secundário “regular”, pois esse radical representa uma denotação não ambígua da palavra “regularizar”.

No experimento realizado foram incluídas as seguintes classes gramaticais: substantivo, verbo, adjetivo, advérbio, contração de preposição e pronome.

Segundo [5], *stemmers* podem ser julgados quanto à sua eficiência de recuperação – geralmente medida com abrangência e precisão, velocidade e tamanho, entre outros parâmetros. Neste estudo, apenas o aspecto de precisão foi levado em consideração, pois para ser feita uma apreciação completa e profunda de um *stemmer* é necessário, pelo menos, um corpus mais amplo. Na análise de precisão dos algoritmos foram utilizadas 500 palavras, sem repetição, retiradas de textos⁵ escolhidos aleatoriamente.

No caso de as palavras acentuadas, quando processadas corretamente pelos algoritmos, optou-se por considerar correto tanto o *stem* com acento quanto o *stem* sem acento. O algoritmo S₁ não faz tratamento para artigos, conjunções e preposições. Essas categorias gramaticais, geralmente são caracterizadas em PLN como *stop words* e não contribuem, em um primeiro momento, para um processo de indexação ou de RI. Logo, as palavras pertencentes a estas categorias não foram incluídas na análise.

3.2 Análise de Precisão de Dois Algoritmos de Stemming

O processo de análise começou com o procedimento de *stemming* manual, que permitiu verificar o *stem* correto de cada palavra. Com isso obteve-se um parâmetro para medir a precisão dos *stemmers* quando esses processam automaticamente um texto. Em seguida, foi realizada a execução dos *stemmers* utilizando o conjunto de palavras selecionadas. Para facilitar a análise dos resultados originados da execução dos *stemmers*, foi elaborada a Tabela 1, que apresenta os resultados do experimento. Finalmente, foi realizada a análise comparativa dos algoritmos com o objetivo de aferir a precisão dos mesmos.

Tabela 1. Resultados do Experimento

Categoria Gramatical	Quantidade	Acertos		Over stemming		Under stemming		Concordância	
		S ₁	S ₂	S ₁	S ₂	S ₁	S ₂	Acertos	Erros
Substantivo	263	185	209	22	44	31	8	163	30
Verbo	134	85	104	7	13	31	6	78	13
Adjetivo	59	34	46	14	9	12	4	31	10
Pronome	15	4	11	12	3	0	0	4	5
Contração de Preposição	14	5	8	6	6	0	0	5	6
Advérbio	15	6	12	8	3	1	0	5	2
Total (Abs.)	500	318	389	69	77	75	18	285	66
Total (%)	100	63,8	78	13,8	15,6	15	3,6	57,2	13,2

A Tabela 1 apresenta na primeira coluna as categorias gramaticais analisadas neste estudo. A segunda coluna mostra a quantidade de palavras pertencentes a cada

⁵ Os textos são resumos de dissertações de mestrado em Ciência da Computação do PPGCC.

categoria gramatical. A coluna 3 (Acertos) apresenta a quantidade de palavras processadas corretamente por cada algoritmo. Nessa coluna são somados os acertos comuns a ambos os algoritmos (valores apresentados na coluna “Concordância - Acertos”) com o acerto de cada algoritmo. Por exemplo, na categoria substantivo, o algoritmo S_1 processou corretamente as mesmas 163 palavras do algoritmo S_2 mais 22 palavras diferentes do algoritmo S_2 . A soma 185, é apresentada na coluna 3. As próximas duas colunas apresentam a quantidade de erros atribuídos a *overstemming* e a *understemming*, respectivamente. Por fim, a última coluna (Concordância) mostra a quantidade de acertos e erros comuns a ambos os algoritmos, ou seja, a quantidade de palavras que ambos processaram correta ou equivocadamente para cada categoria gramatical. Na análise dos resultados apresentados na Tabela 1 podem-se inferir as seguintes informações:

- A quantidade total de erros atribuídos à *overstemming* foi bastante próxima para os dois algoritmos, 69 palavras processadas por S_1 e 77 palavras processadas por S_2 ;
- O algoritmo S_2 apresentou somente 18 erros atribuídos à *understemming*;
- Ambos os algoritmos processaram 57,2% das palavras corretamente, ao passo que 13,2% foram processadas equivocadamente. Esses resultados indicam que houve uma concordância entre os algoritmos de 70,4% das palavras utilizadas no experimento, ou seja, as mesmas palavras foram processadas ou de forma correta ou equivocada;
- Para a categoria gramatical substantivo, o algoritmo S_2 cometeu o dobro de erros de *overstemming* do que algoritmo S_1 ;
- Na categoria gramatical pronome, o algoritmo S_1 não processou a maior parte das ocorrências. A esse fato foi atribuído erro de *overstemming*. Por esta razão o valor apresenta-se bastante elevado em relação ao total de palavras nesta categoria. Com isso, o algoritmo S_1 teve seu pior desempenho na categoria gramatical pronome;
- O pior desempenho do algoritmo S_2 foi na categoria gramatical contração de preposição, na qual o algoritmo processou de forma equivocada 42% das palavras.
- Para as categorias pronome, contração de preposição e advérbio, ambos os algoritmos não apresentaram erros de *understemming*, exceto uma ocorrência para o algoritmo S_1 na categoria advérbio;
- O algoritmo S_1 apresentou a melhor precisão para a categoria substantivo processando 70% das palavras corretamente;
- O algoritmo S_2 obteve a melhor precisão para a categoria advérbio processando mais de 80% das palavras corretamente;
- O algoritmo S_2 apresentou um desempenho melhor que o algoritmo S_1 em todas as categorias gramaticais. O percentual total de acerto de S_2 foi de 78% contra 63,8% de S_1 . Este melhor desempenho do algoritmo S_2 pode ser atribuído ao fato de o mesmo já ter sido testado em um conjunto de palavras bem maior do que o S_1 , conforme estudo de [8]. Quanto maior o número de palavras distintas processadas por um algoritmo de *stemming* maior a probabilidade de se aumentar as listas de exceções do algoritmo.

Tanto os resultados apresentados neste artigo, quanto os realizados por [8], constataam a dificuldade para se obter uma precisão próxima aos 100% esperados. Entretanto, estes algoritmos, que são os primeiros encontrados na literatura

desenvolvidos especificamente para a língua portuguesa, servem como uma base consistente para estudos futuros.

4 Considerações Finais e Trabalhos Futuros

Neste artigo foram apresentados conceitos voltados ao processo de *stemming*. Na análise de precisão dos *stemmers* para a língua portuguesa foi possível observar que esta área do PLN ainda está pouco desenvolvida para o português, mesmo levando em consideração o pequeno grupo de palavras selecionadas. Procurou-se verificar a precisão de cada algoritmo nas classes gramaticais selecionadas no experimento. Diante disso, como trabalhos futuros, pode-se realizar uma análise de precisão com um número maior de textos e/ou aplicar os algoritmos a um sistema de RI e avaliar esse sistema com e sem a utilização de um algoritmo de *stemming*.

Agradecimentos

Ao professor Marco Antonio Insaurriaga Gonzalez e a doutoranda da Universidade de Middlesex, Londres, Viviane Moreira Orenge pelo fornecimento dos algoritmos.

Referências

1. Honrado , A., Leon, R., O'Donnel, R., Sinclair, D.: A Word Stemming Algorithm for the Spanish Language. In Proceedings of the Seventh International Symposium on String Processing Information Retrieval (SPIRE'00), A Coruña, Spain, September 27 - 29, (2000).
2. Arampatzis, A., van der Weide, Th.P., van Bommel, P., Koster, C. H. A.: Linguistically-motivated Information Retrieval. Encyclopedia of Library and Information Science. Published by Marcel Dekker, Inc. - New York – Basel, Vol. 69 (2000) 201-222.
3. Bechara, E. Moderna Gramática Portuguesa. Rio de Janeiro. (2001) 672.
4. Spark-Jones, K., Willet, P.: Readings in Information Retrieval. San Francisco: Morgan Kaufmann, 1997.
5. Frakes, W. B., Baeza-Yates, R.: Information retrieval: data structures algorithms. ed. Upper Saddle River, NJ : Prentice Hall, (1992) 504.
6. Strzalkowski, T. (ed.): Natural language information retrieval. Dordrecht: Text, Speech and Language Technology; v.7. Kluwer Academic. (1999) 384.
7. Baeza-Yates, R.: Modern information retrieval. New York, N.Y: Addison-Wesley, (1999) 513.
8. Orenge, V. M., Huyck, C.. A Stemming Algorithm for Portuguese Language. In: Proceedings of Eighth Symposium on String Processing and Information Retrieval (SPIRE 2001), Chile, (2001) 186-193.