PODes
Revista Eletrônica Pesquisa Operacional para o Desenvolvimento

Rio de Janeiro, v.5, n.1, p. 84-99, janeiro a abril de 2013

ANALISE COMPARATIVA DE ALGORITMOS DE REDUÇÃO DE RADICAIS E SUA IMPORTÂNCIA PARA A MINERAÇÃO DE TEXTO

Bruno Missi Xavier
Dataci - Companhia de Tecnologia da Informação de Cachoeiro de Itapemirim-ES
bmissix@gmail.com

Alcione Dias da Silva
Dataci - Companhia de Tecnologia da Informação de Cachoeiro de Itapemirim-ES
diasalcione@gmail.com

Geórgia Regina Rodrigues Gomes
UCAM – Universidade Candido Mendes
georgia@ucam-campos.br

Resumo

O processo de radicalização de palavras desempenha um papel fundamental para a mineração de texto. Definir qual algoritmo melhor se adapta a instância de dados utilizada, influencia diretamente na qualidade dos resultados dos processos de mineração. Este trabalho tem por objetivo apresentar uma comparação entre os algoritmos mais utilizados para radicalização da língua portuguesa, além de classificar os algoritmos quanto ao peso na tarefa de eliminar sufixos e apresentar qual o algoritmo melhor se adapta a base de dados confeccionada. Para a avaliação dos radicalizadores foi utilizado o método de Paice (1994), aplicado a grupos conceituais em uma amostra de 731 palavras. Os resultados apresentados demonstram uma boa qualidade dos radicalizadores e são conclusivos para a seleção de um *stemmer* que melhor se adéque à amostra. O atual estudo contribui para o entendimento do peso dos radicalizadores, e ainda para determinar o algoritmo que melhor se aplica a instância de dados selecionada.

Palavras-Chaves: Mineração de Texto, Radicalização, Paice.

Abstract

The word radicalization process plays a fundamental role in text mining. To define which algorithm best fits with the data instances that was used, directly influences the quality of the results of the mining processes. This paper aims to present a comparison among the most used algorithms to the radicalization of the Portuguese language, in addition to classify the weight on the task of removing suffixes and present the algorithm which best fits the database prepared. To evaluate the radicalization, Paice's method (1994) was used and applied to conceptual groups in a sample of 731 words. The presented results show a good quality of radicalization and are conclusive to select a stemmer that best fits the sample. The current study contributes to the understanding of the level of aggressiveness of radicalization and also to determine the algorithm that best fits the selected data instance.

Keywords: Text Mining, Stemming, Paice.

Versão inicial submetida em 24/04/2011. Versão final recebida em 06/08/2012.

1. Introdução

Com o crescimento e popularização da web e em virtude da disponibilização de grandes volumes de informação textual na internet, técnicas de mineração de texto vem sendo abordadas em estudos recentes. Em 2001, Chen (2001) relatou que 80% das informações disponíveis na web estão em formato textual. Branski (2004) afirmou existir cerca de 2,5 bilhões de documentos textuais disponíveis na internet com uma taxa de crescimento de 7,5 milhões ao dia. Segundo Royal Pingdom (2012), 300 milhões de novos sites foram publicados na web no ando de 2011, totalizando 555 milhões de sites disponíveis para consulta. Mineração da informação é a área de pesquisa de sistemas de informação, utilizada para navegar, organizar e descobrir informações em documentos textuais, podendo ainda ser considerada uma expansão da mineração de dados (ARANHA; PASSOS, 2006).

Algoritmos de radicalização, ou como conhecido do inglês, *Stemmer*, tem um papel fundamental na descoberta do conhecimento através de informação não estruturada. A principal função deste processo é permitir o mapeamento de palavras que sejam semântica e morfologicamente relacionadas, desta forma, podendo agrupar maior quantidade de palavras que tenham o mesmo sentido através de seu radical. Um exemplo pode ser analisado observando o conjunto de palavras: {prática, praticada, praticados, praticando, praticante, praticar, praticaram, praticidade, prático} onde apesar de terem características diferentes, reservam um mesmo radical, PRATIC. A radicalização do termo permite a utilização de vocábulos primitivos anteriores às variações, como plurais e inflexões verbais (PORTER, 1980).

Existem várias maneiras de avaliar a qualidade dos algoritmos de radicalização. Paice (1994) propõe um método de avaliação das particularidades dos algoritmos de radicalização levando em consideração quatro medidas: *Overstemming Index* (OI), *Understemming Index* (UI), *Stemming Weight* (SW) e *Error Rate Relative to Truncation* (ERRT), que aplicadas a amostras de palavras não repetidas, separadas por grupos semânticos e morfologicamente relacionados.

Os algoritmos de radicalização analisados neste trabalho foram, PORTER (PORTER, 2007), RSLP (ORENGO e HUYCK, 2001) e SAVOY (SAVOY, 2006), utilizando suas implementações para a língua portuguesa.

Este trabalho tem o objetivo de fazer um estudo comparativo entre os algoritmos de radicalização, sua classificação quanto ao peso para executar tarefas de remoção de sufixos do conjunto de palavras, e a determinação de qual algoritmo se adapta melhor a amostra confeccionada para o experimento realizado. Este estudo se torna relevante a medida que ele provoca maior compreensão acerca do comportamento dos *stemmers* e promove uma comparação entre os algoritmos.

2. Revisão da Literatura

2.1. Mineração de texto

A Mineração de Textos (MT) surge no contexto do processamento de grandes volumes de documentos com o objetivo de extrair informações significativas para a formação do conhecimento. Frawley, Piatetsky-shapiro e Matheus (1992) definem MT como o processo de extração não trivial de informação implícita, previamente desconhecida e potencialmente útil.

Mineração de Textos é um conjunto de técnicas com o objetivo de extrair de textos não estruturados ou semi-estruturados, informação inovadora e até então desconhecida. O processo de MT pode ser dividido minimamente em três etapas: preparação dos dados textuais, processamento do texto e avaliação dos resultados (GOMES, 2006).

A etapa de preparação dos dados textuais envolve desde a seleção do conteúdo a ser processado, redução dimensional do texto até agrupamentos morfológicos e sintáticos dos termos presentes no texto. Para eliminar palavras irrelevantes ao entendimento do texto, aplica-

se o processo de remoção de *Stop Words*, baseado em uma lista pré-definida de artigos, pronomes, advérbios etc. O processo de radicalização também é aplicado nesta etapa com o intuito de reduzir as variações sintáticas das palavras (GOMES, 2006).

A etapa de Mineração do Textos é responsável por estruturar a informação até então não estruturada. Vários processos podem ser aplicados nesta fase: indexação, clusterização, sumarização, classificação/categorização, extração da informação, entre outros (BARION; LAGO, 2008).

Gomes (2009) descreve o pós-processamento como o momento de avaliação do conhecimento descoberto na fase anterior. Esta etapa deve ser o mais intuitiva e visual possível afim de transmitir ao usuário, maior clareza para sua avaliação. Alem da avaliação supervisionada, o método *Precision and Recall* pode ser aplicado para obter uma medida qualitativa do resultado deste processamento (GOMES, 2006). A medida *Precision* indica o percentual de documentos/respostas relevantes recuperadas no processamento. *Recall* indica o percentual de todas de documentos/respostas relevantes recuperados em relação a todo o conjunto de documentos relevantes previstos (CORRÊA, 2003).

2.2. *Porter Stemmer*

Porter (1980) descreve um método para extração de radicais para a língua inglesa. Segundo o autor, a vantagem é sua construção razoavelmente simples e resultados com tempos computacionais interessantes. Este trabalho foi posteriormente adaptado pelo autor para a língua portuguesa (PORTER, 2007). O algoritmo foi construído originalmente, na linguagem BCPL , definidos em 5 passos:

- Passo 1: remoção de sufixos comuns (por exemplo: eza, ismos, ável, ível, oso, ações, mente, ânsia...);
- 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, arão, 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”.

Porter define uma série de regras na construção do algoritmo para identificação de vogais e consoantes na ultima posição da palavra.

2.3. Removedor de Sufixos da Língua Portuguesa (RSLP)

O RSLP, proposto por Orengo e Huyck (2001), foi projetado especialmente para a língua portuguesa, com objetivo de ser simultaneamente simples e eficaz melhorando a precisão de consultas em grupos de documentos. O diferencial deste algoritmo é que além de analisar diversas regras específicas do idioma português, ainda conta com um dicionário de exceções. Este radicalizador é composto por oito etapas, cada etapa contendo um conjunto de regras, onde, apenas uma regra em cada etapa pode ser aplicada.

- Passo 1: remoção do plural das palavras (normalmente “s”). Possui 11 regras definidas;
- Passo 2: transformação do gênero da palavra de feminino para masculino. São aplicadas 15 regras;
- Passo 3: redução adverbial, possui somente um sufixo (“mente”);
- Passo 4: reduz o grau das palavras de aumentativo, superlativo ou diminutivo para normal. Aplicação de 23 regras;
- Passo 5: redução nominal, originalmente era definido 61 sufixos para substantivos e

adjetivos, que foram expandidos para 84. Caso a palavra seja reduzida neste passo, os passos 6 e 7 não são executados;

- Passo 6: redução verbal, são definidas 101 regras para mapear os verbos regulares em suas mais de 50 formas. Após a execução deste passo, o termo é reduzida a sua raiz, indo direto para a execução do passo 8;
- Passo 7: remoção da última vogal (“a”, “e” ou “o”) de palavras que não foram reduzidas pelos passos 5 e 6;
- Passo 8: substitui todas as letras acentuadas por seus equivalentes sem acentos.

A Figura 1 demonstra o fluxo de execução dos passos descritos por Orengo, Huyck (2001).

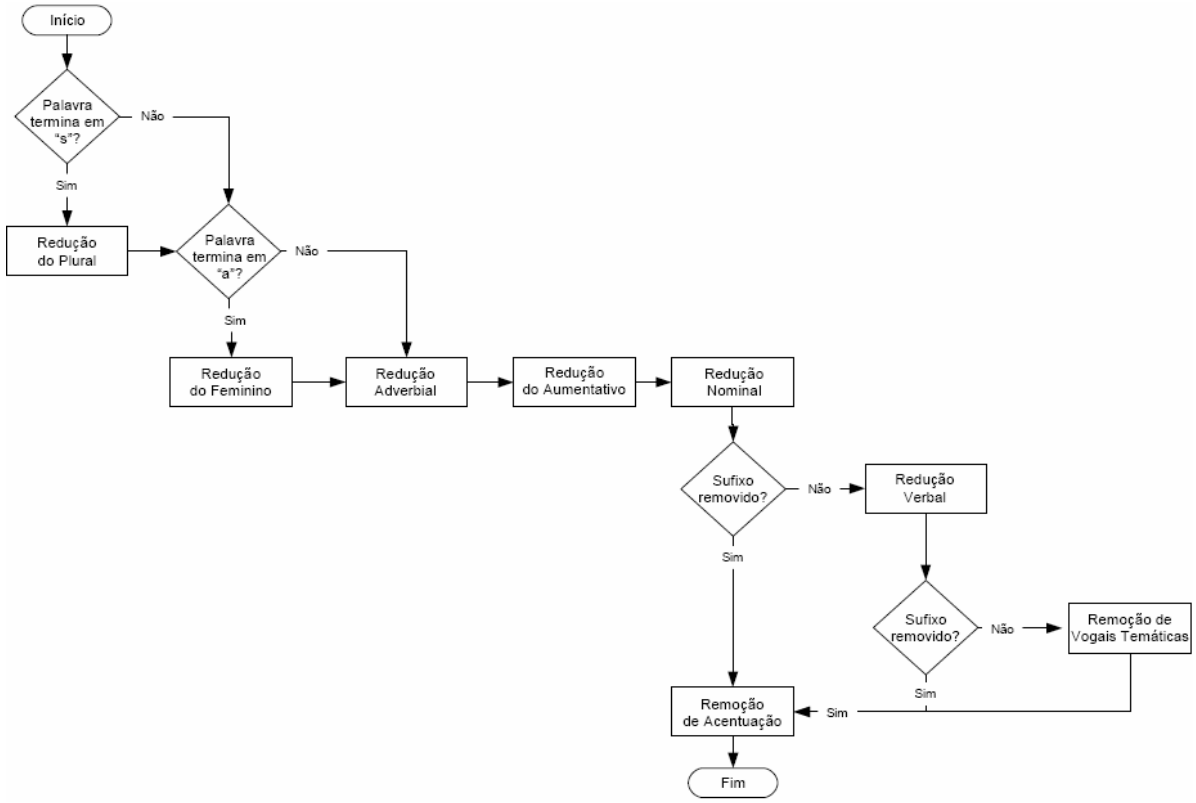

Figura 1 – Conceito do fluxo de execução dos passos do algoritmo RSLP (ORENGO; HUYCK, 2001).

2.4. *Savoy Stemmer*

O algoritmo de Savoy, originalmente escrito para a língua francesa (SAVOY, 1993) e posteriormente adaptado para o português, alemão e húngaro (SAVOY, 2006), é o mais simples entre os anteriores e foi desenvolvido com o intuído de ser um algoritmo leve. Este radicalizador é dividido em quatro passos e executa a remoção dos sufixos flexionais dos substantivos, além do sufixo adverbial “mente”.

- Passo 1: remoção do plural e do sufixo adverbial "mente";
- Passo 2: transformação de gênero feminino para masculino;
- Passo 3: retirar a vogal temática da palavra mantendo um mínimo de 3 caracteres;
- Passo 4: remoção da acentuação da palavra.

2.5. Comparação conceitual entre os radicalizadores

Segundo Flores (2009), na comparação entre os radicalizadores, Savoy apresenta o algoritmo mais leve, ocasionando muitos erros por não retirar completamente o sufixo do radical. Isto se deve pela simplicidade das etapas propostas e a implementação de poucas regras para a remoção sufixal.

O RSLP é mais ousado e considerado de maior peso, cometendo assim, erros por não extrair o sufixo completamente e por extrair parte do radical na tentativa da eliminação sufixal. Estas características são confirmadas por Flores (2009), Alvares, Garcia e Ferraz (2005) e Orengo e Huyck (2001).

Porter (2007) apresenta um algoritmo de complexidade intermediária e não tão pesado quanto o RSLP. Por se tratar de um stemmer mais leve que o RSLP, este apresenta uma quantidade inferior de erros por extrair partes do radical (FLORES, 2009). Ao contrário de RSLP, Porter não utiliza dicionário de exceções.

2.6. Avaliação qualitativa dos algoritmos

2.6.1. Método manual

Alvares (2005) descreve um método manual para avaliação da qualidade dos radicalizadores. O autor propõe três etapas, sendo: (i) a radicalização da palavra de forma direta, por um dicionário ou pelo ser humano, (ii) a execução dos algoritmos de radicalização e coleta dos radicais gerados, e (iii) o confronto dos radicais coletados na primeira etapa e os radicais gerados pelos algoritmos. Desta comparação são obtidas três medidas:

- Percentual de acertos: gerado através do número de radicais iguais gerados na primeira e na segunda etapa;
- Percentual de *overstemming*: contabilizado pelo número de vezes que o radical gerado pela segunda etapa é menos que o radical da primeira etapa;
- Percentual de *understemming*: calculado pelo número de vezes que o radical gerado na segunda etapa é maior que o radical obtido na primeira etapa.

2.6.2. Método de Paice

Paice (1994) descreve um método para medir os algoritmos de radicalização, avaliando dois erros comuns neste processo: *overstemming* e *understemming*. Erros de *overstemming*, que segundo o autor, se referem a palavras que são convertidas para o mesmo radical mas tem conceitos diferentes, e erros de *understemming*, que são gerados por palavras de mesmo conceito que são reduzidas a radicais diferentes.

Este método utiliza quatro medidas.

- *Overstemming Index* (OI): contabiliza o número de vezes que o algoritmo remove frações do radical como parte do sufixo. O OI assume um *range* de 0 a 1, onde 0 não há erros de remoção de sufixos, e 1, não há acertos.
- *Understemming Index* (UI): representa a quantidade de vezes que o sufixo não é removido completamente. Da mesma forma que o OI, este índice assume uma faixa de 0 a 1, sendo 0, todas as palavras reduzidas corretamente e 1, todas erroneamente.
- *Stemming Weight* (SW): Peso do radicalizador calculado por OI / UI. Esta medida não pode ser adotada como medida de qualidade, visto que ela apenas mede o quão pesado o algoritmo será em sua execução. Segundo o autor, algoritmos “leves” executam com segurança a tarefa de retirar apenas os sufixos das palavras, trabalhando de forma segura. Uma característica destes algoritmos é evitar erros de *Overstemming*. Já os algoritmos “pesados”, são mais corajosos, retirando todo o tipo de terminação identificada. Estes são considerados inseguros pelo fato de

remover não apenas os sufixos mas também parte do radical. Nestes radicalizadores, erros de *Overstemming* são recorrentes.

- *Error Rate Relative to Truncation* (ERRT): Medida de qualidade do algoritmo. Para aplicar esta medida é necessário calcular os valores UI e OI a uma série de algoritmos de truncamento. As coordenadas (UI, OI) de cada algoritmo, definem uma linha na qual os radicalizadores podem ser avaliados. Quanto mais distante a coordenada do ponto P (UI, OI) do radicalizador estiver da linha, melhor ele será considerado. A medida de desempenho chamada de taxa de erro em relação ao truncamento, ou ERRT, é obtida estendendo a linha que parte do ponto zero passando pelo ponto P até cruzar a linha do truncamento T. Paice menciona truncamentos de três, quatro, cinco, seis e sete letras. A Equação 1 expressa o cálculo do ERRT:

$$ERRT = length(OP)/length(OT) \quad (1)$$

onde, OP representa a linha traçada do ponto zero ao ponto P (UI, OI) de cada radicalizador e OT representa a linha do ponto zero até o cruzamento com de truncamento.

A Figura 2 demonstra a linha de truncamento exemplificando os pontos P e T de um suposto algoritmo de radicalização.

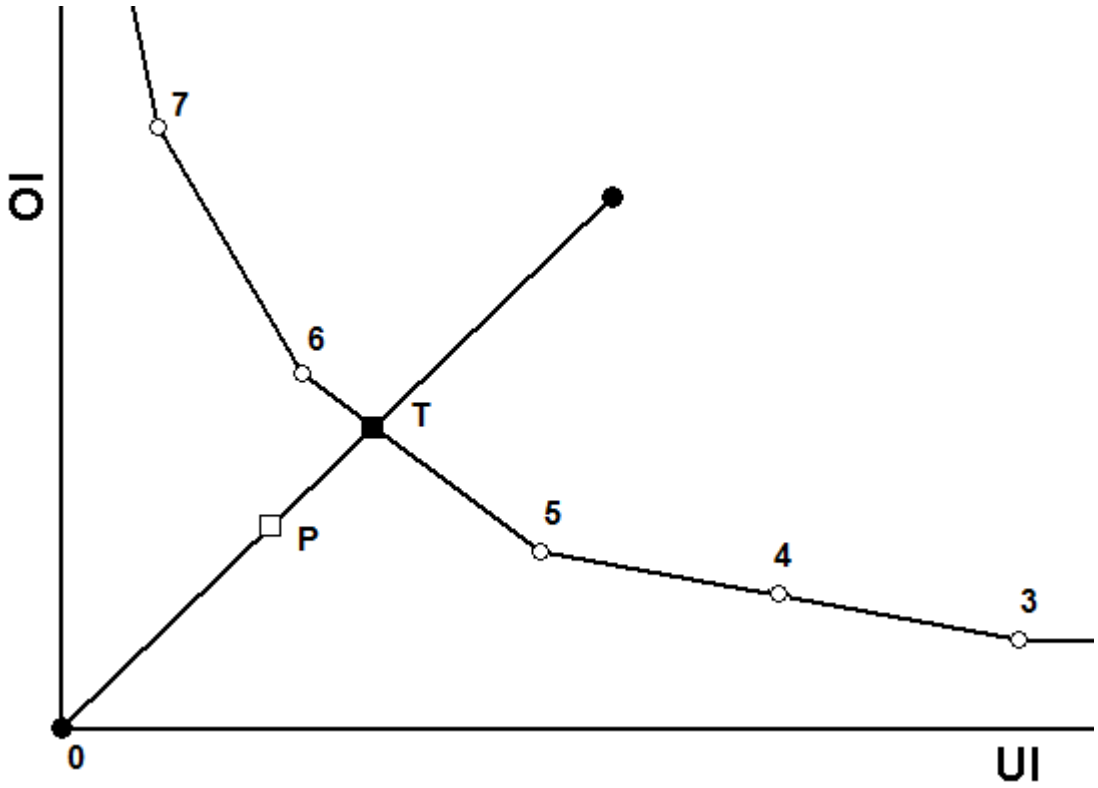

Figura 2 – Demonstração do valor de ERRT. (PAICE, 1994)

Para avaliação segundo o método, é necessário selecionar uma instância de palavras não repetidas, agrupando-as conceitualmente por seu relacionamento semântico e morfológico.

Alvares (2005) ainda categoriza a instância em duas classes, a "Classe 0" representa palavras em diferentes formas, porém semanticamente semelhantes. A "Classe 1" representa as palavras de formas diferentes, e de semântica distinta. Um algoritmo de radicalização adequado, gera o mesmo radical para as palavras da "Classe 0" e radicais diferentes para palavras da "Classe 1".

A Tabela 1 apresenta a criação dos grupos conceituais, comunicar e jogar.

Tabela 1 – Exemplo da criação dos grupos conceituais.

GRUPO 1	GRUPO 2
comunica	joga
comunicação	jogada
comunicações	jogadas
comunicada	jogadinhas
comunicador	jogado
comunicam	jogador
comunicar	jogadora
comunicaram	jogadoras
comunicou	jogadores
	jogados
	jogam
	jogando
	jogar
	jogaram
	jogaria
	jogatina
	jogava

Assim, o algoritmo de melhor qualidade seria aquele que reduzisse as palavras de cada grupo a um mesmo radical.

O método de Paice pode ser relacionado com o *Precision and Recall* de forma que se um algoritmo de radicalização gera para um grupo conceitual mais de um radical, este algoritmo comete um erro de *understemming*. Em um sistema de recuperação da informação, isto corresponde a um efeito negativo sobre a medida *Recall*. Se o algoritmo de radicalização comete erros de *overstemming*, gerando o mesmo radical para dois ou mais grupos conceituais diferentes, a medida *Precision* é prejudicada (KRAAIJ; POHLMANN, 1995).

Ainda são calculados dois totais para cada grupo:

- *Disired Merge Total* (DMT): Somatório dos pares de palavras que podem ser encontrados no grupo. Este total é calculado pela Equação 2:

$$DMTg = 0{,}5\, ng\,(ng - 1) \tag{2}$$

  onde, ng se refere ao número de palavras no grupo. Seguindo o exemplo do Grupo 1, ng = 9, o DMTg = 36. Isto significa que pode-se formar 36 pares de palavras diferentes com o referido grupo.

- *Desired Non-merge Total*: Somatório de pares de palavras que podemos formar entre uma palavra membro do grupo e outras não-membro deste grupo, expresso pela Equação 3:

$$DNTg = 0{,}5\, ng\,(W - ng) \tag{3}$$

  onde, ng representa o número de palavras no grupo, e W o número total de palavras na amostra. Assim, calculando o DNT para o Grupo 1 tem-se, ng = 9, W = 26 e DNTg = 76,5.

O Somatório dos valores DMT de cada grupo conceitual é chamado de *Global Disired Merge Total* (GDMT). Da mesma forma, somando os valores DNT de cada grupo tem-se o *Global Disired Non-merge Total* (GDNT).

O parâmetro *Unachieved Merge Total* (UMT), calculado após o processamento das

amostras pelo algoritmo de radicalização, representa o número de erros de *understemming* do processo, como apresentado na Equação 4:

$$UMTg = 0{,}5 \sum_{i=1..s} ui(ng - ui) \tag{4}$$

onde, ng representa o número de palavras no grupo, s representa o número de radicais distintos em cada grupo e ui o número de instancias de cada radical no grupo. A soma de todos os UMT é representada pelo *Global Unachieved Merge Total* (GUMT). O *Understemming Index* (UI), é calculado pela divisão GUMT / GDMT.

Para contabilizar os erros de *Overstemming*, onde um mesmo radical aparece em grupos diferentes, a amostra deve ser reorganizada em grupos que compartilham o mesmo radical e calculado o *Wrongly Merged Total* (WMT), obtido pela Equação 5:

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

Onde, t representa o número de grupos conceituais originais que compartilham do mesmo radical, ns o número de repetições do radical e vi o número de palavras de cada grupo conceitual original que compartilham o radical. A soma de todos os valores de WMT é representada por *Global Wrongly Merged Total* (GWMT). *Overstemming Index* (OI) é calculada através da divisão: GWMT / WMT. A ultima etapa é calcular SW, sendo expresso pela Equação 6:

$$SW = OI / UI \tag{6}$$

Paice (1994) avalia seu método comparando três radicalizadores, *Lovins Stemmer* (LOVINS, 1968), *Porter Stemmer* (PORTER, 1980) e *Paice / Husk Stemmer* (PAICE, 1990). Para a aplicação da medida ERRT foram utilizados os truncamentos com quatro, cinco, seis, sete e oito letras. A instância de dados coletada consiste em 9.757 não repetidas extraídas de uma coleção de artigos sobre Biblioteconomia e Ciência da Informação. A classificação dos algoritmos segundo a medida ERRT apresentou *Paice / Husk Stemmer* com melhor qualidade, seguido por *Porter Stemmer* e *Lovins Stemmer.*

Alvares (2005) compara o seu radicalizador STEMBR com o RSLP (ORENGO; HUYCK, 2001) utilizando o método de Paice (1994). Para este trabalho foram formadas duas instâncias de dados, uma extraída do Dicionário Aurélio Eletrônico e outra do LexWeb (MONTEIRO JUNIOR, 2004). Avaliando a primeira instância, o algoritmo STEMBR se apresentou mais pesado que o RSLP. Nesta instância o percentual de acertos foi de 62,20% e 55,30% respectivamente. A avaliação da segunda instância da mesma forma demonstrou que o STEMBR é mais pesado que o RSLP e os percentuais de acerto foram de 69,20% e 67,60% respectivamente.

Flores (2009) aplica o método de Paice para comparar os radicalizadores STEMBR (ALVARES, 2005), *Porter Stemmer* (PORTER, 1980), *Savoy Stemmer* (SAVOY, 2006), RSLP (ORENGO; HUYCK, 2001), alem de RSLP-S que é uma variação de RSLP que executa apenas a redução do plural e *Stemmer-S* que retira a letra "s" do final das palavras. A avaliação da medida ERRT apresenta RSLP como melhor *stemmer,* seguido por *Porter Stemmer,* STEMBR, RSLP-S, Stemmer-S e *Savoy Stemmer.*

Outros trabalhos apresentam comparações entre algoritmos de radicalização, dentre os quais pode-se citar Harman (1991), Krovetz (1993), Hull (1996), Fuller e Zobel (1998) e Orengo, Buriol e Coelho (2007), porem estes não utilizam o método de Paice (1994) para medir a qualidade dos radicalizadores.

3. Materiais e Métodos

3.1. Preparação da instância de dados

Para a realização de uma avaliação consistente dos algoritmos de radicalização, foi necessário preparar uma instância de dados significativa para a abordagem do problema.

No desenvolvimento desta etapa, foram avaliados os arquivos do Diário Oficial[1] do Município de Cachoeiro de Itapemirim-ES, sendo 15 publicações entre os dias 03 e 25 de Março 2008 e 35 publicações entre 22 de Março e 30 de Maio de 2009. Dentre estes arquivos foram selecionados inicialmente 250 palavras de maior ocorrência. Logo após, filtradas as palavras mais significativas, restando 106 palavras de grande representatividade no contexto dos documentos selecionados.

Assim, formados os 106 grupos, foi utilizado o Dicionário Português do OpenOffice[2] para selecionar palavras semântica e morfologicamente relacionadas, totalizando 731 palavras distintas. Como exemplo da construção dos grupos, as palavras "MUNICIPAL" e "SECRETARIA", com um percentual de representatividade de 15,377% e 5,904% respectivamente, e ocorrência de 2.464 e 946 vezes nos documentos selecionados.

Na Tabela 2 são apresentados os grupos conceituais formados a partir das palavras “MUNICIPAL” e “SECRETARIA”, com seus respectivos números de ocorrência e percentual de representação nos documentos selecionados.

Tabela 2 – Grupos conceituais criados a partir das palavras “MUNICIPAL” e “SECRETARIA”.

GRUPO	TERMO	OCORRÊNCIAS	REPRESENTAÇÃO
MUNICIPAL	Municipais municipal município municípios municipalidade	2464	15,337
SECRETARIA	secretaria secretária secretariado secretarias secretárias secretário secretários	946	5,904

3.2. Execução dos algoritmos

Os algoritmos desenvolvidos na linguagem de programação Java, fazem parte da biblioteca de algoritmos para radicalização PTStemmer[3]. Os radicalizadores utilizados foram PORTER, SAVOY e RSLP.

A execução dos algoritmos gerou a uma lista contendo os grupos de palavras e, para cada grupo, uma sublista contendo o radical de cada palavra associada ao grupo, para cada algoritmo proposto. A Tabela 3 mostra os grupos “MUNICIPAL” e “SECRETARIA”, as palavras associadas, os *stemmings* gerados pelos algoritmos e o radical extraído do Lexicográfico da Língua Portuguesa (HOUAISS, VILLAR e FRANCO, 2001).

[1] Disponível em http://www.cachoeiro.es.gov.br/transparencia/site.php?id=DIARIO.
[2] Disponível em http://openoffice.caixamagica.pt/stable/3.0.1/extensions/Dictionary_pt/oo3x-pt-PT.oxt.
[3] Disponível em http://code.google.com/p/ptstemmer.

Tabela 3 – Resultados da execução dos algoritmos de radicalização.

GRUPO	TERMO	DIC	PORTER	RSLP	SAVOY
MUNICIPAL	municipais	municip	municip	municip	municipal
	municipal	municip	municipal	municip	municipal
	município	municip	municipi	municipi	municipi
	municípios	municip	municipi	municipi	municipi
	municipalidade	municip	municipal	municipal	municipalidad
SECRETARIA	secretaria	secret	secret	secret	secretari
	secretária	secret	secret	secret	secretari
	secretariado	secret	secretari	secretari	secretariad
	secretarias	secret	secret	secret	secretari
	secretárias	secret	secret	secret	secretari
	secretário	secret	secretari	secreta	secretari
	secretários	secret	secretari	secreta	secretari

3.3. Aplicação da medida ERRT

De acordo com a Equação 1, para determinar a interseção da reta OP formada pelos pontos (0, 0) e P(UI, OI) de cada algoritmo de radicalização com a linha de truncamento através da equação paramétrica das retas, inicialmente é preciso calcular se estas retas são paralelas. A Equação 7 expressa o cálculo de paralelismo das retas:

$$det = (n.x - m.x) * (l.y - k.y) - (n.y - m.y) * (l.x - k.x) \quad (7)$$

Onde, det é o determinante de paralelismo a ser obtido, k representa o ponto de origem da reta 1, l o ponto de destino da reta 1, m é o ponto de origem da reta 2 e n o ponto de destino da reta 2. Se o valor de det for diferente de zero, significa que as retas não são paralelas. Caso contrário, a identificação do ponto de interseção das retas é inviável e impossível continuar.

Sendo possível calcular o determinante det das retas não paralelas, o passo seguinte é calcular o valor dos parâmetros s e t das retas, representadas pelas Equações 8 e 9 respectivamente.

$$s = ((n.x - m.x) * (m.y - k.y) - (n.y - m.y) * (m.x - k.x)) / det \quad (8)$$

$$t = ((l.x - k.x) * (m.y - k.y) - (l.y - k.y) * (m.x - k.x)) / det \quad (9)$$

Onde, s representa o valor do parâmetro no ponto de interseção sobre a reta kl, t representa o valor do parâmetro no ponto de interseção sobre a reta mn, k é o ponto de origem da reta 1, l o ponto de destino da reta 1, m representa o ponto de origem da reta 2, n o ponto de destino da reta 2 e det o determinante obtido através da Equação 7.

As Equações 10 e 11 definem as coordenadas x e y respectivamente do ponto de interseção na reta kl.

$$x = k.x + (l.x - k.x) * s \quad (10)$$

$$y = k.y + (l.y - k.y) * s \quad (11)$$

Onde, x representa a coordenada x do gráfico cartesiano a ser obtida e y a coordenada y do gráfico cartesiano, k o ponto de origem da reta 1, l é o ponto de destino na reta 1.

Após o cálculo do ponto de interseção das retas, é possível obter as distâncias das retas OP composta por O(0, 0) e P(UI, OI), OT que é composta por O(0,0) e T(x, y) obtidos através das equações 10 e 11, utilizando a Equação 12 das distâncias cartesianas.

$$D=\sqrt{(x2-x1)^2+(y2-y1)^2} \quad (12)$$

onde, D representa a distancia a ser calculada, x1 a coordenada x no ponto de origem, y1 a coordenada y no ponto de origem, x2 a coordenada x no ponto de destino e y2 a coordenada y no ponto de destino.

4. Resultados

Confrontando os radicais gerados pelos algoritmos de radicalização e os radicais extraídos do Dicionário Lexicográfico da Língua Portuguesa (HOUAISS, VILLAR e FRANCO, 2001), calculou-se os níveis de erro de cada *stemmer*, também um cálculo manual de o*verstemming*, quando o radical de Houaiss, Villar e Franco (2001) for maior que o radical gerado, e *understemming*, quando o radical do dicionário for menor que o radical gerado.

Em contraposição ao método manual de avaliação qualitativa dos radicalizadores, foi calculado o percentual de erros, enquanto o método apresenta o percentual de acertos. Esta variação foi necessária devido às características da medida ERRT de Paice, onde, quanto maior o valor, pior é considerado o algoritmo. Desta mesma forma, quanto maior o valor do "% ERRO", pior pode ser considerado o radicalizador. A Tabela 4 demonstra os percentuais de erros encontrados no confronto entre os radicais gerados pelos algoritmos de radicalização e Houaiss, Villar e Franco (2001).

Tabela 4 – Percentual de erros na execução dos algoritmos calculados pelo método manual.

STEMMER	% ERRO	% OVERSTEMMING	% UNDERSTEMMING
PORTER	34,06	6,43	93,57
RSLP	29,82	16,97	82,57
SAVOY	85,69	0,00	99,67

O percentual de erro é calculado com base no número de palavras que o algoritmo gerou erroneamente. Já os percentuais de *overstemming* e *understemming* são calculados a dentre os radicais gerados de forma errada, quantos são menores ou maiores, respectivamente, que o radical obtido a partir do dicionário.

Para a avaliação dos grupos conceituais segundo o método de Paice (1994), calculou-se as medidas *Understemming Index* (UI), *Overstemming Index* (OI), *Stemming Weight* (SW), *Global Unachieved Merge Total* (GUMT) e *Global Wrongly Merged Total* (GWMT) a partir dos resultados obtidos através da execução dos algoritmos.

A Tabela 5 demonstra os índices obtidos através da avaliação dos algoritmos.

Tabela 5 – Aplicação do método de Paice.

STEMMING	GUMT	GWMT	UI	OI	SW
PORTER	1197	44	0,35341010	0,00016703	0,00047262
RSLP	1215	64	0,35872453	0,00024295	0,00067726
SAVOY	3011	18	0,88898730	0,00006833	0,00007686

É possível comparar os resultados da execução do método através da notação que Paice propõe:

UI(SAVOY) > UI(RSLP) > UI(PORTER)
OI(RSLP) > OI(PORTER) > OI(SAVOY)
SW(RSLP) > SW(PORTER) > SW(SAVOY)

É importante ressaltar que os valores encontrados no *Stemming Weight* (SW) não são conclusivos para julgar a qualidade do radicalizador, visto que esta medida é descrita pelo autor do método como a relação entre OI e UI. Contudo é possível classificar os algoritmos quanto ao peso.

O algoritmo de PORTER apesar de ter alcançado valores próximos ao RSLP para o índice de OI, pode ser classificado através do SW, como o radicalizador de peso intermediário para este estudo. SAVOY, por outro lado, devido ao alto valor de UI e baixo valor de OI, recebe o menor peso em SW, sendo classificado como o *stemmer* mais leve. O algoritmo RSLP, com maior valor de OI, indicando muitos erros de *overstemmig*, foi classificado aqui, como o mais pesado.

Considerando que RSLP foi classificado como um *stemmer* de maior peso, este apresentou os menores percentuais de erro na comparação do radical gerado pelo algoritmo e o radical extraído do Dicionário Houaiss, Villar e Franco (2001).

Para aplicação da medida ERRT, foram utilizados os algoritmos de truncamento de três, quatro, cinco, seis e sete letras, aplicados a instancia de dados proposta. O cálculo dos valores de UI e Oi dos algoritmos de truncamento estão descritos na Tabela 6.

Tabela 6 – Valores de UI e OI calculados a partir dos truncamentos.

STEMMING	GUMT	GWMT	UI	OI	SW
TRUNC 3	0	5226	0,00000000	0,01983844	--
TRUNC 4	17	805	0,00501919	0,00305586	--
TRUNC 5	125	356	0,03690582	0,00135141	--
TRUNC 6	623	28	0,1839385887	0,0001062909	--
TRUNC 7	1477	0	0,4360791261	0	--

Após a aplicação do método para os algoritmos de truncamento, foi construído o gráfico cartesiano a partir das coordenadas (UI, OI) e traçada uma linha iniciando no ponto TRUNC 3, passando por cada ponto gerado. A Figura 3 demonstra as linhas geradas a partir dos pontos de truncamento.

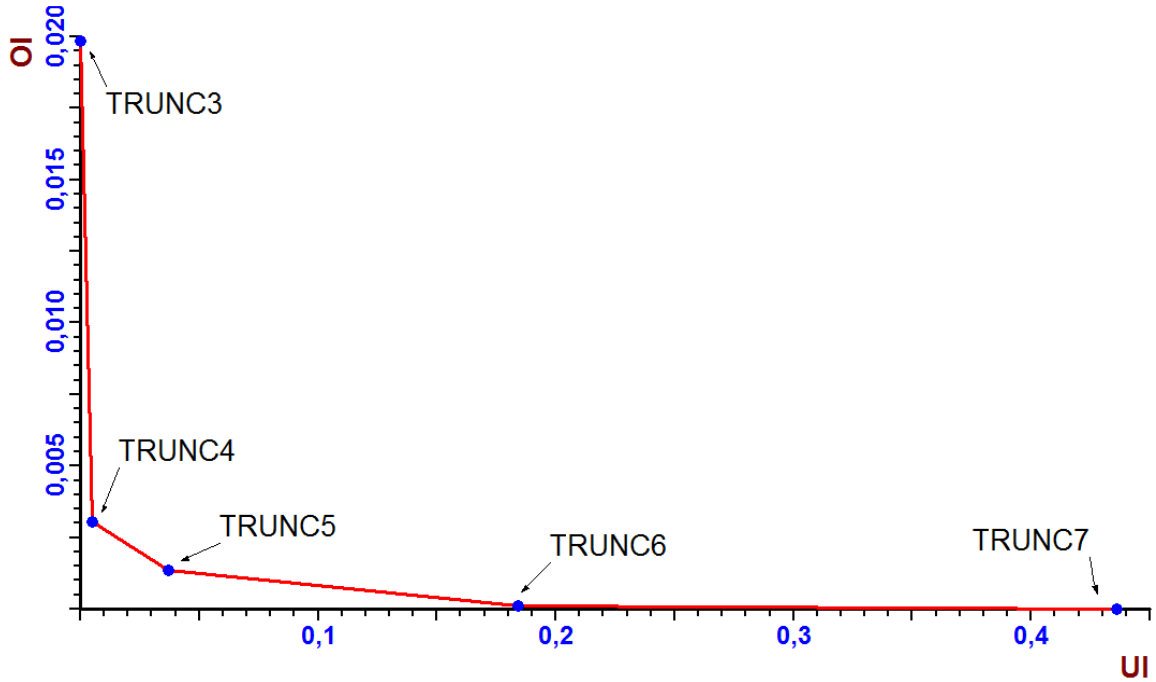

Figura 3 – Linha de corte de qualidade dos radicais.

O cálculo de ERRT expresso pelas distancias das retas OP e OT são apresentados na Tabela 7.

Tabela 7 – cálculo dos valores de ERRT.

	OP	OT	ERRT
PORTER	0,1467347374	0,2055875150	0,7137336983
RSLP	0,1489412924	0,1819394410	0,8186311421
SAVOY	0,3691046335	0,3691046335	1

É possível avaliar a qualidade dos *stemmers* pela medida ERRT. O algoritmo que demonstrou pior qualidade para este estudo foi SAVOY apresentando ERRT = 1. PORTER obteve o melhor resultado, com valores de ERRT ≅ 0,7137. RSLP esteve próximo a PORTER, porém com valores de ERRT ≅ 0,8189.

A Figura 4 demonstra a aplicação da medida ERRT dos algoritmos de radicalização através dos pontos P(UI, OI), comparando com a linha de qualidade gerada pelo truncamento T.

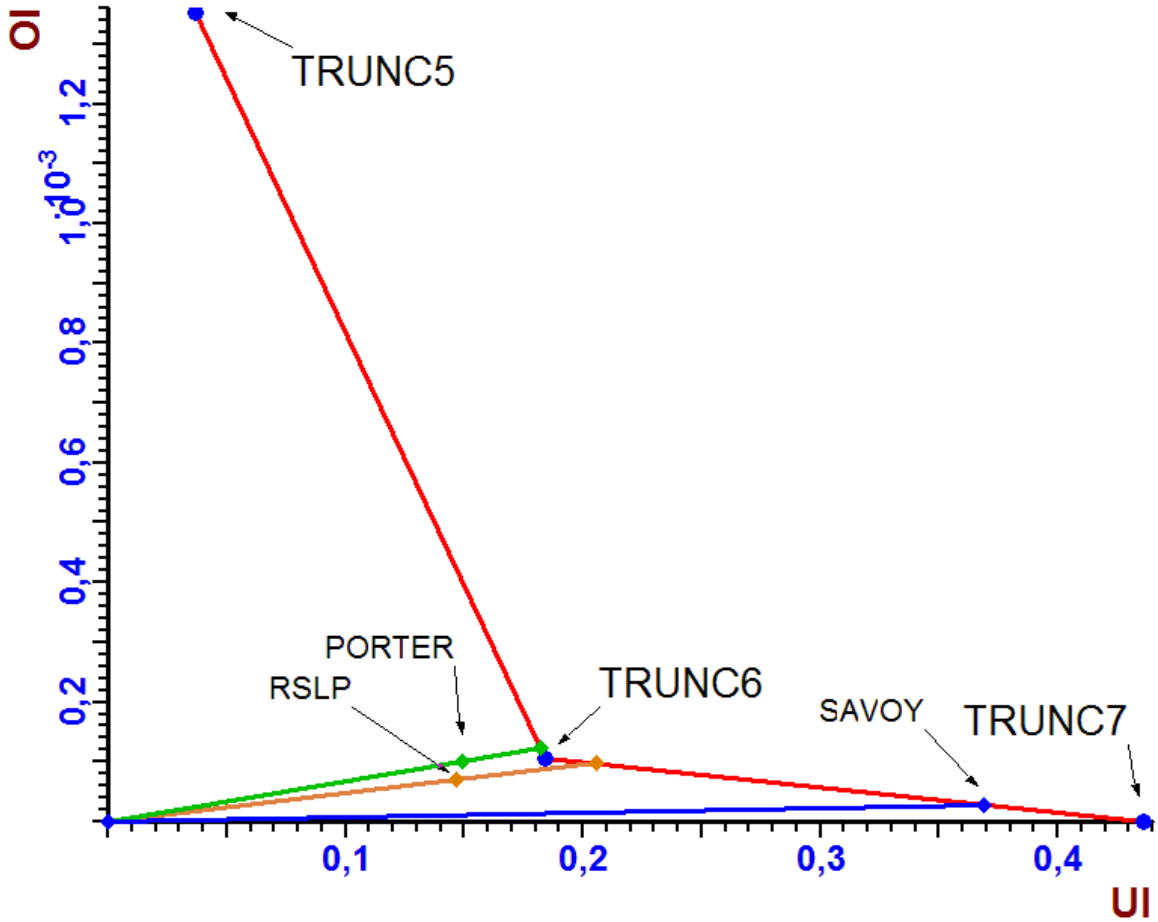

Figura 4 – Representação dos pontos P(UI, OI) dos algoritmos de radicalização.

Através do gráfico da Figura 4, é possível visualizar a linha de truncamento em vermelho, em verde a linha do algoritmo de PORTER, em laranja RSLP e em azul SAVOY. Os losangos marcam o ponto P(UI, OI) dos algoritmos e a interseção com a linha de truncamento. SAVOY que obteve ERRT = 1, por isso o ponto P(UI, OI) é o mesmo ponto T da interseção com alinha de truncamento.

Na avaliação através dos percentuais de erro dos algoritmos, RSLP obteve melhor desempenho que PORTER, alcançando 29,82% de erro contra 34,06%, respectivamente. No método proposto por Paice (1994), PORTER apresentou melhores valores de ERRT, seguido de

RSLP, sendo, PORTER ≅ 0,7137 e RSLP ≅ 0,8189.

Como critério de desempate, foram analisadas as subtrações dos valores de % de erro e ERRT para cada radicalizador. O algoritmo que obtiver menor valor de subtração destas medidas pode ser considerado de melhor qualidade. A Tabela 8 apresenta os valores de ERRT e % de erro para os algoritmos de PORTER e RSLP. A medida ERRT foi multiplicada por 100 para se obter o percentual de ERRT.

Tabela 8 – Critério de desempate entre algoritmos de radicalização.

STEMMER	ERRTx100	% Erros	Subtração
PORTER	71,37	34,06	37,31
RSLP	81,89	29,82	52,07

Analisando a diferença entre ERRTx100 e % Erros, PORTER obteve diferença significativa em relação a RSLP. Os valores da subtração das medidas de qualidade são suficientes para julgar qual melhor se adapta a instancia de dados proposta. O algoritmo de PORTER obteve 37,31 e RSLP 52,07, isto significa uma diferença de 14,76 pontos de PORTER em relação a RSLP.

5. Conclusão

Este trabalho apresentou uma comparação entre três algoritmos de radicalização da língua portuguesa, PORTER, RSLP e SAVOY. Através deste estudo foi possível classificar os algoritmos quanto ao peso em relação a radicalização da amostra gerada para este estudo.

O algoritmo RSLP foi considerado o mais pesado dos três, seguido por PORTER, que apresentou peso intermediário apesar de não muito distante de RSLP. SAVOY apresentou peso SW mais leve entre os três algoritmos.

O RSLP obteve menor percentual de erros na radicalização da instância de dados criada, onde das 731 palavras, obteve 29,82% de erros. Destes erros 16,97% erros de *Overstemming* e 82,56% erros de *Understemming*. Estes resultados comparados com PORTER (OI = 6,43%) e SAVOY (OI = 0%) reforçam a tendência de maior peso deste radicalizador.

O método de Paice por outro lado, mostrou que o radicalizador de maior qualidade é PORTER, que se destacou com índice ERRT de menor valor, com RSLP logo em seguida. SAVOY por sua vez, devido a sua construção leve, apresenta muitos erros de *Understemming* e seus resultados não foram adequados para esta tarefa. Este *stemmer* obteve valor de ERRT igual a 1, o que significa dizer que é tão eficiente quanto a rotinas de truncamento.

Através da subtração das duas medidas de qualidade propostas neste trabalho, % Erro e ERRT, é possível estabelecer um critério de desempate adequado. Através desta medida, o algoritmo de PORTER foi considerado mais adequado para radicalização conjunto de dados proposto.

Referências Bibliográficas

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

ALVARES, R. V.; GARCIA, A. C. B.; FERRAZ, I.. (2005) STEMBR: A Stemming Algorithm for the Brazilian Portuguese Language. 12th Portuguese Conference On Artificial Intelligence, p.693-701.

ARANHA, C.; PASSOS, E. (2006). A Tecnologia de Mineração de Textos. Resi-revista Elerônica de Sistemas de Informação, Rio de Janeiro, n. 2, p.1-8.

BARION, E. C. N.; LAGO, D. (2008). Mineração de Textos. Revista de Ciência Exatas e Tecnologias, São Paulo, v. , n. 3, p.123-140.

BRANSKI, R. M.(2004). Recuperação de informações na Web. Perspectivas em Ciência da Informação, Belo Horizonte, v. 9, n. 1, p. 70-87.

CHEN, H. (2001). Knowledge Management Systems: A Text Mining Perspective. Tucson, Arizona: University Of Arizona (knowledge Computing Corporation).

CORRÊA, A. C. G.. (2003). Recuperação de documentos baseada em Informação Semântica no ambiente AMMO. 2003. 92 f. Dissertação de Mestrado (1) - Universidade Federal de São Carlos, São Carlos.

FLORES, F. N. (2009). Avaliando o Impacto da Qualidade de um Algoritmo de Stemming na Recuperação de Informações. Trabalho de Graduação (Bacharelado em Ciência da Computação) – UFRGS, Porto Alegre.

FRAWLEY, W. J.; PIATETSKY-SHAPIRO, G..; MATHEUS, C. J. (1992). Knowledge Discovery in Databases: An Overview. Ai Magazine, [s. L.], v. 13, n. 3, DOI: 10.1007/3540635149_30.

FULLER, M.; ZOBEL, J. (1998). Conflation-based Comparison of Stemming Algorithms. In: Proceedings of the Third Australian Document Computing Symposium, Sydney, Australia.

GOMES, G. R. (2006). Integração de Repositórios de Sistemas de Bibliotecas Digitais e de Sistemas de Aprendizagem. Tese (Doutorado em Informática) – Pontifícia Universidade Católica, Rio de Janeiro.

GOMES, R. M. (2009). Desambiguação de Sentido de Palavras Dirigida por Técnicas sob o Enfoque da Mineração de Texto. Dissertação (Mestrado em Engenharia Elétrica) – Pontifícia Universidade Católica, Rio de Janeiro.

HARMAN, D. (1991). How Effective is Suffixing? In: Journal of the American Society for Information Science, New York, v. 42, n. 1, p. 7-15. DOI: 10.1002/(SICI)1097-4571(199101)42:1<7::AID-ASI2>3.0.CO;2-P

HOUAISS, A.; VILLAR, M. S.; FRANCO, F. M. M. (2001). Dicionário Houaiss da Língua Portuguesa. Instituto de Lexicografia e Banco de Dados da Língua Portuguesa. Editora Objetiva. ISBN 9788573029635.

HULL, D. A. (1996). 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. DOI: 10.1002/(SICI)1097-4571(199601)47:1<70::AID-ASI7>3.3.CO;2-Q

KRAAIJ, K.; POHLMANN, R. (1995). Evaluation of a Dutch Stemming Algorithm. In: The New Review of Document & Text Management, T. G. Publishing, London, UK, p. 25- 43. DOI: 10.1.1.41.5707

KROVETZ, R. (1993). Viewing Morphology as an Inference Process. In: 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 191-202. DOI: 10.1145/160688.160718.

MONTEIRO JÚNIOR, A. (2004). LexWeb: um léxico da Língua Portuguesa extraído automaticamente da Internet. 78 f. Dissertação (Mestrado em Engenharia Elétrica) –Universidade Federal do Pará, Belém.

LOVINS, J. B. (1968). Development of a stemming Algorithm. Mechanical Translation and Computational Linguistics. 11, 22-31.

ORENGO, V. M.; HUYCK, C. (2001). A Stemming Algorithm for the Portuguese Language. 8th International Symposium On String Processing And Information Retrieval (spire), Laguna de San Raphael, Chile, n. , p.183-193, DOI: 10.1109/SPIRE.2001.989755

ORENGO, V. M.; BURIOL, L. S.; COELHO, A. R. (2007). A Study on the Use of Stemming for

Monolingual Ad-Hoc Portuguese Information Retrieval. In: CLEF 2006, Alicante, Berlin Heidelberg: Springer-Verlag. v. 4730 p. 91-98. DOI: 10.1007/978-3-540-74999-8_12.

PAICE, C. D. (1990). Another stemmer. SIGIR Forum; 24,56-61. DOI: 10.1145/101306.101310.

PAICE, C. D. (1994). An evaluation method for stemming algorithms. In SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval. Springer-Verlag New York, Inc., pp. 42#50. ISBN 0-387-19889-X

ROYAL, P.. Internet 2011 in numbers. Disponível em: <http://royal.pingdom.com/2012/01/17/internet-2011-in-numbers/>. Acesso em: 28 Mai.2012;

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

PORTER, M. F. (2007) Portuguese stemming algorithm. Disponível em <http://snowball.tartarus.org/algorithms/portuguese/stemmer.html>. Acesso em 12 set. 2011.

SAVOY, J. (1993). Stemming of French words based on grammatical categories. Journal of the American Society for Information Science, 44(1), 1–9. DOI: 10.1002.

SAVOY, J. (2006). Light stemming approaches for the French, Portuguese, German and Hungarian languages. Proceedings ACM-SAC, The ACM Press, p. 1031-1035. DOI: 10.1145/1141277.1141523