UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

FELIPE NUNES FLORES

Avaliando o Impacto da Qualidade de um Algoritmo de Stemming na Recuperação de Informações

Trabalho de Graduação.

Prof. Dra. Viviane Pereira Moreira
Orientadora

Prof. Dr. Carlos Alberto Heuser
Co-orientador

Porto Alegre, dezembro de 2009.

AGRADECIMENTOS

Agradeço aos meus pais, irmãos, tios, avós e ao restante de minha família, pela educação que me deram e por terem sempre me tratado com muito amor e carinho, me apoiando durante essa caminhada.

Agradeço, também, aos meus colegas e amigos que tive nesses cinco anos de faculdade, que muito me ajudaram a vencer essa etapa com sucesso. Além disso, agradeço aos meus antigos e atuais colegas de trabalho do PET, Axur e CREA, os quais me proporcionaram muito aprendizado e crescimento.

Por fim, gostaria de expressar minha gratidão aos meus orientadores, Viviane Moreira e Carlos Heuser, pelo grande e valioso apoio e suporte que me deram durante toda a confecção desse trabalho. Sem sua ajuda e conselhos, a jornada teria sido bem mais árdua.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

AvP	Precisão Média
CA	Coeficiente Angular
CL	Coeficiente Linear
CLEF	*Cross-Language Evaluation Forum*
DMT	*Desired Merge Total*
DNT	*Desired Non-Merge Total*
ERRT	*Error Rate Relative to Truncation*
GDMT	*Global Desired Merge Total*
GDNT	*Global Desired Non-Merge Total*
GUMT	*Global Unachieved Merge Total*
GWMT	*Global Wrongly Merged Total*
MAP	*Mean Average Precision*
NIST	*National Institute of Standards and Technology*
OI	*Overstemming Index*
RI	Recuperação de Informações
RSLP	Removedor de Sufixos da Língua Portuguesa
SW	*Stemming Weight*
TREC	*Text Retrieval Conference*
UI	*Understemming Index*
UMT	*Unachieved Merge Total*
WMT	*Wrongly Merged Total*

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A qualidade de um algoritmo de *stemming* é tipicamente medida de duas formas: *(i)* quão corretamente o algoritmo mapeia as diferentes formas de uma palavra para o mesmo *stem*; ou *(ii)* qual o nível de melhora que o algoritmo proporciona à Recuperação de Informações. O presente trabalho apresenta o desenvolvimento de uma ferramenta que implementa o método de Paice, o qual serve para avaliar os algoritmos de *stemming* de acordo com a primeira métrica mencionada acima. Além disso, diversos algoritmos de *stemming* para a língua portuguesa são avaliados de acordo com as duas métricas, a fim de verificar se os *stemmers* de maior qualidade são também aqueles que trazem o maior ganho para a Recuperação de Informações. Os resultados mostram que essa relação existe, porém ela não é tão forte quanto se poderia esperar.

Palavras-Chave: stemming, método de Paice, recuperação de informações, avaliação.

Assessing the Impact of a Stemming Algorithm's Accuracy on Information Retrieval

ABSTRACT

The quality of a stemming algorithm is typically measured in two ways: *(i)* how accurately the algorithm maps the different forms of a word to the same stem; or *(ii)* the level of improvement that the algorithm provides to Information Retrieval. This paper presents a tool that implements Paice's method, whose purpose is to evaluate stemming algorithms according to the first metric mentioned above. Besides, different Portuguese stemming algorithms are evaluated according to both metrics, in order to assess whether the most accurate stemmers are also the ones that bring the most gain to Information Retrieval. The results show that this relation does exist, but it is not as strong as one might have expected.

Keywords: stemming, Paice's method, information retrieval, evaluation.

1 INTRODUÇÃO

Stemming é o processo de conflação[1] de formas diferentes de uma palavra em uma representação única. Essa representação é chamada de *stem*. Por exemplo, um *stemmer* poderia mapear as palavras *abastecimento*, *abastecida* e *abastecem* para *abastec*. O *stem* resultante não precisa ser uma palavra válida do idioma, porém deve conter o significado base original de suas palavras.

Conforme Frakes & Baeza-Yates (1992), s*temming* é bastante usado em um contexto de Recuperação de Informações (RI) com o intuito de aumentar a revocação das consultas, pois o usuário entra com um termo (ex: *abastecimento*) e o sistema torna-se capaz de retornar mais documentos relevantes, visto que também retorna os que tenham palavras similares (ex: *abastecida* e *abastecem*). Além disso, *stemming* também reduz o tamanho dos arquivos de índices, pois um *stem* pode corresponder a várias palavras distintas e, portanto, menos índices precisam ser guardados.

Existem diversas técnicas para realizar o processo de *stemming*, como o uso de dicionários, variedade de sucessores e n-gramas. No entanto, a técnica mais comum é a de remoção de afixos da palavra. Essa técnica retira prefixos e/ou sufixos da palavra através da aplicação sucessiva de um conjunto de regras até chegar ao *stem* resultante. A técnica de remoção de afixos, diferentemente do método de n-gramas, por exemplo, é dependente do idioma para o qual foi desenvolvida. Neste trabalho, os *stemmers* que serão avaliados usam essa técnica de remoção de afixos.

Já em relação à qualidade de um algoritmo de *stemming*, ela é tipicamente medida em uma dessas duas maneiras: *(i)* quão corretamente o *stemmer* mapeia palavras que são morfológica e semanticamente relacionadas para o mesmo *stem*; ou *(ii)* qual a melhora que o *stemmer* traz à RI.

Vários estudos tratam do uso de *stemming* em um sistema de RI, especialmente para a língua inglesa, como Harman (1991), Hull (1996) e Krovetz (1993). Outros abordam a qualidade e correção de um algoritmo de *stemming*, usando métricas de um método proposto por Paice (1994), como Alvares et al (2005), Kraaij & Pohlmann (1995) e Orengo & Huyck (2001). Entretanto, um estudo sobre o impacto da precisão de um algoritmo de *stemming* na performance da RI ainda não foi feito. Dessa forma, o

[1] O termo 'conflação' é empregado em lugar do inglês *conflation*, por não haver uma boa tradução da palavra para o português. Entendemos por conflação de dois ou mais termos seu uso indistinto para uma determinada finalidade, conforme Freund (1982).

principal objetivo deste trabalho é encontrar a resposta da seguinte pergunta: o *stemmer* com mais qualidade é o melhor para fins de RI?

Foram feitos experimentos com todos os *stemmers* do português encontrados na literatura, para medir suas precisões e também para medir o ganho que trazem para RI. Assim sendo, como um subproduto, este trabalho identifica o *stemmer* mais preciso e o que resulta na maior melhora da qualidade de RI.

Para poder realizar os experimentos do método de Paice, foi também desenvolvida, neste trabalho, uma ferramenta que implementa o método, a qual calcula todas as métricas propostas por ele.

1.1 Objetivos

Os objetivos deste trabalho são:

a) estudar diferentes algoritmos de *stemming* para o português já propostos na literatura;

b) analisar e implementar o método de Paice, construindo uma ferramenta;

c) avaliar o resultado do método de Paice para os diferentes *stemmers*;

d) avaliar os mesmos *stemmers* em um contexto de RI;

e) analisar a relação entre os resultados obtidos no método de Paice e na RI.

1.2 Organização do Trabalho

O restante deste trabalho encontra-se organizado da seguinte maneira:

a) Capítulo 2: neste capítulo, é feita uma revisão bibliográfica, mostrando trabalhos relacionados e introduzindo alguns conceitos e medidas de qualidade usados na avaliação de algoritmos de *stemming*;

b) Capítulo 3: aqui são apresentados os diferentes algoritmos de *stemming* para o português que serão avaliados, com uma breve explicação de como cada um funciona;

c) Capítulo 4: este capítulo mostra a implementação da ferramenta desenvolvida para o método de Paice, contendo detalhes de suas características e de como usá-la;

d) Capítulo 5: aqui são mostrados todos os experimentos e resultados obtidos, já com a correlação entre Paice e RI;

e) Capítulo 6: são apresentadas as conclusões do trabalho.

2 MEDIDAS DE QUALIDADE

Neste capítulo, serão apresentados conceitos usados na avaliação de algoritmos de *stemming*, cujo entendimento é necessário para o acompanhamento do restante do trabalho. Além disso, também mostraremos trabalhos que tratam de assuntos correlatos ao presente trabalho.

Paice (1994) propôs um método de avaliação da qualidade de um algoritmo de *stemming* usando quatro métricas:

- ***Overstemming Index*** (OI), o qual calcula o número de vezes que um *stemmer* erroneamente remove partes do *stem* como se fossem partes de um sufixo. Esse tipo de erro vai tipicamente fazer com que palavras não relacionadas sejam combinadas, como por exemplo, *adoção* e *adoçantes* serem ambas reduzidas para *adoç*. O OI é 0 quando não há erros de *overstemming* e 1 quando todas as palavras são reduzidas para o mesmo *stem*. Em RI, um alto OI vai potencialmente levar a uma redução da precisão, isto é, muitos documentos não relevantes serão retornados em uma consulta.
- ***Understemming Index*** (UI), o qual calcula o número de vezes que o *stemmer* falha em remover um sufixo. Esse tipo de erro vai tipicamente impedir que palavras relacionadas sejam combinadas. Por exemplo, pode ocorrer que *jornalista* seja reduzida para *jornalist* e *jornalismo* seja reduzida para *jornalism*. O UI é 0 quando não há erros de *understemming* e 1 quando nenhuma palavra é corretamente combinada pelo *stemmer*. Em RI, erros de *understemming* vão potencialmente levar a uma redução da revocação, ou seja, muitos documentos relevantes não serão retornados em uma consulta.
- ***Stemming Weight*** (SW), que é a razão OI / UI e representa o "peso" do *stemmer*. Um *stemmer* "pesado" aplica várias regras de remoção de afixos e, portanto, tem um OI alto e um UI baixo. Já um *stemmer* "leve" aplica poucas regras de remoção de afixos e, dessa forma, tem um OI baixo e um UI alto. Ter um SW alto ou baixo não implica em ser um *stemmer* melhor ou pior, visto que o SW é usado para medir apenas o "peso" do *stemmer* e não sua qualidade.
- ***Error Rate Relative to Truncation*** (ERRT). A ideia é que os valores dos pontos (UI, OI) de uma série de algoritmos de truncamento de palavras de diferentes tamanhos (exemplo: truncar até a 1ª letra, até a 4ª letra, até a 7ª letra, etc.) determinam uma linha com a qual a qualidade de um *stemmer* pode ser medida. As coordenadas (UI, OI) de um bom *stemmer* deveriam estar abaixo dessa linha de truncamento (ver Figura 2.1). Quanto mais distante da linha de truncamento, melhor é o *stemmer*, contanto que esteja

abaixo dessa linha. Caso contrário, quanto mais afastado, pior é o *stemmer*. O ERRT é obtido estendendo uma linha da origem, passando pelo ponto P (UI, OI), até que ela interseccione a linha de truncamento no ponto T. O ERRT é, então, calculado através da seguinte fórmula: ERRT = tamanho(OP) / tamanho(OT). Quanto menor for o ERRT, melhor é o *stemmer*.

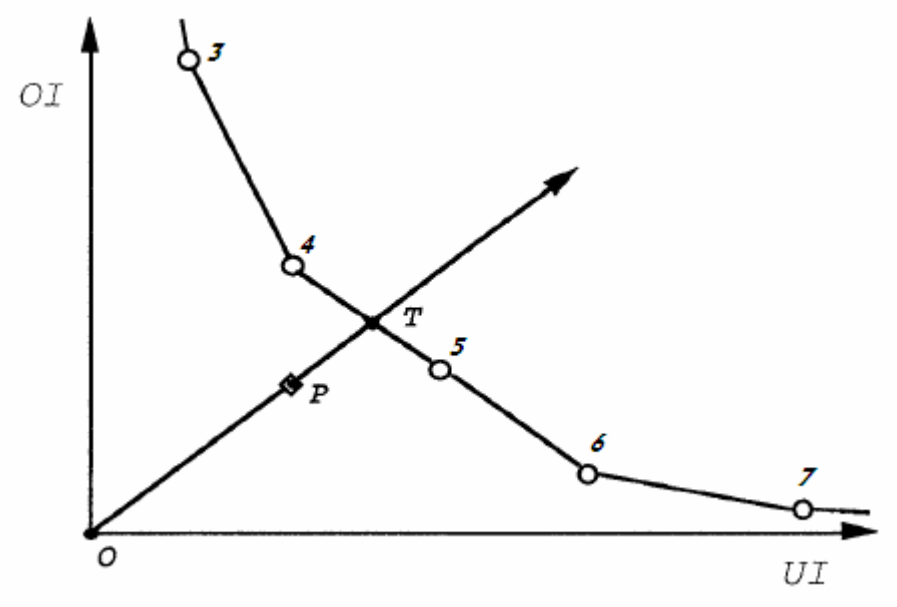

Figura 2.1: Computação do ERRT (PAICE 1994 adaptado)

Paice (1994) também usa o seu método para avaliar três *stemmers* para o inglês: Lovins (1968), Paice (1990) e Porter (1980). Ele conclui que o *stemmer* de melhor qualidade é Paice, seguido de Porter. O autor também observa que Porter é mais leve do que Paice. Esse estudo fez testes com amostras de diferentes tamanhos (de 2654 a 9757 palavras) e concluiu que o padrão de valores era similar para todos os tamanhos de amostra. O estudo não comparou o efeito dos *stemmers* na RI.

Kraaij & Pohlmann (1995) aplicaram o método de avaliação de Paice para medir a qualidade de uma versão holandesa do algoritmo de Porter. Novamente, esse estudo também não testou o efeito do *stemmer* na performance da RI.

Para o português, Orengo & Huyck (2001) usam o método de Paice para comparar o seu algoritmo de *stemming* proposto no artigo, chamado RSLP, e a versão para o português do *stemmer* de Porter. Eles concluem que o RSLP tem maior qualidade que o Porter para o português. Também para a língua portuguesa, Alvares et al (2005) compara, usando o método de Paice, os *stemmers* STEMBR (proposto no artigo), RSLP e Porter para o português. O estudo conclui que o STEMBR é o mais pesado dos três e, portanto, faz um pouco menos de erros de *understemming* que o RSLP e o Porter. Ambos os trabalhos, entretanto, não usaram a métrica do ERRT na sua avaliação. Além disso, nenhum dos dois avalia o efeito de seus *stemmers* na RI.

Há uma série de trabalhos que estudam o impacto de *stemming* em um sistema de RI, sendo que os mais proeminentes foram Harman (1991), Krovetz (1993) e Hull (1996). A conclusão geral é de que, olhando para o resultado médio de várias consultas, a melhoria trazida pelo *stemming* é pequena. Contudo, para algumas consultas, o *stemming* traz uma grande melhora. Para a língua portuguesa, Orengo et al (2007) avalia

os efeitos de três *stemmers* diferentes na RI e conclui que o *stemmer* leve é a melhor alternativa. Esse estudo, no entanto, não avaliou a qualidade dos *stemmers* usando o método de Paice.

Até onde sabemos, a comparação entre a qualidade de um algoritmo de *stemming* e sua eficácia em um sistema de RI ainda não foi feita. Portanto, o principal objetivo deste trabalho é preencher essa lacuna e investigar a relação entre esses dois indicadores de qualidade.

Nas próximas duas seções, serão apresentados mais conceitos relativos ao trabalho.

2.1 Métodos de avaliação na Recuperação de Informações

Para avaliar o impacto de *stemming* sobre a RI, a abordagem padrão é comparar uma execução base em que nenhum algoritmo de *stemming* é aplicado com uma execução experimental em que o *stemmer* é usado como um passo de pré-processamento sobre os documentos e as consultas. Existem duas medidas simples que são usadas para testar a qualidade dos resultados retornados para uma determinada consulta, quais sejam:

- **Revocação**: é a razão entre o número de documentos relevantes retornados pelo sistema para uma consulta e o número de documento relevantes que existem na base de dados para aquela consulta. A revocação é sempre um valor entre 0 e 1. Quando a revocação atinge 1, significa que todos os documentos relevantes para a consulta foram retornados pelo sistema.
- **Precisão**: é a razão entre o número de documentos relevantes retornados e o número total de documentos retornados (contando os relevantes e não-relevantes). A precisão é sempre um valor entre 0 e 1. Quando a precisão atinge 1, significa que todos os documentos retornados pelo sistema para a consulta foram relevantes.

Além dessas duas, existe uma medida que é mais usada, chamada de *precisão média* (AvP), a qual é calculada pela equação da figura 2.2. A AvP dá mais ênfase para o retorno de documentos relevantes nas primeiras posições do ranking.

$$AvP = \frac{\sum_{r=1}^{k} (P(r) \times \mathrm{Re}l(r))}{\sum_{r=1}^{k} (\mathrm{Re}l(r))}$$

Figura 2.2: Equação da precisão média

Na equação acima, *k* é o número de documentos retornados, *r* é uma posição do ranking, Rel( ) é uma função binária de uma posição de ranking dada (1 se o elemento retornado é relevante ou 0, caso contrário) e P( ) é a precisão para uma dada posição do ranking. Quando mais de uma consulta é usada, é necessário calcular a média das precisões médias, a qual é chamada de *mean average precision* (MAP). O MAP é bastante usado para avaliar os resultados obtidos em um sistema de RI. Para computar as medidas de qualidade de RI, é necessário usar um conjunto de teste composto por documentos, consultas e listas que indiquem quais os documentos relevantes de cada consulta.

2.2 Detalhamento do Método de Paice

No início deste capítulo, vimos quais os resultados que o método de Paice disponibiliza: UI, OI, SW e ERRT. Os valores de SW e ERRT podem ser calculados a partir do UI e do OI, conforme explicado anteriormente. No entanto, nesta seção, serão apresentados os detalhes de como se chegar a esses valores de UI e OI.

Primeiramente, deve-se pegar uma amostra de palavras e, manualmente, dividi-las em grupos conceituais que contenham palavras que sejam morfológica e semanticamente relacionadas. O *stemmer* de melhor qualidade seria aquele que reduzisse todas as palavras de um mesmo grupo para o mesmo *stem* e esse *stem* fosse único, isto é, nenhuma palavra de nenhum outro grupo deve ter o mesmo *stem*.

Abaixo temos um exemplo de palavras separadas por grupos conceituais e os respectivos *stems* gerados por um *stemmer* hipotético. Usaremos esse exemplo para a explicação do cálculo dos índices do método. Note que, no exemplo, apenas o grupo 1 foi corretamente conflacionado. No grupo 2, houve erros de *understemming*, visto que nem todas as palavras tiveram o mesmo *stem*, enquanto que no grupo 3 e 4 houve erros de *overstemming*, já que palavras de grupos diferentes ficaram com o mesmo *stem*.

Originais:

- Grupo 1: adaptações, adaptada, adaptar, adaptou
- Grupo 2: adiciona, adicionada, adicionais, adicionar, adicionasse
- Grupo 3: falamos, falante, falar
- Grupo 4: falência, falido

Stems:

- Grupo 1: adapt, adapt, adapt, adapt
- Grupo 2: adic, adicion, adic, adicion, adicion
- Grupo 3: fal, fal, fal
- Grupo 4: fal, fal

2.2.1 DMT, DNT, GDMT, GDNT

Primeiramente, para cada grupo conceitual, devem-se calcular dois totais: *Desired Merge Total* (DMT) e *Desired Non-Merge Total* (DNT). O DMT é o número total de pares de palavras que podem ser formados a partir da combinação das palavras do grupo entre si, pois esse é exatamente o número de pares de mesmo *stem* que queremos ter (porque queremos que todas as palavras dentro do grupo tenham o mesmo *stem*). A fórmula para o cálculo do DMT encontra-se na figura 2.3. O grupo 2, por exemplo, como possui 5 palavras, teria um DMT de 10.

$$DMT_g = 0{,}5 n_g \, (n_g - 1)$$

Figura 2.3: Fórmula do DMT

Na fórmula acima, n_g representa o número de palavras do grupo conceitual. Já o DNT representa o número de pares que podemos formar entre uma palavra pertencente ao grupo e outra não-pertencente ao grupo, pois esse é exatamente o número de pares de

palavras que não queremos que tenham o mesmo *stem*. A fórmula para o cálculo do DNT encontra-se na figura 2.4. O grupo 1, por exemplo, como possui 4 palavras (e a amostra possui 14), teria um DNT de 20.

$$DNT_g = 0{,}5 n_g \, (W - n_g)$$

Figura 2.4: Fórmula do DNT

Na fórmula acima, W representa o número de palavras da amostra. Se somarmos os DMTs de todos os grupos conceituais, teremos um *Global Desired Merge Total* (GDMT) para a amostra. Da mesma forma, ao somar os DNTs de todos os grupos conceituais, achamos o *Global Desired Non-Merge Total* (GDNT) da amostra. O GDMT e o GDNT serão usados em cálculos posteriores, da próxima subseção.

Tabela 2.1: Exemplo de cálculo do DMT, DNT, GDMT e GDNT

Grupo	DMT	DNT
1	6	20
2	10	22,5
3	3	16,5
4	1	12
Total (GDMT / GDNT)	20	71

2.2.2 UMT, GUMT, UI

Após a aplicação do *stemmer* na amostra, alguns grupos ainda contêm dois ou mais *stems* distintos, como no grupo 2, o que significa que há erros de *understemming*. O *Unachieved Merge Total* (UMT) representa o número de erros de *understemming* e é calculado pela fórmula da figura 2.5 abaixo.

$$UMTg = 0,5 \sum_{i=1..s} u_i \left(n_g - u_i\right),$$

Figura 2.5: Fórmula do UMT (ALVARES 2005 adaptado)

Na fórmula acima, n_g representa o número de palavras do grupo, s é o número de stems distintos e u_i (u_1, u_2, ..., u_s) é o número de instâncias de cada *stem* no grupo. O grupo 2, por exemplo, teria um UMT de 6 (0,5 * ((2 * (5 – 2)) + (3 * (5 – 3)))). Somando os UMTs de todos os grupos, temos o *Global Unachieved Merge Total* (GUMT) da amostra.

Tabela 2.2: Exemplo de cálculo do UMT e GUMT

Grupo	UMT
1	0
2	6
3	0
4	0
GUMT	6

Para calcular o UI, que é um dos quatro resultados finais que o método de Paice retorna, simplesmente efetuamos a divisão: UI = GUMT / GDMT. Calculando o UI da amostra, temos UI = 6 / 20 = 0,3.

2.2.3 WMT, GWMT, OI

Após o processo de *stemming*, também há casos em que um mesmo *stem* aparece em dois ou mais grupos conceituais diferentes, o qual resulta em erros de *overstemming*. Para contabilizá-los, a amostra deve ser reorganizada em grupos que compartilham o mesmo *stem*, conforme o exemplo abaixo, no qual o número entre parênteses indica o grupo original da palavra que gerou o *stem*:

- Grupo *adapt*: adapt (1), adapt (1), adapt (1), adapt (1)
- Grupo *adic*: adic (2), adic (2)
- Grupo *adicion*: adicion (2), adicion (2), adicion (2)
- Grupo *fal*: fal (3), fal (3), fal (3), fal (4), fal (4)

Feita essa reorganização, podemos então calcular o *Wrongly Merged Total* (WMT) de cada grupo gerado, o qual é obtido através da fórmula da figura 2.6 abaixo.

$$WMT_s = 0,5 \sum_{i=1..t} v_i (n_s - v_i),$$

Figura 2.6: Fórmula do WMT (ALVARES 2005 adaptado)

Na fórmula acima, n_s é o número de itens do grupo, t é o número de grupos conceituais originais diferentes que estão presentes no grupo e v_i é o número de itens de cada grupo original t. O grupo *fal*, por exemplo, teria um WMT igual a 6 (0,5 * ((3 * (5 – 3)) + (2 * (5 – 2)))). Somando os WMTs de todos os grupos, obtemos o *Global Wrongly Merged Total* (GWMT) da amostra. Para calcular o OI, que é mais um dos quatro resultados finais que o método de Paice retorna, basta efetuar a divisão: OI = GWMT / GDNT.

Tabela 2.3: Exemplo de cálculo do WMT e GWMT

Grupo	WMT
adapt	0
adic	0
adicion	0
fal	6
GWMT	6

Calculando o OI da amostra, temos OI = 6 / 71 ≈ 0,0845.

3 STEMMERS PARA A LÍNGUA PORTUGUESA

Neste capítulo, serão mostrados os diversos algoritmos de *stemming* para a língua portuguesa existentes, os quais serão usados nos experimentos do capítulo 5. O presente capítulo está organizado através de uma seção para cada *stemmer*.

3.1 Porter

O algoritmo de Porter (1980) foi projetado originalmente para a língua inglesa. Posteriormente, foi feita uma versão para o português, disponível em Porter (2007). O algoritmo se baseia em uma série de regras condicionais que são aplicadas em sequência. Uma regra é definida da forma mostrada na figura 3.1.

(condição) sufixo1 -> sufixo2

Figura 3.1: Regra do algoritmo de Porter

Na regra acima, *(condição)* é uma condição a ser testada para a aplicação da regra (por exemplo, tamanho mínimo do *stem* resultante após a remoção do *sufixo1*), *sufixo1* é o sufixo a ser procurado e removido da palavra e *sufixo2* é o sufixo a ser adicionado na palavra após a remoção do *sufixo1*. Assim sendo, a regra é aplicada se o *sufixo1* for encontrado na palavra e as condições de *(condição)* forem satisfeitas, caso em que o *sufixo1* será removido da palavra e substituído por *sufixo2*, se este estiver definido na regra.

Foram definidos 5 passos de regras no algoritmo, quais sejam:

- Passo 1: remoção de sufixos comuns (por exemplo: eza, ismos, ável, ível, oso, aço~es, mente, ância...)
- Passo 2: remoção de sufixos verbais, caso a palavra não tenha sido alterada pelo passo 1 (por exemplo: ada, ida, aria, ará, ava, isse, iriam, aram, endo, indo, ara~o, íreis, êssemos...)
- Passo 3: remoção do sufixo "i", se precedido de "c" e se a palavra foi alterada pelos passos 1 ou 2.
- Passo 4: remoção de sufixos residuais (os, a, i, o, á, í, ó) se nenhum dos passos anteriores alterou a palavra;
- Passo 5: remoção dos sufixos "e", "é" e "ê", tratamento do cedilha e das sílabas "gue", "gué" e "guê".

Além disso, antes das aplicações das regras, as palavras que possuem "ã" e "õ" são substituídas por "a~" e "o~". Ao final da aplicação das regras, elas retornam para "ã" e "õ" para formar o *stem* resultante.

Nos experimentos do capítulo 5, esse *stemmer* será referenciado como sendo **Porter**.

3.2 RSLP

O Removedor de Sufixos da Língua Portuguesa (RSLP) foi desenvolvido por Orengo & Huyck (2001) e, posteriormente, aprimorado por Coelho (2007). Assim como Porter, o RSLP também aplica sucessivos passos de remoção de sufixos, os quais são definidos através de regras. No entanto, ele possui mais regras que o Porter, por ter sido desenvolvido especificamente para o português, além de possuir um dicionário de exceções, o qual evita remover sufixos de palavras cuja terminação é similar a um sufixo, mas, na verdade, faz parte de seu radical (por exemplo, o sufixo aumentativo *–ão* deve ser retirado de *cachorrão*, mas não de *coração*).

Uma regra, no RSLP, é declarada como no exemplo da figura 3.2 abaixo.

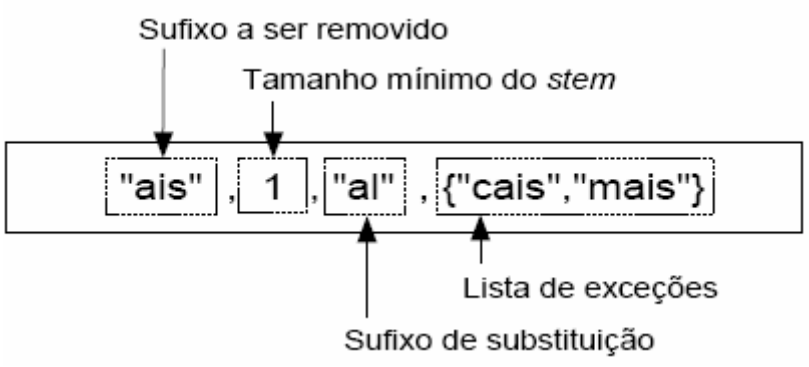

Figura 3.2: Exemplo de regra do RSLP (COELHO 2007)

No caso, para uma determinada regra, primeiramente verifica-se se o sufixo a ser removido (no exemplo, "ais") está presente na palavra a ser processada. Em caso afirmativo, verifica-se se a palavra não está na lista de exceções da regra (no exemplo, "cais" e "mais"). Em não estando na lista de exceções, é testado se o *stem* resultante da remoção do sufixo possui o tamanho mínimo do *stem* definido pela regra (no exemplo, 1). Se possuir o tamanho mínimo, então a regra pode ser aplicada. Nesse caso, remove-se o sufixo, adiciona-se o sufixo de substituição, se existir (no exemplo, "al"), e o algoritmo continua no próximo passo de remoção de sufixos. Dentro de um mesmo passo de redução, as regras estão ordenadas de tal forma que os maiores sufixos são testados primeiro, a fim de que, por exemplo, o sufixo *–ão*, da redução do aumentativo, não seja incorretamente retirado da palavra antes do sufixo *–zão* ser testado.

Os passos de redução do RSLP, na ordem em que são executados, são os seguintes, conforme figura 3.3:

- **Redução do Plural**: remove os sufixos de plural das palavras, como *–ns*, *–ães*, *–óis* e *–s*. A palavra "anzóis", por exemplo, torna-se "anzol" nesse passo. Possui 11 regras definidas.
- **Redução do Feminino**: modifica o gênero da palavra de feminino para masculino. A palavra deve terminar com "a" para entrar nesse passo. A palavra "receosa", por exemplo, torna-se "receoso" nesse passo. Ele possui 15 regras definidas.

- **Redução Adverbial**: esse passo apenas retira o sufixo "mente" dos advérbios. A palavra "alegremente" torna-se "alegre" nesse passo. Possui apenas 1 regra.
- **Redução do Aumentativo**: modifica o grau da palavra, de aumentativo, diminutivo ou superlativo, para o grau normal, retirando sufixos como *–inho*, *–íssimo*, e *–ão*. A palavra "belíssimo", por exemplo, vira "bel" nesse passo. Possui 23 regras definidas.
- **Redução Nominal**: esse passo retira sufixos comuns de substantivos e adjetivos, como *–izado*, *–ência*, *–ico* e *–ável*. A palavra "confortável" torna-se "confort" nesse passo. Possui 84 regras definidas. Caso a palavra seja reduzida nesse passo, o algoritmo pula os próximos dois passos (Verbal e de Vogais).
- **Redução Verbal**: remove os sufixos dos verbos regulares do português. Possui 101 regras diferentes. A palavra "engordassem" torna-se "engord" nesse passo. Caso a palavra seja reduzida nesse passo, o algoritmo pula o próximo passo (de Vogais).
- **Redução de Vogais**: remove a última vogal "a", "e" ou "o", caso a palavra não tenha sido reduzida pelos últimos dois passos. A palavra "belo", por exemplo, ainda não teria sido modificada e, nesse passo, vai para a forma "bel", ficando com o mesmo *stem* de "belíssimo", o qual foi reduzido no passo de redução do aumentativo.
- **Remoção de Acentos**: esse passo final simplesmente remove os acentos das palavras.

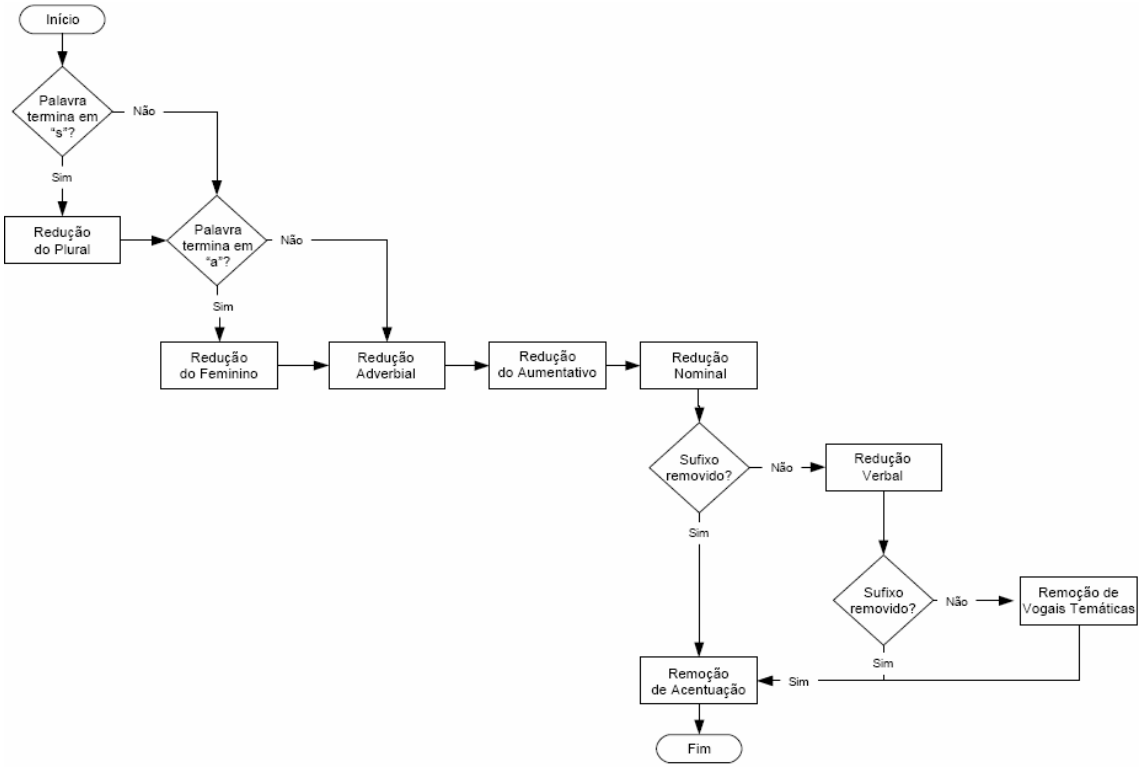

Figura 3.3: Ordem dos passos do algoritmo RSLP (ORENGO & HUYCK 2001 adaptado)

Além disso, o RSLP também possui uma funcionalidade chamada Dicionário de Nomes Próprios, na qual o usuário pode fazer uma lista de nomes próprios que o *stemmer* desconsidera na hora da aplicação das regras. Nos experimentos do capítulo 5, o *stemmer* completo, isto é, com os oito passos de redução, será referenciado como **RSLP**. Outrossim, também serão realizados experimentos com uma versão mais "leve" do algoritmo, a qual só usa o passo da Redução do Plural. Essa forma "light" do RSLP será referenciada como **RSLP-S**.

3.3 STEMBR

O STEMBR foi proposto por Alvares et al (2005) e diferencia-se dos demais por incluir também o tratamento da remoção de prefixos e de verbos irregulares. No algoritmo, são realizadas três etapas em sequência: Tratador de Exceção, Tratador Prefixal e Tratador Sufixal, conforme figura 3.4.

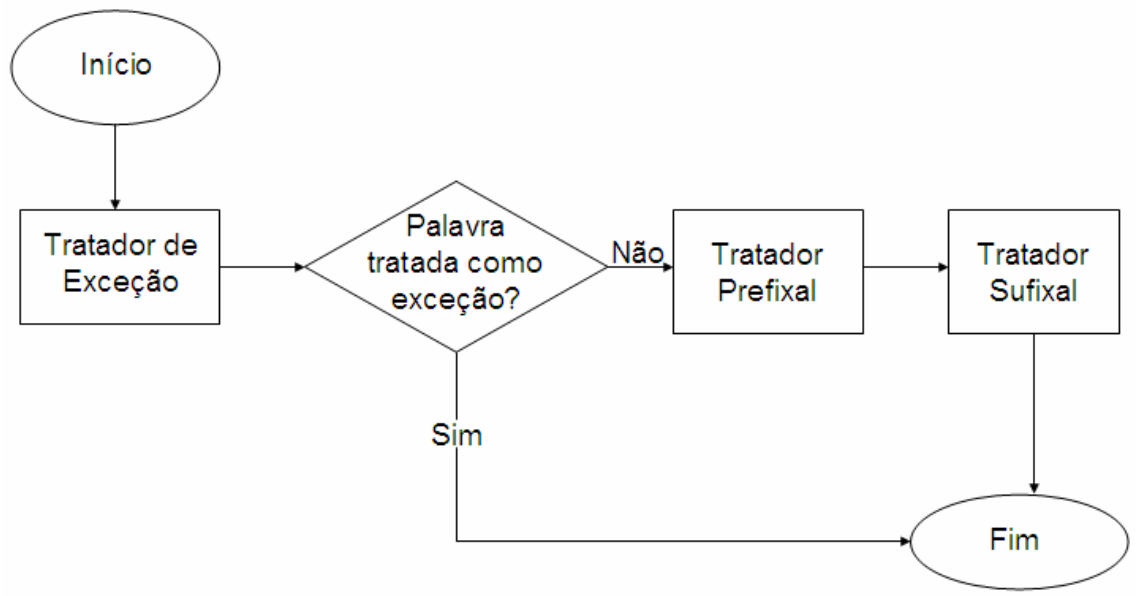

Figura 3.4: Sequência de etapas do STEMBR

O Tratador de Exceção verifica se a palavra a ser analisada encontra-se em uma de duas exceções: *stop words* ou verbos irregulares. As *stop words* são uma lista de palavras de exceção que o *stemmer* possui. Caso a palavra a ser analisada seja uma *stop word*, nenhum processamento é realizado nela e a palavra original vira o *stem*. Se a palavra analisada estiver na lista de verbos irregulares do STEMBR, o algoritmo retornará a forma infinitiva armazenada para esse verbo como o *stem*. Somente se a palavra não estiver em nenhuma das duas listas (*stop words* ou verbos irregulares) é que o processamento continuará na próxima etapa, o Tratador Prefixal.

O objetivo do Tratador Prefixal é retirar um possível prefixo que a palavra pode ter. Ele simplesmente compara o *substring* inicial de cada palavra com uma lista de prefixos predefinidos, removendo-o, caso o prefixo seja encontrado. Para evitar que palavras como "imagem" tenham o prefixo *im-* erroneamente removidos, o *stemmer* possui uma lista de exceções para essa etapa, que contém os *stems* de palavras cujo *substring* inicial não é um prefixo. Adicionalmente, prefixos que geram muitas exceções não são tratados. A tabela 3.1 mostra alguns prefixos e suas exceções. Independentemente de o prefixo ser ou não retirado, o Tratador Sufixal é executado após o Tratador Prefixal.

Tabela 3.1: Exemplos de prefixos e suas exceções no STEMBR

Prefixo	***Stem* das exceções**	**Exceções**
ante	antecip	antecipar, antecipação
ciclo	ciclon	ciclone, ciclones
contra	contrari, contrast	contrariar, contraste
trans	transa, transp	transado, transpirar

Fonte: Alvares (2005)

Por fim, temos o Tratador Sufixal, cujo objetivo é retirar o sufixo da palavra. Ele usa algumas regras aproveitadas do RSLP, inserindo outras novas. Ao total, existem 394 regras. A regra do STEMBR possui a mesma estrutura da do RSLP, com a exceção de não possuir a parte de substituir um sufixo por outro, isto é, no STEMBR os sufixos são sempre removidos, enquanto que no RSLP o sufixo pode ser tanto removido quanto substituído.

Para aplicar as regras, o algoritmo ordena de forma decrescente por tamanho os sufixos de cada subconjunto, fazendo a substituição do maior sufixo que esteja presente na palavra, conforme figura 3.5. Além disso, foi definida uma ordem de execução do *stemmer*, a qual mostra os sufixos que possuem qual letra final devem ser executados primeiro, qual seja: “S”, “R”, “M”, “L”, “O”, “A”, “U”, “E” e “I”.

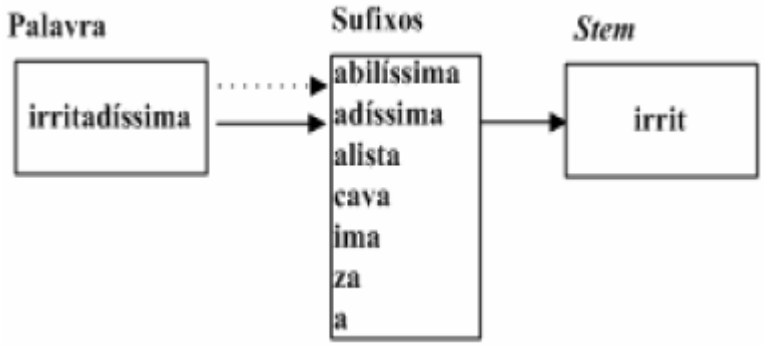

Figura 3.5: Retirada do maior sufixo da palavra no STEMBR (ALVARES 2005)

Nos experimentos do capítulo 5, esse *stemmer* será referenciado pelo nome de **StemBR**.

3.4 Savoy

Savoy (2006) apresenta *stemmers* para diversas línguas, entre elas o português[2]. Esse *stemmer* é mais simples que os três anteriores, pois possui menos regras e apenas tenta remover sufixos flexionais de substantivos e adjetivos, além do sufixo derivacional adverbial *–mente*.

O algoritmo de Savoy é dividido em quatro partes. A primeira parte possui várias regras de remoção do plural, como, por exemplo, a substituição dos sufixos *–is* por *–il* e

[2] Disponível em: http://members.unine.ch/jacques.savoy/clef/portugueseStemmer.txt Acesso em 15 nov. 2009.

–ns por *–m*. Dessa forma, "barris" se tornaria "barril" nessa etapa, por exemplo. Além disso, essa etapa também retira o sufixo *–mente* dos advérbios.

A segunda parte do *stemmer* realiza a transformação da palavra do gênero feminino para o masculino, através de regras como a substituição dos sufixos *–esa* por *–ês* e *–iva* por *–ivo*. Nessa segunda parte, portanto, a palavra "camponesa" seria transformada em "camponês".

A terceira parte do algoritmo simplesmente retira a vogal final da palavra, desde que essa vogal seja "o", "e" ou "a" e a palavra restante tenha mais que três caracteres. Por fim, a quarta parte do *stemmer* realiza a remoção dos acentos que a palavra restante tem para a formação do *stem* final.

No capítulo 5, esse *stemmer* será referenciado como **Savoy**.

3.5 Truncamento e Redução do "S"

Nesta seção, mostraremos dois *stemmers* bastante simples, que foram implementados pelo autor do presente trabalho, com o intuito único de utilizá-los nos experimentos do capítulo 5, isto é, eles não foram feitos para serem usados em um sistema de RI real.

O algoritmo Redução do "S" é um *stemmer* "*light*" que retira o "s" final da palavra a ser analisada, caso ela termine com essa letra. Não é feita nenhuma lista de exceções. Essa seria uma forma aproximada de fazer um *stemmer* que apenas remove o plural, como é o caso do RSLP-S. Assim sendo, a palavra "grandes" viraria "grande", "questões" viraria "questõe", "mas" seria transformado em "ma" e "quaisquer" ficaria intacta. Esse stemmer de Redução do "S" será referenciado no capítulo 5 como **Stemmer-S**.

O segundo algoritmo de *stemming* a ser apresentado nesta seção é o truncamento. Ele se baseia na remoção de todas as letras da palavra a ser analisada, exceto as *n* primeiras. Esse *n* pode variar desde 1 até um número tão alto quanto se queira. Nos experimentos do capítulo 5, iremos testar com *n* variando de 3 a 8, sendo o *stemmer* referenciado como **Trunc3**, **Trunc4**, **Trunc5**, **Trunc6**, **Trunc7** ou **Trunc8**, dependendo do valor de *n* utilizado. Como um exemplo, para esses valores de *n*, a palavra "abrangência" seria transformada em "abr", "abra", "abran", "abrang", "abrangê" e "abrangên", respectivamente.

Nos experimentos do capítulo 5, também será testada uma abordagem que não faz nenhum tipo de *stemming*, a qual será chamada de **NoStem**. Na tabela 3.2 a seguir, mostramos um exemplo do resultado do processo de *stemming* dos diferentes algoritmos para um mesmo texto de entrada.

Tabela 3.2: Resultado do processo de *stemming* para os diferentes *stemmers*

No Stem	O debate político, pelo menos o que vem a público, parece, de modo nada surpreendente, restrito a temas menores. Mas há, evidentemente, grandes questões em jogo nas eleições que se aproximam.
Porter	o debat polit, pel men o que vir a public, parec, de mod nad surpreendent, restrit a tem menor. mas hav, evident, grand questõ em jog nas eleiçõ que se aproxim.
RSLP	o debat politic, pel menos o que vem a public, parec, de mod nad surpreend, restrit a tem men. mas ha, evid, grand quest em jog na ele que se aproxim.
RSLP-S	o debate político, pelo menos o que vem a público, parece, de modo nada surpreendente, restrito a tema menor. mas há, evidentemente, grande questão em jogo na eleição que se aproximam.
Savoy	o debat politic, pelo meno o que vem a public, parec, de modo nada surpreendent, restrit a tema menor. mas ha, evident, grand questa em jogo nas eleica que se aproximam
Stemmer-S	o debate político, pelo meno o que vem a público, parece, de modo nada surpreendente, restrito a tema menore. ma há, evidentemente, grande questõe em jogo na eleição que se aproximam.
StemBR	o deba polit, pelo meno o que vem a public, parec, de modo nad surpreend, restrit a tem meno. ma ha, evident, grand quest em jogo na ele que se aproxim.
Trunc-4	o deba polí, pelo meno o que vem a públ, pare, de modo nada surp, rest a tema meno. mas há, evid, gran ques em jogo nas elei que se apro.

4 FERRAMENTA DESENVOLVIDA PARA O MÉTODO DE PAICE

Neste capítulo, será mostrada a ferramenta que foi desenvolvida neste trabalho para a avaliação dos algoritmos de *stemming* pelo método de Paice. Serão apresentados alguns detalhes de sua implementação e interface, mostrando como ela pode ser usada por outras pessoas que queiram avaliar *stemmers* diferentes dos apresentados aqui.

A ferramenta foi toda desenvolvida em C/C++ e baseada no método de Paice original, apresentado no artigo de Paice (1994) e explicado no capítulo 2 deste trabalho. A ferramenta foi feita não só com o intuito de ser usada nos testes do capitulo 5, mas também de ser utilizada por qualquer pessoa interessada em usar o método de Paice na avaliação de seus próprios *stemmers*. Dessa forma, o código do programa foi disponibilizado *online*[3] e será apresentada nesse capítulo a forma como a ferramenta deve ser utilizada.

É interessante destacar que a ferramenta é independente de idioma, isto é, ela pode ser usada para avaliar algoritmos de *stemming* de qualquer língua formada por palavras de caracteres latinos, como Inglês, Português, Espanhol, Italiano, Finlandês, etc.

4.1 Formato dos Arquivos de Entrada

Como entrada para o sistema, o usuário deve confeccionar, no mínimo, três arquivos texto. O primeiro arquivo, cujo nome deve ser digitado pelo usuário ao executar a ferramenta, é uma simples lista que deve conter o nome dos próximos arquivos que a ferramenta vai usar. Essa lista deve possuir uma formatação específica simples, conforme exemplo da figura 4.1 abaixo.

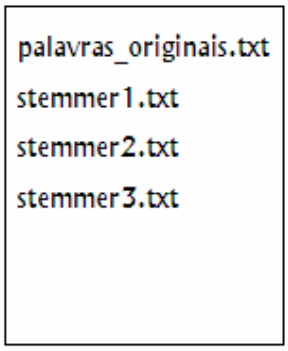

Figura 4.1: Exemplo de 1º arquivo de entrada

[3] Disponível no endereço http://www.inf.ufrgs.br/~fnflores/paice_tool

Conforme mostrado no exemplo acima, a primeira linha do arquivo deve conter o nome do arquivo que contém as palavras originais que serão usadas como base para os cálculos do método de Paice, no exemplo, *palavras_originais.txt*. Nas linhas subsequentes (da 2ª linha em diante), deve-se colocar o nome dos arquivos que contêm os *stems* relativos às mesmas palavras do arquivo de palavras originais, após elas terem sido processadas pelo *stemmer* que deve ser avaliado. Podem ser avaliados ao mesmo tempo desde um até quantos *stemmers* se queiram. No exemplo da figura 4.1, serão avaliados três diferentes algoritmos de *stemming*. Todos os arquivos nomeados na lista do arquivo de entrada devem estar no mesmo diretório no qual o programa está sendo executado.

Conforme explicado na seção 2.2, o método de Paice precisa de uma amostra de palavras divididas em grupos conceituais que sejam morfológica e semanticamente relacionadas. O arquivo que contém essas palavras da amostra (*palavras_originais.txt*, no exemplo da figura 4.1) deve seguir uma formatação específica: cada palavra deve ser separada por um *enter*, cada grupo deve ser separado por um * e o final do arquivo deve ser indicado por um **, conforme exemplo da figura 4.2.

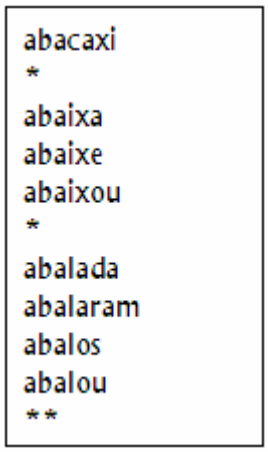

Figura 4.2: Exemplo de arquivo de palavras originais

Por fim, os arquivos que contêm os *stems* processados pelos *stemmers* (*stemmer1.txt*, *stemmer2.txt* e *stemmer3.txt* no exemplo da figura 4.1) também devem possuir exatamente a mesma formatação do arquivo de palavras originais, conforme é mostrado no exemplo da figura 4.3 a seguir.

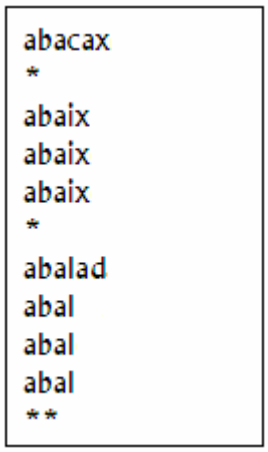

Figura 4.3: Exemplo de arquivo de *stems*

4.2 Detalhes da Implementação

Nesta seção, veremos alguns detalhes interessantes do código-fonte da ferramenta. O funcionamento do método de Paice como um todo já foi explicado anteriormente, porém aqui será visto como a ferramenta o implementou.

Primeiramente, o algoritmo lê o arquivo com as palavras originais, indicado no primeiro arquivo de entrada, e guarda essas palavras, separadas nos grupos conceituais, em uma estrutura de dados em memória. Após, ele irá ler os *stems* gerados pelos *stemmers* nos arquivos subseqüentes, a fim de efetuar os cálculos necessários para obter os resultados de cada *stemmer*.

É interessante ressaltar que alguns cálculos são feitos para todos os *stemmers*, como o UMT e o WMT de cada *stemmer*, por exemplo, porém outros são feitos apenas quando estamos trabalhando com o primeiro *stemmer* a ser avaliado. O principal exemplo desse caso é o cálculo do UI, OI e SW (e todos os cálculos necessários para chegar nesses valores) dos truncamentos. Sabemos que é necessário calcular esses índices para os truncamentos antes de calcular o ERRT de um *stemmer*, porém os valores desses truncamentos independem de qual *stemmer* está sendo avaliado. Dessa forma, eles são calculados somente uma vez no algoritmo, independente de quantos *stemmers* serão avaliados, a fim de economizar tempo de processamento.

Para calcular o UI, OI e SW de cada *stemmer*, primeiramente são obtidos os valores de DMT, DNT, GDMT, GDNT, UMT, GUMT, WMT e GWMT, conforme mostrado na seção 2.2. Para o cálculo do ERRT, são necessários os valores do truncamento. Para isso, a ferramenta calcula o UI, OI e SW para os truncamentos, desde o que mantém apenas a primeira letra da palavra, até o que deixa intactas as 12 primeiras letras. Esse número de 1 a 12 foi escolhido com base nos testes empíricos realizados, pois vimos que eles já seriam suficientes para calcular o ERRT de todos os *stemmers* avaliados.

Por fim, para calcular o ERRT, precisamos descobrir qual a linha que começa na origem, passa pelo ponto (UI, OI) do *stemmer* e vai até a linha de truncamento, além de calcular as distâncias entre esses pontos, conforme explicado no capítulo 2. Para a linha de truncamento, usamos uma aproximação como a da figura 2.1, em que a linha de truncamento é formada pela ligação entre os segmentos de reta limitados pelos pontos (UI, OI) de cada truncamento (1 e 2, 2 e 3, 3 e 4, e assim por diante, até 11 e 12). Dessa forma, cada segmento de reta é representado pelo seu Coeficiente Angular (CA) e Coeficiente Linear (CL), os quais são calculados, no nosso algoritmo, pelas fórmulas da figura 4.4 abaixo.

$$CA_n = \frac{OI_{n+1} - OI_n}{UI_{n+1} - UI_n} \qquad CL_n = OI_n - (CA_n * UI_n)$$

Figura 4.4: Fórmulas do Coeficiente Angular e Coeficiente Linear

Nas fórmulas acima, UI_n é o *understemming index* do truncamento das *n* primeiras letras, OI_n é o *overstemming index* do truncamento das *n* primeiras letras, UI_{n+1} e OI_{n+1} são os índices do truncamento das *n+1* primeiras letras, CA_n é o coeficiente angular do segmento de reta que vai do ponto (UI_n, OI_n) até (UI_{n+1}, OI_{n+1}) e CL_n é o coeficiente linear do mesmo segmento de reta.

De posse dos CA e CL de todos os segmentos de reta que compõem a linha de truncamento, temos agora que descobrir em que ponto a reta que passa pela origem e

pelo ponto (UI, OI) do *stemmer* avaliado intersecciona a linha de truncamento e, após, fazer o cálculo das distâncias necessárias para a obtenção do ERRT. Para isso, a ferramenta realiza o algoritmo representado no pseudocódigo da figura 4.5 abaixo.

```
Para n de 1 até 11
   X = CL_n / (SW - CA_n);
   Y = SW * X;
   Se UI_n <= X <= UI_n+1 // achamos o ponto correto
      Sair do laço "para";
   FimSe
FimPara

OP = sqrt((UI * UI) + (OI * OI)); // Distância Euclidiana entre (UI,OI) e a Origem
OT = sqrt((X * X) + (Y * Y)); // Distância Euclidiana entre (X,Y) e a Origem

ERRT = OP / OT;
```

Figura 4.5: Pseudocódigo do final do cálculo do ERRT

Como podemos perceber no pseudocódigo acima, primeiramente o programa calcula o ponto (X, Y) em que a reta que passa pelo ponto (UI, OI) do *stemmer* interseccionaria o segmento de reta representado pelo CA_n e CL_n, caso esse segmento tivesse extensão infinita (linha 2 e 3 da figura 4.5). Após isso, é verificado se o X desse ponto calculado realmente está dentro do segmento (linha 4 da figura 4.5). Em caso afirmativo, já achamos o ponto (X, Y) desejado e saímos do laço. Caso contrário, efetuamos novamente o cálculo de um novo ponto para o próximo valor de *n*. Executamos com *n* de 1 a 11 porque já haviam sido calculados anteriormente justamente esses 11 segmentos de reta.

De posse do ponto (X, Y), que é o ponto que intersecciona a linha de truncamento, calculamos os valores de OP e OT, através da fórmula da Distância Euclidiana, conforme as linhas 9 e 10 da figura 4.5. Finalmente, o ERRT é calculado, através da razão OP / OT.

4.3 Formato dos Arquivos de Saída

O objetivo principal da ferramenta é gerar os valores de UI, OI, SW e ERRT dos *stemmers* avaliados. Porém, além disso, a ferramenta dá três opções do nível de detalhamento que aparecerá no resultado, visto que pode vir a ser interessante para o usuário saber exatamente em quais casos o *stemmer* está errando, por exemplo. O arquivo com os resultados é sempre gerado no mesmo diretório no qual o programa está sendo executado e tem o nome de “resultados.txt”.

No nível de detalhamento Baixo, somente os quatro índices finais do método de Paice são mostrados para cada algoritmo de *stemming* avaliado, conforme exemplo da figura 4.6.

```
Algorithm #1:
UI: 0.1905226632
OI: 0.0002680360
SW: 0.0014068458
ERRT: 0.5691374097

Algorithm #2:
UI: 0.9515289525
OI: 0.0000004927
SW: 0.0000005178
ERRT: 0.9959400023

Algorithm #3:
UI: 0.3014530471
OI: 0.0001468286
SW: 0.0004870694
ERRT: 0.7159116889
```

Figura 4.6: Exemplo de arquivo de saída com nível de detalhamento Baixo

Já no nível de detalhamento Médio, além dos quatro índices anteriores, também são mostrados os valores intermediários de DMT e DNT para cada grupo da amostra original, GDMT e GDNT da amostra, UMT e WMT de cada grupo de cada algoritmo, além do GUMT e GWMT de cada algoritmo. A figura 4.7 mostra um exemplo hipotético de um arquivo de detalhamento médio.

```
Group 0 -> DMT 0, DNT 1426.0
Group 1 -> DMT 10, DNT 7120.0
Group 2 -> DMT 78, DNT 18460.0
GDMT: 88 GDNT: 27006
Algorithm #1:
Group 0 -> UMT 0
Group 1 -> UMT 3
Group 2 -> UMT 12
GUMT: 15
Group abaet -> WMT 0
Group abal -> WMT 2
Group abandon -> WMT 6
GWMT: 8
UI: 0.3014530471
OI: 0.0001566828
SW: 0.0005197586
ERRT: 0.7224725063
Algorithm #2:
Group 0 -> UMT 0
Group 1 -> UMT 0
Group 2 -> UMT 9
GUMT: 9
Group abaet -> WMT 0
Group abal -> WMT 4
Group abandon -> WMT 6
GWMT: 10
UI: 0.1905226632
OI: 0.0002680360
SW: 0.0014068458
ERRT: 0.5691374097
```

Figura 4.7: Exemplo de arquivo de saída com nível de detalhamento Médio

Por fim, o nível de detalhamento Alto mostra, além das informações do nível Médio, quais palavras formam cada grupo original e seus respectivos *stems*, os valores de UI, OI e SW para os diversos truncamentos calculados, o ponto de intersecção na linha de truncamento e os valores de OP e OT. A figura 4.8 mostra um exemplo pequeno

hipotético, onde algumas partes do arquivo (como os truncamentos de 4 a 12) foram intencionalmente retiradas para diminuir o tamanho da figura.

```
BEFORE: abacaxi
Group 0 -> DMT 0, DNT 1365.0

BEFORE: abafar
BEFORE: abafaram
Group 3 -> DMT 1, DNT 2729.0

GDMT: 8622 GDNT: 3719193


Algorithm #1:

AFTER: abacax
Group 0 -> UMT 0

AFTER: abaf
AFTER: abaf
Group 3 -> UMT 0

GUMT: 1607

Group abacax -> WMT 0

Group abaf -> WMT 0

Group abaix -> WMT 0

GWMT: 846


UI: 0.1863836697

UI1: 0.0017397356

UI2: 0.0146137787

UI3: 0.0272558571

OI: 0.0002274687

OI1: 0.2589300959

OI2: 0.0975996675

OI3: 0.0167340065

SW: 0.0012204326

SW1: 148.8330190985

SW2: 6.6786058158

SW3: 0.6139600188

Intersection Point between the Truncation Line and the Stemmed Line: (0.3412746699,0.0004165027)

OP: 0.1863838085 e OT: 0.3412749240

ERRT: 0.5461397699
```

Figura 4.8: Exemplo de arquivo de saída com nível de detalhamento Alto

5 EXPERIMENTOS E RESULTADOS

Neste capítulo, mostraremos os experimentos realizados, com a descrição da metodologia de como foram feitos, além dos resultados obtidos. Os *stemmers* usados neste capítulo são os descritos no capítulo 3 do presente trabalho.

Na seção 5.1, será avaliada a qualidade dos algoritmos de *stemming* sob a ótica do Método de Paice. Na seção 5.2, serão mostrados os experimentos relativos ao contexto de RI. Na seção 5.3, será feita a correlação entre a qualidade do *stemmer* e seu impacto na RI. Por fim, na seção 5.4, faremos uma análise consulta a consulta da RI, para verificar quais informações adicionais poderemos obter.

5.1 Qualidade dos Algoritmos de Stemming usando o Método de Paice

Conforme visto anteriormente, para computarmos as métricas de qualidade propostas por Paice, precisamos de grupos de palavras morfológica e semanticamente relacionadas. Para gerar esses grupos, obtivemos uma lista de 31986 palavras[4] e, após, selecionamos uma amostra de 2854 palavras. Os principais critérios para selecionar as palavras da amostra foram ter palavras com diferentes caracteres iniciais e grupos de tamanhos diferentes. A amostra foi dividida, manualmente, em 888 grupos, o que significa que cada grupo possui, em média, 3,21 palavras.

Após a confecção da amostra, nós a aplicamos para cada um dos *stemmers* a serem avaliados, a fim de obter os arquivos com os *stems* relativos às palavras originais. Em seguida, usamos esses arquivos como entrada para a ferramenta desenvolvida e mostrada no capítulo 4. A tabela 5.1 mostra os resultados obtidos das 4 métricas para todos os *stemmers*. Os melhores resultados para cada métrica encontram-se em negrito.

Em termos de UI, o melhor resultado foi obtido pelo Trunc3, já que ele raramente deixa de conflacionar palavras relacionadas. Por outro lado, Trunc3 também possui o maior valor de OI, o que é ruim. O melhor UI dentre os *stemmers* reais foi obtido pelo RSLP.

Em termos de OI, o melhor resultado ocorre, obviamente, quando nenhum *stemming* é aplicado (NoStem), porém essa abordagem também produz o maior UI. O melhor OI dentre os *stemmers* reais foi obtido pelo RSLP-S (o Stemmer-S também obteve o mesmo OI). O UI e o OI, sozinhos, não permitem a identificação dos melhores algoritmos.

---

[4] Disponíveis em http://snowball.tartarus.org/algorithms/portuguese/voc.txt. Acesso em 15 nov. 2009.

Tabela 5.1: Resultados da avaliação com o método de Paice

	UI	**OI**	**SW**	**ERRT**
Porter	0.3014530471	0.0001468286	0.0004870694	0.7159116889
RSLP	**0.1905226632**	0.0002680360	0.0014068458	**0.5691374097**
RSLP-S	0.9515289525	**0.0000004927**	0.0000005178	0.9959400023
Savoy	0.8863372767	0.0000123683	0.0000139544	1.1300856452
Stemmer-S	0.9580351334	**0.0000004927**	0.0000005143	1.0027066652
StemBR	0.2668618521	0.0003271616	0.0012259587	0.7650424144
Trunc3	**0.0304706137**	0.0157951554	0.5183733933	1.0000000000
Trunc4	0.1078941661	0.0029855960	0.0276715241	1.0000000000
Trunc5	0.2676209065	0.0006777271	0.0025324146	1.0000000000
Trunc6	0.4520711342	0.0001096287	0.0002425032	1.0000000000
Trunc7	0.6484493602	0.0000209403	0.0000322929	1.0000000000
Trunc8	0.8150075905	0.0000083761	0.0000102774	1.0000000000
NoStem	1.0000000000	**0.0000000000**	0.0000000000	1.0000000000

De acordo com o ERRT, os melhores *stemmers* são, em ordem, RSLP, Porter, StemBR e RSLP-S, com o quarto bastante atrás dos três primeiros. Também notamos que o RSLP e o StemBR são mais pesados que o Porter e o Savoy, de acordo com o SW, fato que também foi encontrado em Alvares et al (2005) e Orengo & Huyck (2001).

Além disso, podemos perceber que, apesar do RSLP ter um SW similar ao StemBR, o primeiro é melhor que o segundo tanto em UI quanto em OI. Em contrapartida, o RSLP é melhor que Porter e Savoy apenas em UI, visto que estes têm melhor OI.

Podemos perceber, também, que os *stemmers* leves, RSLP-S e Stemmer-S, têm resultados bem ruins de ERRT, isto é, muito próximos de 1. Isso ocorre porque o UI deles é bem alto, também perto de 1, já que eles têm poucas regras de remoção de afixos e, portanto, um número pequeno de palavras relacionadas ficam com o mesmo *stem*.

Por fim, vemos que o ERRT é exatamente 1 para todos os algoritmos de truncamento (visto que eles estão obviamente na linha de truncamento) e para o NoStem, pois ele pode ser visto como uma extensão do TruncN, com o *n* sendo o tamanho da maior palavra da amostra.

A figura 5.1 mostra uma interpretação gráfica do ERRT para os *stemmers* avaliados. Um bom *stemmer* deve estar abaixo da linha de truncamento e o mais longe possível dela. Dessa forma, o RSLP pode ser considerado o melhor *stemmer*, seguido do Porter e StemBR. Os *stemmers* leves RSLP-S, Stemmer-S e o de peso médio Savoy estão perto

ou acima da linha de truncamento e, portanto, podem ser considerados piores que os anteriores.

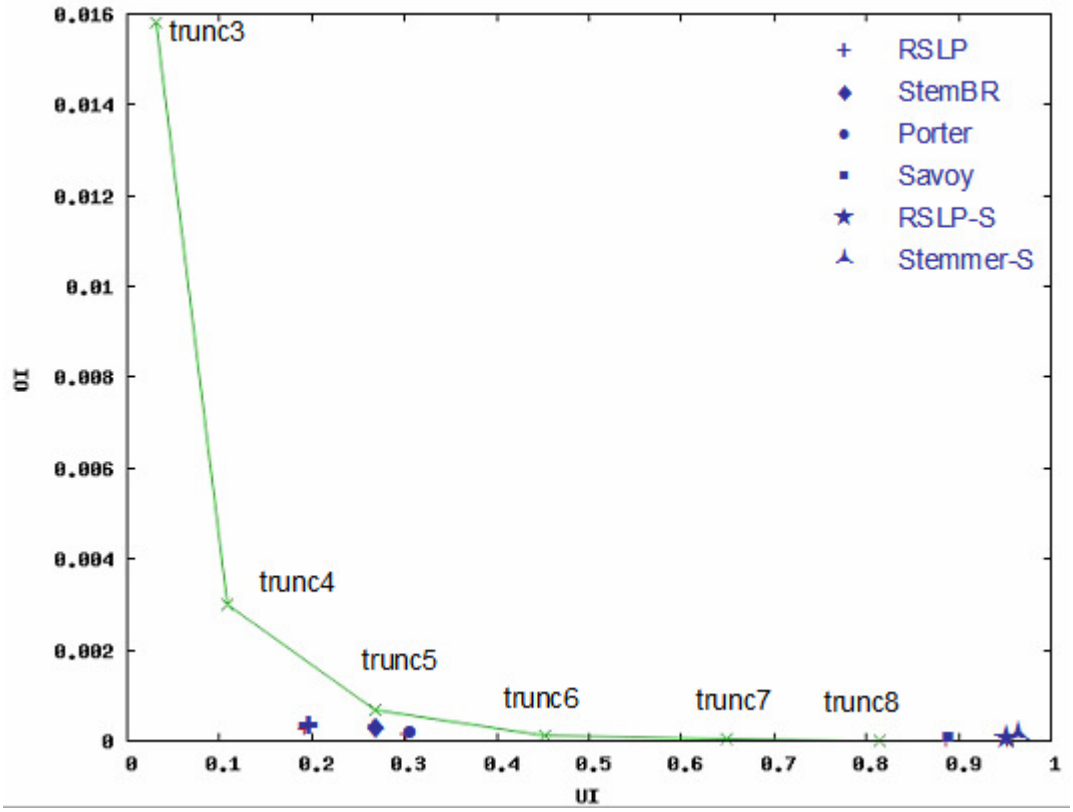

Figura 5.1: Gráfico do ERRT para os *stemmers* avaliados

5.2 Avaliação na Recuperação de Informações

Nesta seção, veremos os experimentos feitos com os *stemmers* em um sistema de RI. Na subseção 5.2.1, serão vistos os procedimentos adotados para a realização dos experimentos. Já na subseção 5.2.2, veremos os resultados que foram obtidos para os experimentos.

5.2.1 Procedimentos Utilizados

Para a realização dos testes, utilizamos uma coleção de dados da língua portuguesa das campanhas do *Cross-Language Evaluation Forum* (CLEF)[5] de 2005 e 2006. O CLEF é um projeto de avaliação e teste de sistemas de RI, focado nas línguas européias e financiado pela União Européia. Os dados em questão são de artigos do jornal "Folha de São Paulo", dos anos de 1994 e 1995. Na coleção também havia artigos do jornal "O Público", um jornal de Portugal, os quais não foram usados, visto que os *stemmers* que estamos testando tratam mais do português brasileiro. Um total de 98 tópicos de consulta foram usados nos experimentos.

Os documentos e tópicos utilizados nos testes estão no formato do *Text Retrieval Conference* (TREC), que é uma conferência patrocinada pelo departamento de defesa

[5] Disponível em http://www.clef-campaign.org. Acesso em 15 nov. 2009.

americano e pelo *National Institute of Standards and Technology* (NIST). As figuras 5.2 e 5.3 a seguir mostram, respectivamente, um exemplo de uma parte de um documento no formato TREC e um exemplo de uma parte de um arquivo de tópicos de consulta nesse formato.

```
<DOC>
<DOCNO>FSP940101-001</DOCNO>
<DOCID>FSP940101-001</DOCID>
<DATE>940101</DATE>
<TEXT>
Pesquisa Datafolha feita nas dez principais capitais do país, após um a
</TEXT>
</DOC>
<DOC>
<DOCNO>FSP940101-002</DOCNO>
<DOCID>FSP940101-002</DOCID>
<DATE>940101</DATE>
<TEXT>
Os quenianos dominaram a corrida de São Silvestre ontem. Simon Chemwoyc
</TEXT>
</DOC>
<DOC>
<DOCNO>FSP940101-003</DOCNO>
<DOCID>FSP940101-003</DOCID>
<DATE>940101</DATE>
<TEXT>
O ano que se encerrou não trouxe a tão desejada estabilização da econom
Há ângulos setoriais que merecem comemoração. O setor externo teve um d
Além disso, o governo fechou um acordo com os bancos estrangeiros e, co
No campo da abertura da economia, efetivou-se, em julho, a última rodad
O governo fechou ainda um acordo sobre as dívidas de Estados e municípi
Quanto às privatizações, não há muito o que comemorar em 93. É verdade
Esses avanços setoriais são, no entanto, ofuscados pela crise social e
É preciso ressaltar que a maior parte do pouco que há a comemorar em 93
É um papel que não pode se repetir em 94, pois a questão central -a inf
</TEXT>
</DOC>
```

Figura 5.2: Documento no formato TREC

```
<top>
<num> 251 </num>
<title> Medicina alternativa </title>
<desc> Encontrar documentos sobre tratamentos que empreguem medicina n
<narr> Documentos relevantes devem fornecer informação, específica ou
</top>

<top>
<num> 252 </num>
<title> Sistemas de reforma e pensões na Europa </title>
<desc> Encontrar documentos sobre os esquemas de pensões e benefícios
<narr> Documentos relevantes devem conter informação sobre os actuais
</top>

<top>
<num> 253 </num>
<title> Países com pena de morte </title>
<desc> Em quais países ainda é usada, ou pelo menos constitucionalment
<narr> Documentos que afirmem de forma explícita que a constituição de
</top>
```

Figura 5.3: Tópicos de consulta no formato TREC

Para indexar a coleção de dados, usamos o sistema de RI Zettair, da Search Engine Group (2006). Vamos mostrar, a seguir, quais os passos realizados para a execução dos testes. Eles foram executados em uma plataforma Windows.

Primeiramente, executamos o Zettair com a opção de indexar "-i", conforme exemplo da figura 5.4 abaixo.

```
zet -i -f rslp --stem none -t TREC < arquivos.txt
```

Figura 5.4: Comando Zettair para indexar a coleção de documentos

No comando acima, *zet* é o nome do programa a ser executado, *-i* indica opção de construção dos índices, *-f* indica o nome do arquivo de índices que será criado (no exemplo, será criado o arquivo *rslp*), *--stem none* indica que não será usada nenhuma forma de *stemming* que o Zettair possui nativa. Fizemos isso porque os *stemmers* do Zettair são específicos para a língua inglesa, e os *stemmers* que testaremos são da língua portuguesa. *-t TREC* indica que usaremos o índice no formato TREC. Por fim, *< arquivos.txt* indica que o índice *rslp* será criado a partir dos arquivos listados no arquivo de texto indicado. O conteúdo de *arquivos.txt* é simplesmente uma lista de nomes de arquivos que estejam no formato da figura 5.2. Esses arquivos já devem ter sido previamente processados pelo *stemmer* (no exemplo, o RSLP), antes de executar o comando do Zettair. Ao processar os arquivos, deve-se tomar o cuidado de fazer o *stemming* apenas das partes que estejam entre as *tags* <TEXT> e </TEXT>.

Após a execução desse primeiro comando, o qual cria o arquivo de índices, devemos executar um outro programa, *o zet_trec*, também do sistema Zettair, o qual recebe como entrada um arquivo de índices no formato TREC e produz como saída um arquivo que pode ser avaliado pelo programa *trec_eval*, o qual foi desenvolvido pelo NIST e será explicado mais adiante. Nessa parte, devemos executar um comando como o da figura 5.5.

```
zet_trec -okapi -f top-pt0506.txt rslp > rslp0506.out
```

Figura 5.5: Comando *zet_trec* que gera uma saída para o *trec_eval*

No comando acima, *zet_trec* é o nome do programa a ser executado, *-okapi* indica que usaremos a métrica Okapi BM25[6], *-f* indica o nome do arquivo de tópicos TREC que será usado (no exemplo, o arquivo *top-pt0506.txt*). Esse arquivo deve estar no formato da figura 5.3 e já deve ter sido previamente processado pelo *stemmer* a ser avaliado. Novamente, deve-se tomar o cuidado de só fazer o *stemming* do que estiver entre as *tags* <title> e </title>, <desc> e </desc>, e <narr> e </narr>. Ainda no comando acima, *rslp* é o arquivo de índices criado no comando da figura 5.4 e *rslp0506.out* foi o nome do arquivo escolhido para ser gravada a saída desse comando.

Por fim, devemos executar o programa *trec_eval*, desenvolvido pelo NIST, o qual avalia resultados no formato TREC e produz resultados como MAP e número de documentos relevantes retornados. O comando a ser executado é como o da figura 5.6 abaixo.

---

[6] Função de similaridade muito usada na avaliação de Sistemas de RI. Definição disponível em http://en.wikipedia.org/wiki/Probabilistic_relevance_model_(BM25). Acesso em: 15 nov. 2009.

```
trec_eval qrels_pt0506.txt rslp0506.out > trec_eval_rslp.txt
```

Figura 5.6: Comando trec_eval que gera os resultados desejados

No comando acima, *trec_eval* é o nome do programa a ser executado, *qrels_pt0506.txt* é o arquivo de avaliação dos resultados, conforme figura 5.8, *rslp0506.out* é o arquivo gerado como saída no comando da figura 5.5 e *trec_eval_rslp.txt* é o arquivo com os resultados, exemplificado na figura 5.7.

```
Queryid (Num):       98
Total number of documents over all queries
    Retrieved:    96322
    Relevant:      2159
    Rel_ret:       1684
Interpolated Recall - Precision Averages:
    at 0.00       0.5706
    at 0.10       0.4942
    at 0.20       0.4214
    at 0.30       0.3865
    at 0.40       0.3453
    at 0.50       0.3043
    at 0.60       0.2326
    at 0.70       0.1991
    at 0.80       0.1616
    at 0.90       0.1280
    at 1.00       0.0944
Average precision (non-interpolated) for all rel docs(averaged over queries)
                  0.2898
Precision:
  At    5 docs:   0.3531
  At   10 docs:   0.2990
  At   15 docs:   0.2673
  At   20 docs:   0.2418
  At   30 docs:   0.2129
  At  100 docs:   0.1085
  At  200 docs:   0.0658
  At  500 docs:   0.0312
  At 1000 docs:   0.0172
R-Precision (precision after R (= num_rel for a query) docs retrieved):
    Exact:        0.2726
```

Figura 5.7: Arquivo de saída do programa *trec_eval*

```
251 0 FSP940128-147 0
251 0 FSP940128-148 0
251 0 FSP940130-099 1
251 0 FSP940130-100 0
251 0 FSP940131-030 0
251 0 FSP940131-079 0
251 0 FSP940204-141 0
251 0 FSP940205-099 0
251 0 FSP940206-094 1
251 0 FSP940221-132 0
251 0 FSP940227-193 0
251 0 FSP940228-080 0
251 0 FSP940304-100 0
251 0 FSP940317-084 0
251 0 FSP940318-123 0
251 0 FSP940319-099 0
251 0 FSP940319-100 0
251 0 FSP940321-033 0
251 0 FSP940329-082 0
251 0 FSP940330-099 1
251 0 FSP940330-110 1
251 0 FSP940404-083 0
251 0 FSP940408-009 0
251 0 FSP940421-065 0
251 0 FSP940424-145 0
251 0 FSP940515-106 1
251 0 FSP940517-050 0
251 0 FSP940529-079 1
251 0 FSP940529-111 1
251 0 FSP940530-021 0
251 0 FSP940531-028 0
```

Figura 5.8: Exemplo de arquivo de avaliação no formato TREC

No arquivo da figura 5.8 acima, a primeira coluna representa o número do tópico ou consulta. A segunda coluna não é utilizada, ficando fixa em 0. A terceira coluna representa o código do documento, que fica entre as *tags* <DOCNO> e </DOCNO> do arquivo da figura 5.2. Por fim, a quarta coluna é 0 se o documento não for relevante para a consulta ou 1, caso contrário.

5.2.2 Resultados Obtidos

Nesta subseção, vamos mostrar os resultados obtidos com os experimentos. Primeiramente, a figura 5.9 mostra um gráfico com a curva da precisão pela revocação para todos os *stemmers*, exceto os de truncamento. Já a figura 5.10 mostra o mesmo gráfico, porém mostrando somente os de truncamento. Por fim, a tabela 5.2 mostra os resultados da precisão média (MAP), o número de documentos relevantes retornados, o número de termos indexados distintos e o tempo de execução de cada *stemmer*, estando os melhores resultados de cada item em negrito.

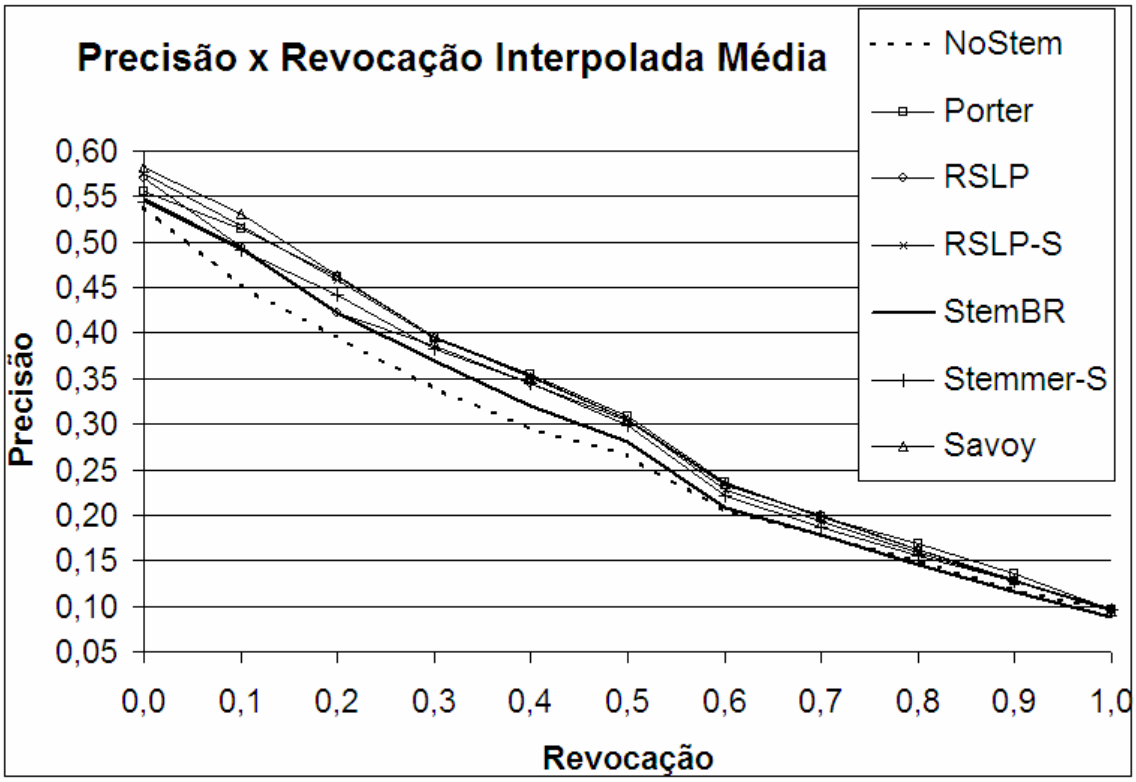

Figura 5.9: Curvas de precisão x revocação dos *stemmers* (exceto truncamento)

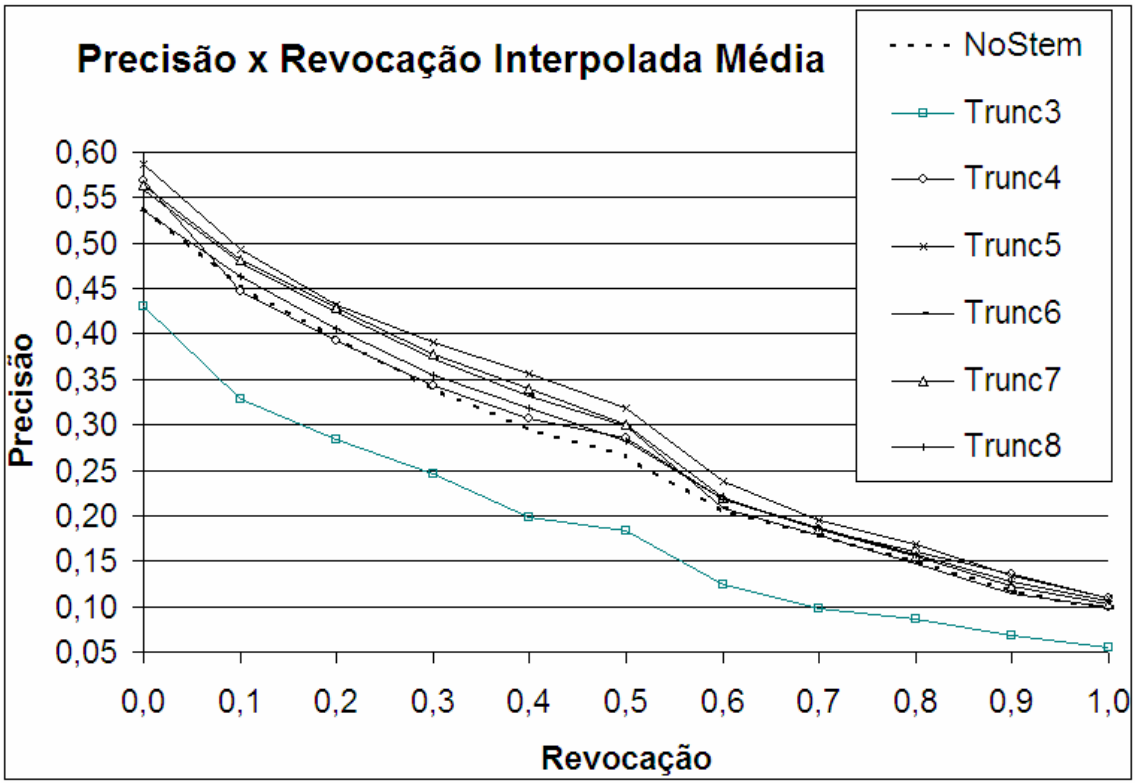

Figura 5.10: Curvas de precisão x revocação dos *stemmers* de truncamento

Tabela 5.2: Resultados dos experimentos de RI

	Termos Indexados Distintos	**Doc. Relevantes Retornados**	**Tempo de Execução**	**MAP**
Porter	250864 (-26.51%)	1680	00:57:35	0.2981
RSLP	237490 (-30.43%)	1684	00:06:43	0.2898
RSLP-S	313737 (-8.09%)	1695	00:06:34	0.2938
Savoy	306836 (-10.11%)	**1725**	00:01:15	**0.2984**
Stemmer-S	316753 (-7.21%)	1664	**00:01:08**	0.2736
StemBR	233536 (-31.59%)	1668	04:24:37	0.2829
Trunc3	**72750 (-77.36%)**	1355	00:01:14	0.1784
Trunc4	109093 (-68.04%)	1685	00:01:14	0.2706
Trunc5	163068 (-52.23%)	1696	00:01:14	0.2957
Trunc6	212723 (-37.68%)	1540	00:01:14	0.2751
Trunc7	250782 (-26.53%)	1512	00:01:14	0.2812
Trunc8	280803 (-17.74%)	1465	00:01:14	0.2718
NoStem	341362	1367	-	0.2587

A segunda coluna da tabela 5.2 mostra o número de termos indexados distintos gerados após o Zettair indexar a coleção. O Trunc3 tem o melhor desempenho nesse aspecto, pois um dos objetivos do *stemming* é reduzir o tamanho do arquivo de índices e é ele que gera o menor número de índices. Entre os *stemmers* reais, a maior redução foi obtida pelo StemBR.

A terceira coluna da tabela 5.2 mostra o número total de documentos relevantes retornados. Com a exceção do Trunc3, todas as estratégias de *stemming* avaliadas permitiram que um número maior de documentos fossem retornados, comparando com o caso em que nenhuma forma de *stemming* é aplicada. O *stemmer* que possibilitou o retorno do maior número de documentos relevantes foi o Savoy, seguido do RSLP-S.

A quarta coluna da tabela 5.2 mostra o tempo de execução, isto é, quanto tempo os *stemmers* levaram para processar todos os arquivos relativos aos documentos da "Folha de São Paulo". Os resultados mostram que o StemBR foi, de longe, o algoritmo mais lento, levando mais de quatro horas para completar o processo de stemming, enquanto o Porter levou aproximadamente uma hora e o RSLP apenas seis minutos. O Stemmer-S e os TruncN, que são bastante simples, demoraram aproximadamente apenas um minuto. O stemmer real mais rápido foi o Savoy, o qual também levou somente cerca de um minuto para processar os arquivos. Outra coisa que também podemos notar é que o RSLP e o RSLP-S não tiveram uma mudança significativa em velocidade, o que mostra que o algoritmo não executa muito mais rápido se ele executar apenas o passo de redução do plural, ignorando todos os restantes.

Por fim, a quinta coluna da tabela 5.2 mostra os resultados para o MAP. Podemos notar, portanto, que o Savoy foi o *stemmer* que obteve o melhor resultado quanto ao aprimoramento da eficácia de RI, enquanto que o Trunc3 foi o pior nesse sentido. Também foi realizado um teste-T pareado, que tem por objetivo verificar se existe uma

diferença estatística significativa entre duas médias obtidas sobre a mesma amostra. No nosso caso, iremos testar se as diferenças de MAP entre os *stemmers* são estatisticamente significantes. Foi utilizado 0,05 como limiar, ou seja, caso o resultado do nosso teste-T entre os MAPs de dois *stemmers* seja menor que 0,05 significa que a diferença entre eles é estatisticamente significante. Caso contrário, a diferença não é estatisticamente significante, tendo sido, possivelmente, fruto do acaso. A tabela 5.3 mostra o resultado do teste para os *stemmers*. Os casos em que o valor obtido é menor que 0,05 estão destacados: em negrito, caso o *stemmer* da linha tenha sido pior em MAP que o *stemmer* da coluna; em itálico, sublinhado e em outra cor, caso o *stemmer* da linha tenha sido melhor em MAP que o da coluna.

	NoStem	Porter	RSLP	RSLP-S	StemBR	Stemmer-S	Trunc3	Trunc4	Trunc5	Trunc6	Trunc7	Trunc8	Savoy
NoStem	x	0,009311	0,025701	0,000432	0,261365	0,004785	0,000052	0,546066	0,034720	0,194127	0,039519	0,154696	0,003379
Porter		x	0,380328	0,728264	0,015413	0,230968	0,000000	0,153210	0,850063	0,107316	0,230343	0,072879	0,974342
RSLP			x	0,762698	0,049372	0,575453	0,000000	0,312910	0,690322	0,337500	0,528368	0,189542	0,342111
RSLP-S				x	0,129654	0,006392	0,000000	0,288958	0,905604	0,074735	0,124347	0,021755	0,659346
StemBR					x	0,484672	0,000000	0,873940	0,146911	0,942215	0,622839	0,875233	0,012759
Stemmer-S						x	0,000001	0,577222	0,436723	0,472143	0,840906	0,227480	0,177160
Trunc3							x	0,000000	0,000000	0,000000	0,000001	0,000004	0,000000
Trunc4								x	0,061330	0,823403	0,623384	0,967257	0,123024
Trunc5									x	0,133017	0,354092	0,129730	0,830262
Trunc6										x	0,381542	0,722121	0,106224
Trunc7											x	0,107912	0,197374
Trunc8												x	0,046006
Savoy													x

Tabela 5.3: Resultados dos testes-T

Analisando o resultado dos testes-T, vemos que, apesar de todos os *stemmers* (exceto Trunc3) terem tido um MAP melhor que o NoStem, somente Savoy, Porter, RSLP, RSLP-S, StemmerS, Trunc5 e Trunc7 mostraram uma superioridade estatisticamente significante. Vemos também que o Trunc3 foi pior que todos os outros de forma significativa, inclusive em relação ao NoStem. Além disso, vemos que o StemBR foi significativamente pior que o Savoy, o Porter e o RSLP. Outrossim, Trunc8 também mostrou uma inferioridade estatisticamente significante em relação ao Savoy. Por fim, vemos que o RSLP-S foi significativamente melhor que o Stemmer-S e o Trunc8.

5.3 Correlação entre Qualidade do Stemmer e seu Impacto na RI

Nesta seção, analisaremos as correlações entre a qualidade do *stemmer* (calculada pelo método de Paice) e a qualidade da RI (medida pelo MAP, número de documentos relevantes retornados e número de termos indexados distintos). Essas correlações são mostradas na tabela 5.4 abaixo.

Tabela 5.4: Correlação entre qualidade do *stemmer* e seu impacto em RI

	MAP	Doc. Relevantes Retornados	Termos Indexados Distintos
Overstemming Index	-0,92	-0,50	-0,71
Understemming Index	0,34	-0,08	0,88
Stemming Weight	-0,92	-0,26	-0,64
ERRT	-0,17	-0,53	0,04

A correlação entre duas variáveis está sempre entre -1 e 1, com o valor 0 indicando que as duas variáveis não estão correlacionadas. Se a correlação for positiva, significa que, quando uma variável cresce, a outra também aumenta seu valor. Já uma correlação negativa indica que, quando uma variável cresce, a outra decresce. Quanto mais próxima de 1, em valor absoluto, for a correlação, mais forte ela é. Ao contrário, quanto mais próxima de 0, mais fraca é a correlação.

Intuitivamente, esperaríamos uma correlação negativa forte entre os erros feitos pelo *stemmer* e suas medidas de qualidade em RI, como o MAP e número de documentos relevantes retornados (segunda e terceira coluna da tabela 5.4). Entretanto, essa suposição nem sempre é correta. A correlação entre MAP e ERRT, por exemplo, embora negativa, é fraca, somente -0,17. Analisando os resultados dos experimentos das seções 5.1 e 5.2, observamos que um *stemmer* que tenha um ERRT mais baixo não necessariamente é melhor para RI que um outro que tenha um ERRT mais alto. Isso é especialmente verdadeiro nos casos de *stemmers* mais leves, como o RSLP-S, que obteve uma performance bastante boa em RI, mas teve o ERRT bem próximo a 1. Além disso, o Savoy e o Stemmer-S, que obtiveram os maiores valores de ERRT, mostraram melhoras estatisticamente significantes em relação ao NoStem.

O ERRT computa a distância em relação à linha de truncamento, assumindo que um algoritmo de truncamento seria ruim e, portanto, deveria ser evitado. No entanto, essa afirmação acabou se revelando falsa, visto que os *stemmers* Trunc5 e Trunc7 obtiveram bons resultados em RI. Não obstante, o ERRT foi capaz de distinguir a qualidade de *stemmers* de peso similar. Em relação aos *stemmers* leves, o RSLP-S foi

significativamente melhor em MAP que o Stemmer-S e teve um ERRT mais baixo também. Já em relação aos *stemmers* mais pesados, o StemBR teve um ERRT maior que o Porter e o RSLP, e acabou também sendo significativamente pior em termos de MAP. O RSLP, todavia, foi bem melhor que o Porter em ERRT, apesar de eles não terem tido diferenças significativas em termos de MAP.

Em relação ao MAP, segunda coluna da tabela 5.4, as correlações negativas fortes ocorreram com o OI e o SW. Esse fato indica que, para propósitos de RI, os erros de *overstemming* são mais sérios que os de *understemming*. Além disso, a tabela 5.4 mostra que o UI é positivamente correlacionado com o MAP, apesar de essa correlação ser fraca. Essas conclusões corroboram estudos anteriores que defendem o uso de alternativas leves de *stemming* para fins de RI.

Também seria esperado que o número de documentos relevantes retornados, terceira coluna da tabela 5.4, decrescesse na presença de erros de *understemming*. Apesar disso, não foi encontrada uma correlação entre essas duas medidas, visto que, apesar de negativa, ela foi muito fraca.

Por fim, podemos analisar a quarta coluna, que mostra a correlação com o número de termos indexados distintos obtidos. Nesse caso, os resultados ocorreram como o esperado. A correlação com o UI foi fortemente positiva, enquanto que a com o OI e o SW foi negativa, o que mostra que quanto mais pesado for o *stemmer*, mais palavras ficarão com o mesmo *stem* e, portanto, menos termos indexados distintos existirão. Já a correlação do número de termos indexados com o ERRT foi bastante próxima de zero, o que mostra que essas duas medidas não estão relacionadas.

5.4 Análise Consulta a Consulta

Nesta seção, faremos a análise de alguns dados obtidos através do exame de cada um dos 98 tópicos de consulta utilizados na seção 5.2, ao invés de olhar para a média geral, como foi feito lá. A tabela 5.5 mostra, para cada *stemmer*, em quantas consultas ele foi melhor que o NoStem, em quantas foi pior e em quantas obteve o mesmo desempenho. Foi considerado que um *stemmer* obteve o mesmo desempenho que o NoStem em uma determinada consulta se a diferença entre os seus AvPs na consulta foi de até 5%. Os melhores resultados aparecem em negrito.

Tabela 5.5: Resultados por consulta em relação ao NoStem

Stemmer	Melhor que NoStem	Pior que NoStem	Diferença Melhor-Pior	Igual a NoStem
Porter	48,98%	27,55%	21,43%	23,47%
RSLP	**54,08%**	25,51%	**28,57%**	20,41%
RSLP-S	42,86%	17,34%	25,52%	39,80%
Savoy	45,92%	23,47%	22,45%	30,61%
Stemmer-S	36,73%	19,39%	17,34%	43,88%
StemBR	44,90%	33,67%	11,23%	21,43%
Trunc3	27,55%	66,33%	-38,78%	6,12%
Trunc4	43,88%	39,79%	4,09%	16,33%
Trunc5	50,00%	25,51%	24,49%	24,49%
Trunc6	37,76%	25,51%	12,25%	36,73%
Trunc7	33,67%	**15,31%**	18,36%	51,02%
Trunc8	22,45%	17,35%	5,10%	60,20%

Pela tabela acima, podemos perceber que o RSLP foi o *stemmer* que obteve o melhor desempenho considerando a percentagem de consultas em que foi melhor e pior que a abordagem sem *stemming*, enquanto que o Trunc3 foi o pior. No entanto, considerando o MAP, conforme visto na subseção 5.2.2, o Savoy teve a melhor média. Isso pode ser explicado pelo fato de que o Savoy, em muitas consultas em que ambos (Savoy e RSLP) foram bem ou em que ambos foram mal, foi melhor que o RSLP. Além disso, também encontramos na tabela um dado que vem ao encontro do resultado do teste-T da subseção 5.2.2: os *stemmers* que se mostraram significativamente melhores que o NoStem no teste-T são também os que tiveram uma diferença entre a segunda e a terceira coluna da tabela 5.5 de no mínimo 17%.

Outro dado interessante obtido a partir da análise consulta a consulta é o número de consultas em que alguma forma de *stemming* foi benéfica para o aprimoramento da precisão média. Dos 98 tópicos de consulta utilizados nos experimentos, 76 tiveram algum *stemmer* com um AvP mais de 5% maior que o obtido pelo NoStem, enquanto que em apenas 22 tópicos nenhum *stemmer* obteve uma melhora significativa em relação ao NoStem. Isso significa que em 77,55% das consultas o uso de alguma forma de *stemming* foi benéfica.

Por fim, vemos também que o Trunc3 foi o responsável pela maior queda de AvP para uma consulta individual comparando com o NoStem (de 0,5135 para 0,0012), enquanto que o Trunc5 teve o maior aumento de AvP individual (de 0,0363 para 0,8833). Esse último resultado específico pode ser explicado pelo fato de que a consulta em questão pedia documentos sobre a guerra civil no "Iémene" e a maioria dos documentos relevantes tratavam do país nas suas variantes ortográficas "Iémen" ou "Iêmen" e, portanto, os *stemmers* que faziam esse tipo de tratamento, como Porter, RSLP, StemBR, Savoy, Trunc4 e Trunc5 tiveram ótimos resultados para essa consulta.

6 CONCLUSÃO

O presente trabalho avaliou os diversos *stemmers* para a língua portuguesa encontrados na literatura, em termos da sua qualidade e correção e em termos do seu impacto na recuperação de informações. O objetivo era descobrir se a qualidade de um algoritmo de *stemming*, particularmente a qualidade medida pelo método de Paice, se traduziria em melhora da RI. Nossa conclusão foi que o *stemmer* mais correto, segundo Paice, não foi o que atingiu a maior melhora na RI.

Foi desenvolvida para este trabalho uma ferramenta que implementa o método de Paice, a qual foi disponibilizada na Internet. Com essa implementação e uma amostra de palavras, torna-se muito simples e rápido analisar *stemmers* de qualquer idioma segundo o referido método. A ferramenta calcula todas as quatro métricas definidas pelo método: UI, OI, SW e ERRT. Dentre os resultados que nós vimos que uma pessoa pode descobrir somente com o método de Paice é que, se ela não possui qualquer informação prévia sobre os *stemmers* a serem testados, o método corretamente mostra quais são os *stemmers* leves e quais são os pesados e, por conseguinte, quais trarão uma grande diminuição no número de termos indexados distintos. Além disso, o método de Paice também foi capaz de mostrar quais eram os melhores *stemmers* quando eles possuíam pesos parecidos, mas diferenças de qualidade estatisticamente significantes.

Durante o desenvolvimento do presente trabalho, foi escrito e submetido um artigo sobre o assunto aqui tratado para a *9th International Conference on the Computational Processing of Portuguese* (PROPOR 2010). O artigo foi aceito e será apresentado em abril de 2010. Seu título é "*Assessing the Impact of Stemming Accuracy on Information Retrieval*" e seus autores são o autor, a orientadora e o co-orientador do presente trabalho.

Em relação a trabalhos futuros, poder-se-ia fazer um estudo para verificar se as mesmas correlações encontradas neste trabalho se aplicariam a outras línguas além do português. Além disso, um estudo mais aprofundado em relação às consultas individuais poderia ser feito, para verificar em quais casos o *stemming* ainda não está sendo benéfico, a fim de melhorá-lo. Finalmente, trabalhos futuros poderiam usar a nossa ferramenta do método de Paice e os resultados obtidos aqui para aprimorar algoritmos de *stemming* existentes.

REFERÊNCIAS

ALVARES, R. V.; GARCIA, A. C. B.; FERRAZ, I. **STEMBR: A Stemming Algorithm for the Brazilian Portuguese Language**. 2005, Springer. p. 693-701.

ALVARES, R. V. **Investigação do Processo de *Stemming* na Língua Portuguesa**. 2005. 83 f. Dissertação (Mestrado em Computação) – UFF, Niterói.

COELHO, A. R. **Stemming para a língua portuguesa: estudo, análise e melhoria do algoritmo RSLP**. 2007. 70 f. Trabalho de Graduação (Bacharelado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

FRAKES, W. B.; BAEZA-YATES, R. **Information Retrieval:** Data Structures & Algorithms. Upper Saddle River, NJ, USA: Prentice Hall, 1992.

FREUND, G. E. **Análise Estrutural para Aumentar a Eficiência de Pesquisas "Online"**. Ciência da Informação, Brasília, v. 11, n. 1, p. 19-26, 1982. Disponível em: <http://revista.ibict.br/index.php/ciinf/article/viewFile/1505/1123>. Acesso em: 15 nov. 2009.

HARMAN, D. **How Effective is Suffixing?** In: Journal of the American Society for Information Science, New York, v. 42, n. 1, p. 7-15, 1991.

HULL, D. A. **Stemming Algorithms: A Case Study for Detailed Evaluation**. In: Journal of the American Society for Information Science, New York, v. 47, n. 1, p. 70-84, 1996.

KRAAIJ, W.; POHLMANN, R. **Evaluation of a Dutch Stemming Algorithm**. In: The New Review of Document & Text Management, T. G. Publishing, London, UK, p. 25-43, 1995.

KROVETZ, R. **Viewing Morphology as an Inference Process**. In: 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. 1993. p. 191-202.

LOVINS, J. B. **Development of a Stemming Algorithm**. Mechanical Translation and Computation Linguistics, 1968, v. 11, p. 22-31.

ORENGO, V. M.; HUYCK, C. R. **A Stemming Algorithm for the Portuguese Languague**. In: 8th International Symposium on String Processing and Information Retrieval (SPIRE). 2001, Laguna de San Raphael, Chile, p. 183-193.

ORENGO, V. M.; BURIOL, L. S.; COELHO, A. R. **A Study on the Use of Stemming for Monolingual Ad-Hoc Portuguese Information Retrieval**. In: CLEF 2006, Alicante, Berlin Heidelberg: Springer-Verlag, 2007, v. 4730 p. 91-98.

PAICE, C. D. **Another Stemmer**. SIGIR Forum, 1990, v. 24, p. 56-61.

PAICE, C. D. **An Evaluation Method for Stemming Algorithms**. In: 17th ACM SIGIR Conference on Research and Development in Information Retrieval, W. B. Croft e C. J. Van Rijsbergen, Editors. 1994, ACM: Dublin, Ireland, p. 42-50.

PORTER, M. F. **An Algorithm for Suffix Stripping**. Program, 1980, v. 14, n. 3, p. 130-137.

PORTER, M. F. **Portuguese stemming algorithm**. 2007. Disponível em <http://snowball.tartarus.org/algorithms/portuguese/stemmer.html>. Acesso em 15 nov. 2009.

SAVOY, J. **Light stemming approaches for the French, Portuguese, German and Hungarian languages**. In: SAC '06: Proceedings of the 2006 ACM symposium on applied computing, p. 1031-1035, New York, NY, USA, 2006, ACM.

SEARCH ENGINE GROUP. **The Zettair Search Engine**. Melbourne, Australia: RMIT University, 2006. Disponível em: <http://www.seg.rmit.edu.au/zettair>. Acesso em: 15 nov. 2009.