3
Mineração de Texto

A existência de ferramentas que realmente suportam todo o processo de KDD ainda é bastante restrita. Ferramentas comerciais, tais como MineSet e IntelligentMiner, geralmente têm um caráter mais exploratório e fazem uso de algoritmos e ferramentas proprietárias, o que dificulta o seu uso por pesquisadores.

De forma geral um processo de mineração de textos contém quatro macro etapas: coleta, pré-processamento, indexação e análise da informação. Nos próximos parágrafos é descrito o processo como um todo e em seguida cada uma das etapas de forma mais detalhada.

A etapa inicial tem por objetivo a coleta de das informações que vão compor a base textual de trabalho, isto é, determinar e selecionar o universo de atuação das técnicas de mineração de texto. Por outro lado, nenhuma informação que não esteja contida na base textual poderá ser extraída, encontrada ou utilizada de alguma forma.

Após a coleta de documentos é necessário transformar os documentos em um formato propício para serem submetidos aos algoritmos de extração automática de conhecimento. Essa segunda etapa, denominada de pré-processamento, é responsável por obter uma representação estruturada dos documentos, geralmente no formato de uma tabela atributo-valor.

Essa tabela atributo-valor que representa os documentos tem como característica valores esparsos dos dados e uma alta dimensionalidade. Essas características são inerentes à problemas relacionados ao processo de MT, pois cada palavra presente nos documentos pode ser um possível elemento do conjunto de atributos dessa tabela atributo-valor. É, portanto, uma etapa bastante custosa e um cuidadoso pré-processamento dos documentos é imprescindível ao sucesso de todo o processo de MT.

Após os documentos serem representados em um formato adequado, é possível aplicar técnicas de extração de conhecimento utilizando sistemas de mineração de dados. Caso os documentos estejam representados no formato de

PUC-Rio - Certificação Digital Nº 0310437/CA

uma tabela atributo-valor, geralmente, na terceira etapa, são empregados métodos de RI como indexação para aumentar a performance do processo.

Finalmente, na última etapa, o objetivo é descobrir padrões úteis e desconhecidos presentes nos documentos. Para a extração de padrões, são utilizadas técnicas de forma semelhante ao processo tradicional de MD.

A seguir, então, explicaremos de forma mais detalhada cada uma das etapas envolvidas no processo de mineração de texto, dando mais ênfase à etapa de pré-processamento. Vale ressaltar que o processo descrito a seguir é o processo clássico, que servirá de comparação com o modelo proposto nessa tese a ser apresentado no próximo capítulo.

3.1. Coleta de Dados

A coleta de dados tem como função formar a base textual de trabalho. Essa base pode ser estática, nos casos mais simples, ou dinâmica, isto é, atualizadas a todo momento através de robôs autônomos coletando novas informações. A atualização é feita pela simples adição de um novo conteúdo, remoção de conteúdos antigos, ou, substituição da base por uma inteiramente nova.

Coletar dados é uma atividade trabalhosa. Um do motivos é que os dados podem não estar disponíveis em um formato apropriado para serem utilizados no processo de mineração de textos. Essa dificuldade não é nova, em (Pyle, D., 1999) é apresentada uma lista de alguns desafios para essa fase.

Para mineração de textos, um dos principais problemas em coletar dados é descobrir onde os dados estão armazenados. Depois disso recuperar documentos relevantes ao domínio de conhecimento. De forma geral, esse procedimento se estabelece basicamente em três ambientes distintos: no diretório de pastas do disco rígido; em tabelas de diferentes bancos de dados e na Internet.

Para o disco rígido temos os sistemas de GED (gerenciamento eletrônico de documentos) para grandes empresas e recentes lançamentos de busca local como Google Desktop, Yahoo! Desktop e Ask Jeeves Desktop.

Nos bancos de dados, a iniciativa de *Data Warehouses* (Kimball, R., 1996) surgiu com o intuito de unificar e centralizar diferentes bancos de dados de forma disponibilizar mais facilmente as informações. Embora os *Data Warehouses*

PUC-Rio - Certificação Digital Nº 0310437/CA

facilitem bastante a coleta de dados, o problema ainda está longe de ser bem resolvido, principalmente quando se trata de textos. Além disso, esse modelo se mostrou bastante custoso e árduo, o que fez com que se estabelecesse apenas em pouco lugares de forma definitiva. (Batista, 2003)

Na Internet temos uma infinidade de páginas pessoais, institucionais, páginas de revistas e diversas fontes disponíveis para coletar os documentos tais como livros e artigos. Para facilitar o acesso a esses documentos na Internet, muitas ferramentas de apoio têm sido construídas usando as seguintes abordagens: Motores de Busca Baseados em Robô (*Robotic Internet Search Engines*), Diretórios de Assunto (*Subject Directo*ries) (Peterson, R. E., 1997).

Trabalhos relacionados à coleta de documentos provenientes da Internet podem ser encontrados na literatura (Baeza-Yates, B. e Ribeiro Neto, B., 1999); (Joachims, T. et al, 1997). Muitos deles combinam técnicas de AM e Recuperação de Informação (RI) (van Rijsbergen, C. J., 1979) para determinar o perfil do usuário visando melhorar a coleta de documentos.

Em qualquer desses ambientes, um *crawler* é o robô responsável por navegar de forma autônoma e exploratória pela rede para fazer a coleta. Esses robôs se tornaram mais conhecidos na Internet com o nome de *webcrawler*. Versões livres de *webcrawlers* podem ser encontradas na Internet, a exemplo do wGet. Uma das importantes funções de um *webcrawler* é saber decodificar os HTMLs, tanto para recortar apenas o que é conteúdo texto como para seguir para o hiperlinks que se encontram na página. Outra função importante é saber gerenciar bem seu caminho de percurso que tem a forma de um grafo de modo a impedir que o robô visite várias vezes a mesma página ou entre em ciclos eternos.

3.2. Pré-processamento

O pré-processamento de textos consiste em um conjunto de transformações realizadas sobre alguma coleção de textos com o objetivo de fazer com que esses passem a ser estruturados em um representação atributo-valor. De modo geral, a etapa de pré-processamento tem por finalidade melhorar a qualidade dos dados já disponíveis e organizá-los. As ações realizadas na etapa de pré-processamento de

PUC-Rio - Certificação Digital Nº 0310437/CA

dados visam prepará-los para serem submetidos a algum algoritmo de indexação ou mineração de dados.

Conceitualmente, em um processo de mineração, essas transformações consistem em identificar, compactar e tratar dados corrompidos, atributos irrelevantes e valores desconhecidos (Batista, G. E. A. P. A., 2003); (Weiss, S. M. e Indurkhya, N., 1998).

Em mineração de textos, pré-processamento normalmente significa dividir o texto em palavras, aplicar técnicas de stemming, remover as stop-words e classificá-las segundo a classe gramatical (Five Steps).

No entanto, a etapa de pré-processamento vai além das ações citadas, pois é necessário transformar os textos em uma representação estruturada adequada para que, a partir disso, os dados possam ser submetidos ao processo como um todo. No entanto, durante a transformação dos textos em formato estruturado existe a possibilidade de que informação intrínseca ao conteúdo dos textos seja perdida. Um desafio, nesse caso, é obter uma boa representação minimizando a perda de informação. A etapa de pré-processamento em um processo de MT é, portanto, fundamental para o desempenho de todo o processo (Martins, C. A., 2003).

3.2.1. Identificação de Palavras no Texto

Em (Gean, C. C. e Kaestner, C. A. A., 2004) podemos perceber uma preocupação na definição da unidade básica de texto, que é denominada de palavra (termo). Na etapa de pré-processamento, o documento, considerado como sendo texto "puro" ou não-anotado, livre de qualquer formato, é tratado de maneira a produzir uma representação mais compacta que seja mais adequada à realização da tarefa objetivo.

Em Salton (1983), a identificação das palavras nos documentos a serem indexados nada mais é do que a identificação de palavras analisando-se as seqüências de caracteres no texto. Salnton aconselha fazer um *Dictionary lookup*, ou seja, comparar as seqüências de caracteres retiradas do texto com um dicionário a fim de validar se essas palavras realmente existem. Esse processo de validação torna-se bastante útil, especialmente quando o documento apresenta muitos caracteres inválidos ou palavras com erros gramaticais. As seqüências de

PUC-Rio - Certificação Digital Nº 0310437/CA

caracteres inválidas devem ser eliminadas e as palavras com erros corrigidas. Pode-se aplicar ainda um processo de filtragem naqueles arquivos que possuem formatos de texto específicos, a fim de eliminar as seqüências de controle e/ou formatação de texto.

O dicionário pode também auxiliar a identificação de termos específicos, quando se deseja utilizar palavras pré-definidas no índice, evitando que palavras desconhecidas sejam identificadas (ou seja, evita a utilização de um vocabulário descontrolado). Um simples Analisador Léxico que identifique seqüências de caracteres e forme palavras pode ser utilizado.

A Figura 12 apresenta o trecho de um documento com diversas seqüências de caracteres. As seqüências riscadas são seqüências inválidas, que não devem passar pela fase de identificação de palavras. As demais seqüências vão para a verificação em um dicionário (léxico). As palavras sublinhadas são palavras inexistentes no dicionário, e devem ser corrigidos ou aprendidos. Os caracteres de pontuação são desprezados.

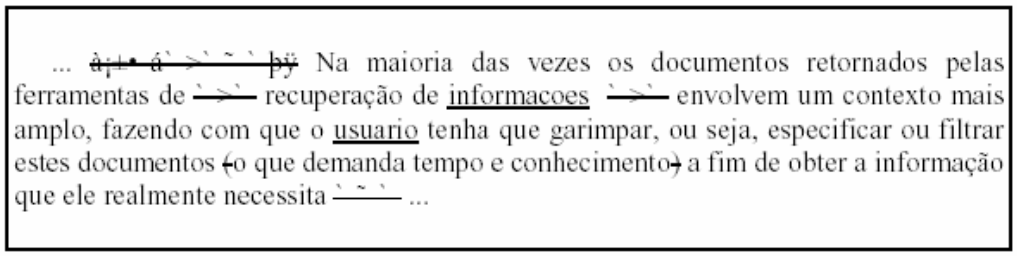

Figura 12 – Indentificação de palavras válidas

Segundo (Spark-Jones, K. e Willet, P., 1997), uma etapa de pré-processamento típica inclui:

- A eliminação de palavras comuns: as palavras comuns (*stop-words*) são elementos de texto que não possuem uma semântica significativa; sua presença não agrega nenhuma indicação do conteúdo ou do assunto do texto correspondente. Normalmente as palavras comuns são constituídas de artigos, preposições, verbos auxiliares, etc, tais como "que", "de/do/das", "o" ou "a". Após sua eliminação obtém-se uma representação reduzida do texto, ainda em formato livre.

PUC-Rio - Certificação Digital Nº 0310437/CA

- A obtenção dos radicais (*stems*): em linguagem natural, diversas palavras que designam variações indicando plural, flexões verbais ou variantes são sintaticamente similares entre si. Por exemplo, as palavras "real", "realidade", "realeza" e "realizado" têm sua semântica relacionada. O objetivo é a obtenção de um elemento único que permita considerar como um único termo, portanto com uma semântica única, estes elementos de texto. Este passo permite uma redução significativa no número de elementos que compõem o texto.

Outra possibilidade de pré-tratamento é a representação em n-gramas do texto (Cavnar, W. B., 1994): constitui-se em uma representação alternativa, onde os termos são obtidos diretamente como sub-cadeias de comprimento *n* das palavras que compõem o texto original. Por exemplo, a partir da palavra "porta" e considerando n = 4, obtém-se as seguintes 4-grams: "_por", "port", "orta" e "orta_", onde "_" é usado para indicar o início ou fim da palavra.

De modo geral, a preocupação das técnicas clássicas presentes na literatura é de reduzir a dimensionalidade do problema, de modo a poder utilizar algoritmos de mineração de dados.

3.2.2. Redução de Dimensionalidade

Um dos maiores problemas de mineração de texto é lidar com espaços de dimensão muito alta se considerado um espaço-vetorial onde cada termo representa uma dimensão, teremos tantas dimensões quanto palavras diferentes. Dessa forma, um dos problemas importantes tratados no pré-processamento dos dados é reduzir o número de termos.

Uma estratégia bastante citada na literatura é a utilização da Lei de Zipf (Zipf, G. K., 1949) e cortes de Luhn (Luhn, H. P., 1958). A técnica tem caráter estatístico e funciona da seguinte forma:

A Lei de Zipf diz que se **f é** a freqüência de ocorrência de qualquer palavra do texto, e **r** a posição de ordenação com relação às outras palavras então produto **f** × **r** é aproximadamente constante. Luhn propôs que, em um gráfico ***f*** versos ***r*** pode-se definir uma limite superior e um limite inferior de corte. A palavras que

PUC-Rio - Certificação Digital Nº 0310437/CA

estiverem fora do intervalo são excluídas da análise. Na **Figura 13** é mostrada uma ilustração desse procedimento.

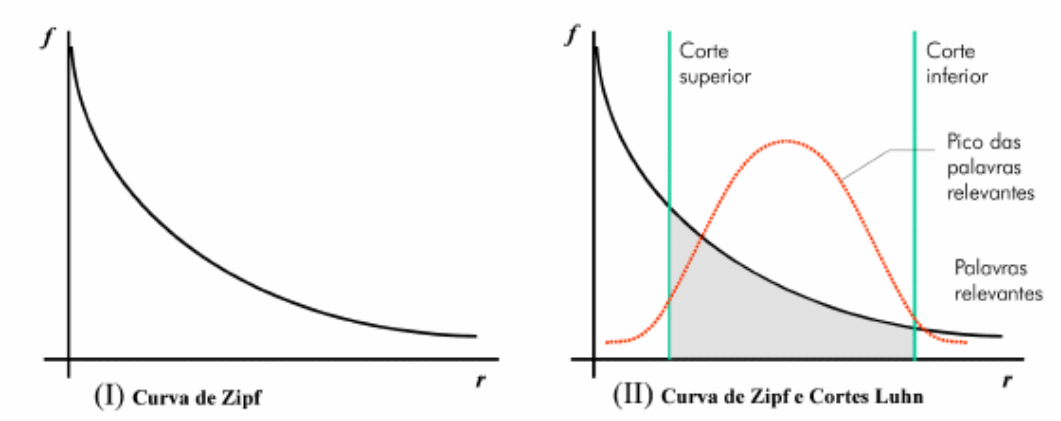

Figura 13 – A curva de Zipf e os cortes de Luhn

PUC-Rio - Certificação Digital Nº 0310437/CA

3.2.3. Remoção de Palavras Não-Discriminantes (*Stop-words*)

Nem todas as palavras dos documentos devem ser adicionadas na estrutura de índice. As palavras que aparecem em todos os documentos ou na maioria deles são um exemplo. Isso porque a utilização de uma palavra com estas características não é capaz de colaborar na seleção de documentos relativos a um assunto específico. (OpenMuscat, 2000)

As preposições são um exemplo deste tipo de palavra, pois são termos que servem para fazer o encadeamento de idéias e palavras, são termos inerentes à linguagem, e não ao conteúdo dos documentos. Normalmente, as palavras que aparecem em muitos documentos não são indexadas pois sua utilização compromete a precisão e a eficiência de um sistema de busca. O prejuízo semântico dessa estratégia é perder a busca exata por compostos como “máquina de lavar”, onde a preposição “de” não pode ser buscada.

Nos sistemas já implementados, foi construída uma estrutura (uma lista) contendo todas as palavras que não devem ser indexadas. A esta estrutura foi atribuído o nome de "*stop-list*", e as palavras presentes nesta lista são conhecidas como *Stop-words*.

O processo de obtenção das *stopwords* pode ser manual, onde o projetista do sistema avalia quais palavras devem ou não ser indexadas (o que varia de

língua para língua, ou até mesmo entre sistemas). Há ainda a possibilidade de se montar esta lista automaticamente, verificando-se quais são as palavras com maior freqüência (que aparecem em mais documentos), selecionando-as como *stop-words*.

Então, após uma palavra ser reconhecida no processo de indexação, sua presença na *Stop-list* é verificada. Caso exista na lista de palavras negativas, ela não é adicionada ao índice.

A Figura 14 abaixo apresenta o documento resultante da etapa anterior, após ser validado por uma *stop-list*. Neste caso a lista de *Stop-words* contém artigos, preposições, conjunções e algumas seqüências de caracteres que não devem ser adicionadas ao índice por possuírem freqüência elevada.

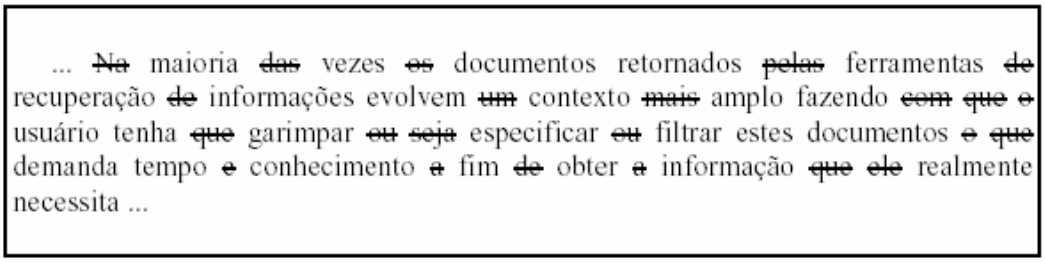

Figura 14 – Identificação de *Stop-Words*

Com estas etapas já é possível criar índices que localizem documentos a partir da comparação direta entre os termos da consulta do usuário e os termos presentes nos documentos. Este é um método ainda ineficiente, e algumas técnicas adicionais podem ser utilizadas a fim de melhorá-lo.

3.3. Indexação

As técnicas de indexação de documentos foram bastante difundidas pela demanda e crescimento da área de Recuperação de Informações (RI). Muitas pessoas acreditam que a área de recuperação automática de informações textuais é uma área nova. Esta idéia talvez tenha surgido com a Web (um dos serviços oferecidos pela Internet), onde milhares de informações, dispostas em forma de páginas (documentos textuais), estão disponíveis. No entanto, segundo (Baeza-

PUC-Rio - Certificação Digital Nº 0310437/CA

Yates, B. e Ribeiro Neto, B., 1999) há aproximadamente 4000 anos já são praticadas técnicas de catalogação manual por índices.

Depois da implantação da Internet em cada vez mais locais, interligando pessoas de diversas partes do mundo, os usuários se deram conta de que as informações disponíveis na rede se encontravam de forma desordenada demandando algum meio de catalogar estas informações automaticamente. Para suprir essa demanda surgiram vários portais de busca: *Altavista*, *Yahoo!*, *Google* entre outros.

Há bastante tempo também, cientistas[3] estudam meios de catalogar informações textuais de forma automática. Com os desdobramentos da informática, catalogar informações se tornou cada vez mais rápido e produtivo. Técnicas bem definidas e testadas foram inseridas na literatura acadêmica, sendo reutilizadas em trabalhos científicos atuais. O *ACM* / *SIGIR*, (*Special Interest Group on Information Retrieval*, da ACM) promove uma conferência internacional de pesquisa e desenvolvimento em Recuperação de Informações que ocorre anualmente, e é um dos meios de divulgação dos estudos na área. Verificando suas publicações (Fox, E. A. et al, 1995), (ACM96b, 1996), constata-se que os métodos atuais buscam aperfeiçoar os métodos mais antigos, permanecendo a metodologia básica.

Dentre os recentes trabalhos, os algoritmos de mineração de textos também reutilizam de técnicas eficientes de indexação para manipulação eficiente dos textos. Como as técnicas de indexação permitem uma busca rápida por palavra-chave em grandes volumes de textos, existe ganho de performance que viabiliza cálculos estatísticos mais sofisticados e, por isso, potencialmente melhores.

No entanto, mineração de textos é um conceito bem mais extenso que busca eficiente por palavras-chave. Por exemplo, uma busca por palavra-chave na Internet retornaria uma lista de páginas que contêm os termos procurados, desconsiderando aspectos semânticos que podem tornar estas ocorrências irrelevantes para o objetivo proposto. Técnicas de mineração de textos promovem análises mais extensas do conteúdo dos documentos, identificando fatos, relações e padrões de forma a obter uma percepção similar àquela tida por um humano lendo o mesmo documento. A informação extraída é geralmente mais relevante, e

[3]Gerard Salton vem trabalhando nessa área desde a década de 60 e já publicou mais de 150 artigos.

PUC-Rio - Certificação Digital Nº 0310437/CA

pode ser usada para diferentes propósitos como categorizar um documento, identificar o significado ou o grupo semântico de expressões dentro do documento, auxiliar a leitura de grandes volumes de texto.

Em um banco de dados textual, os dados não estão distribuídos de forma tabular. Até mesmo porque o texto é uma seqüência de caracteres, não existindo uma pré-especificação de atributos. Não há como saber o que é um nome em um documento, a não ser que se faça uma análise de Linguagem Natural e se descubra o que pode vir a ser um nome.

Logo, para localizar as informações sobre determinada pessoa em um banco de dados textual, seria necessário analisar caractere-por-caractere do texto até que a seqüência de caracteres correspondente ao nome fosse localizada.

Este tipo de análise, comparando todos os caracteres do texto, não é conveniente, é necessário haver alguma forma mais eficiente de acesso aos documentos. Se os documentos textuais possuem um tema, ele pode ser identificado pelas palavras (termos) que esse documento contém, portanto, o termo é o meio de acesso aos documento.

Nas seções seguintes abordaremos as estratégias mais eficientes de busca caractere-por-caractere e como estruturas de dados utilizando essas técnicas chegaram a uma solução bastante eficiente para o problema de encontrar documentos por termos citados dentro deles.

PUC-Rio - Certificação Digital Nº 0310437/CA

3.3.1. Procura Caractere à Caractere

Antes de entrarmos em indexação de textos propriamente dita que carrega consigo estruturas de dados mais complexas, é interessante revisar como o problema de achar uma determinada palavra foi resolvido pela computação.

Uma palavra pode ser entendida como uma cadeia de caracteres específica que se deseja procurar. A informática se apropriou de muitos termos em inglês no seu vocabulário para representar tecnologias específicas, neste caso o termo importado e mais utilizado é s*tring*. Daqui em diante nos referiremos a cadeias de caracteres implementadas no computador como *string*. Sendo assim, procurar por uma *string* no corpo de um texto é preocupação fundamental para qualquer aplicação de mineração de textos.

Inúmeros trabalhos para resolver este problema foram propostos, e eles são comparados normalmente pela complexidade computacional do pior caso e o número de comparações de caracteres feitos. Para o problema de procurar todas as ocorrências de uma dada string de **m** caracteres em um texto de **n** caracteres, o pior caso é expresso pelo número de operações c(n,m). Um primeiro bom resultado estabelecido foi dado por Knuth-Morris-Pratt (KMP) (Knuth, D. E. et al, 1977) como c(n,m) = 2n-m+1. Na mesma época Boyer-Moore (BM) (Boyer, R. S. e Moore, J. S., 1977) desenvolvia um algoritmo que alcançava apenas c(n,m)=6n, porém, ele foi sendo melhorado nos anos seguintes chegando a c(n,m) = (3n-n/m) por (Cole, R., 1990). No entanto uma variante de Boyer-Moore foi desenhada por (Apostolic, A. e Giancarlo, R., 1986) igualando o KMP com c(n,m) = 2n-m+1. Em 1990, Colussi, Gali e Giancarlo (Colussi, L. et al, 1990) criaram um híbrido de KMP e BM para atingir a marca de $7n/6 \leq c(n,m) \leq (4n-m)/3$. Hoje um algoritmo bastante utilizado é o Karp-Rabin (Karp, R. e Rabin, M., 1987) que utiliza a função *hash*. A função *hash* codifica as *strings* em números e com isso ganha uma vantagem significante por utilizar cálculos numéricos.

A maioria desses métodos já estão implementados em pacotes fechados e ferramentas de indexação e busca de forma modular. Não necessitando de re-implementação.

3.3.2. Lista Invertida

Um bom meio de acesso aos documentos são as palavras que ele contém. Para tornar possível o acesso a essas palavras, é preciso colocá-las em uma estrutura auxiliar – o índice, isso porque fica inviável pesquisar todos os textos utilizando consultas booleanas.

Ao final do processo conhecido por Lista-Invertida, os termos resultantes são adicionados a um arquivo de índice cuja estrutura geralmente é baseada em Listas Invertidas (*Inverted Index*). Segundo (Salton, G., 1983), outros tipos de arquivos podem ser utilizados, mas a experiência mostra que este tipo de estrutura é um dos mais eficientes para a indexação de documentos. Na Figura 15 é apresentado um exemplo dessa estrutura. O índice (invertido) contém a lista de todas as palavras indexadas (ex. “diretor”, ‘figura”). Cada palavra é associada à

PUC-Rio - Certificação Digital Nº 0310437/CA

lista dos identificadores dos documentos em que ocorreu (ex. A12, G43). Todas essas listas são armazenadas de forma ordenada para garantir a eficiência dos algoritmos de busca.

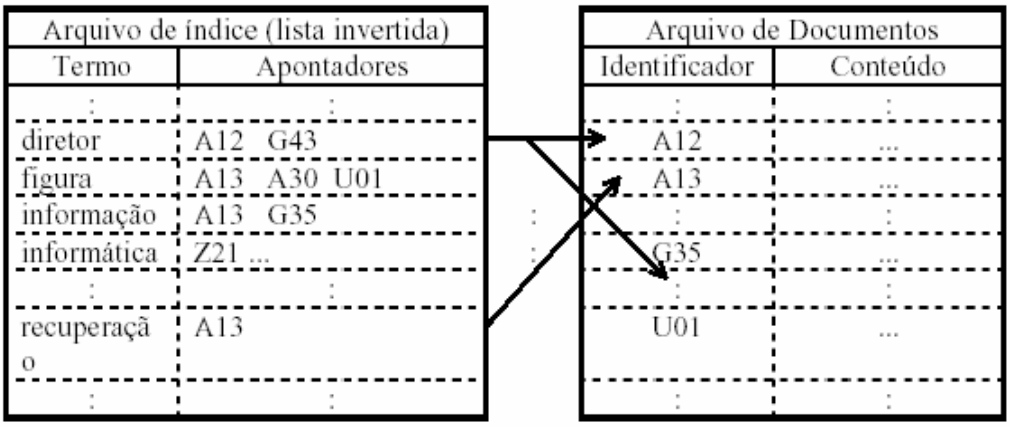

Figura 15 – Estrutura de uma Lista Invertida associada aos documentos indexados.

Basicamente, a estrutura permite que um único termo aponte para vários documentos. Ainda segundo (Salton, G., 1983), as etapas geralmente utilizadas antes da indexação são as de pré-processamento (Seção 3.2):

- Identificação de Palavras
- Remoção de stop-word

Depois que os algoritmos de listas invertidas foram implementados, acessar um documento por uma palavra-chave passou a ser bastante eficiente (logaritmo em relação ao tamanho do banco de dados). Porém, era necessário armazenar todas as ocorrências de palavras, incluindo erros ortográficos e todos os documentos relacionados, o que causava uma explosão de espaço de armazenamento. Em função disso, algumas melhorias foram acrescentadas ao método das listas invertidas e passaram a se chamar de listas invertidas comprimidas (Bell, T. C. et al, 1993); (Bookstein, A. et al, 1992); (Choueka, Y. et al, 1988); (Frakes, W. B. e Baeza-Yates, R., 1992) que aumentaram o desempenho significativamente em termos de requisitos de armazenagem em relação às listas originais.

PUC-Rio - Certificação Digital Nº 0310437/CA

3.3.3. Similaridade

Um sistema de Recuperação de Informações tem como base a seguinte teoria (Salton, G., 1983): perguntas (consulta) são submetidas pelo usuário, perguntas estas baseadas em termos (palavras) que identificam a idéia desejada por este usuário; os documentos são identificados pelos termos que eles contém, portanto, a localização de um documento desejado pelo usuário dá-se a partir da identificação da similaridade entre o(s) termo(s) fornecido(s) pelo usuário e os termos que identificam os documentos contidos na base de dados. A **Figura 16** representa esquematicamente esta teoria:

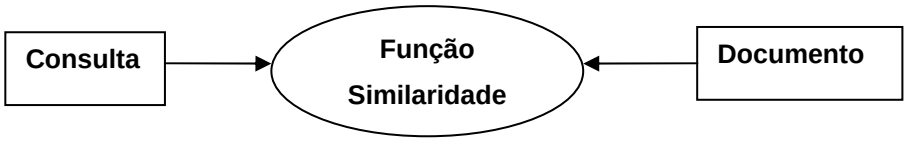

Figura 16 - Função Similaridade

Esta função Similaridade busca identificar uma relação entre os termos da consulta e os termos dos documentos. Teoricamente pode ser feita uma comparação direta entre estes termos, mas na prática é difícil estabelecer esta relação de similaridade entre esses termos devido a alguns problemas descritos nos parágrafos seguintes.

Um destes problemas é analisado por (Chen, H., 1994) em vários de seus trabalhos. O que pode ocorrer é que as palavras utilizadas pelo sistema (palavras contidas nos documentos) sejam diferentes das palavras utilizadas pelo usuário, mesmo que estas palavras (sinônimos) representem a mesma idéia. Esse problema é conhecido por Problema do Vocabulário, e ocorre geralmente quando os usuários desconhecem o sistema, ou possuem um conhecimento superficial dos assuntos que estão tentando localizar.

Há ainda o problema da Busca Incerta (*Search Uncertainly*), ou seja, pode ocorrer que os usuários não saibam quais são as melhores palavras que identificam o assunto que querem localizar. Por conseqüência, acabam não recuperando informações precisas. Este problema também é discutido por (Chen, H., 1994), (Salton, G., 1983) e outros autores.

PUC-Rio - Certificação Digital Nº 0310437/CA

Esses problemas fazem com que sejam recuperados muitos documentos, ou documentos de assuntos variados (pois o termo é muito abrangente), ou ainda, podem não recuperar informação alguma.

É buscando solucionar esses problemas (e alguns outros) que mecanismos de mapeamento entre os diferentes termos similares foram criados. Salton (Salton, G., 1983), cita vários sistemas universitários e comerciais que se utilizam destes mecanismos: STAIRS (IBM), Dialog System (*Lookhead Information Systems*), BRS (*State University of New York*), MEDLARS (*National Library of Medicine*), SMART (*Cornell University*). Em (ACM96a, 1996) são citados mais alguns: WIN (*West Publishing Company*), DOWQUEST (*Dow Jones Newswire*), WAIS, e um muito conhecido, o INQUERY. Nem sempre estes sistemas conseguem satisfazer o usuário, mas foram a base para as técnicas atuais e das que estão por vir. A metodologia básica destes sistemas é discutida a seguir.

PUC-Rio - Certificação Digital Nº 0310437/CA

3.3.4. Processo de Indexação

Um referência internacional sobre o assunto é (Baeza-Yates, B. e Ribeiro Neto, B., 1999). Neste livro encontram-se tanto algoritmos de indexação, formas de implementação em SQL como o processamento de linguagem natural utilizado.

O diagrama da Figura 17 resume o processo total de Indexação. Pode-se ver que os documentos que são fornecidos à ferramenta de indexação passam por uma sucessão de etapas de processamento (em inglês *pipeline*) e ao final é produzido um arquivo de índices que consegue localizar os documentos apresentados.

Em detalhe, a primeira etapa responde por idetificar as palavras, ou as fronteiras das palavras usualmente feita pelo caractere em branco. A segunda elimina, dentre essas palavras, as não-discriminantes (*stop-words*), em seguida, a terceira executa um procedimento de normalização de sufixos, em inglês s*temming* (0). A quarta é reponsável pela detecção de termos compostos, i.e., termos com mais de uma palavra. Finalmente, esses termos, pós-processados, são armazenados na estrutura invertida associados aos documentos origem.

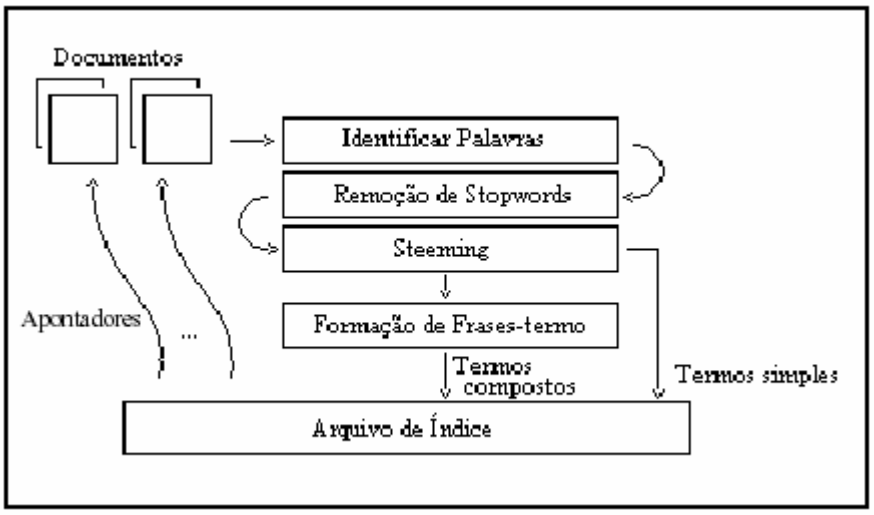

Figura 17 – Sequência do processo de indexação automática.

É importante salientar que esse tipo de indexação automática ainda é bastante simples, não considera a semântica do documento nem a posição sintática das palavras nas orações. Em virtude dessas necessidades surgiram outras formas de indexação mais complexas que usam o mesmo motor de indexação, mas aplicado a uma estrutura de texto enriquecida com metadados. Os metadados são *tags* que marcam informações semânticas ou estruturais do texto. Para o indexador, as *tags* nada mais são do que novas palavras adicionadas ao texto que serão também indexadas (Reis, M., 2005).

3.3.5. Índice do tipo Full-text

A terminologia *full-text* é mais conhecida nos sistemas de recuperação de informação embutidos em pacotes fechados de banco de dados. O nome comumente dado ao índice de bases de dados textuais com textos em linguagem natural é *Full-Text Index*. Muitos SGBDs como Oracle, SQL Server e MySQL já incluem estas funcionalidades prontas. Outros pacotes surgiram especialmente para isso, como o Lucene em C e Java.

Como o objeto principal dessa tese é o pré-processamento dos textos e não a fase de indexação *full-text*, optamos por descrever apenas um deles, com o qual pudemos ter mais contato, o SQL Server 2000. O objetivo da procura Full-text é prover informação relevante de uma coleção de fontes em resposta às necessidades do usuário. Esta necessidade é normalmente expressa por uma

PUC-Rio - Certificação Digital Nº 0310437/CA

consulta que pretende olhar cada registro do banco e procurar por cada palavra requisitada. Uma abordagem simples abriria cada registro e procuraria pela palavra-chave usando um algoritmo de string_matching. No entanto, como vimos, abrir cada documento em tempo de processamento pode ser muito custoso se o volume de documentos for alto.

A solução é fazer parte do trabalho antes da consulta e deixar cálculos pré-armazenados. Isso é feito extraindo informação das palavras em cada documento e armazenando de uma forma que seja fácil de se acessar. Quando a consulta é feita, só é necessário comparar os documentos um com o outro usando um índice invertido (*inverted index*) e escolher os documentos que são mais relevantes.

Para extrair as palavras dos textos, o SQL Server utiliza os chamado w*ord breakers* e *stemmers*. Um *word breaker* é um componente que determina quais são as fronteiras que delimitam uma palavra (é uma solução para o problema teórico definido na sessão 3.10). E os *stemmers* são especializados em uma determinada língua e usam o conhecimento lingüístico com o objetivo de compactar o léxico (flexões verbais são armazenadas como apenas uma palavra).

O índice invertido é uma estrutura de dados com um registro para cada palavra. Nesse registro, existe a informação sobre os documentos em que ela ocorre, o número de ocorrências e a posição em cada um deles. O índice invertido contém ainda algoritmos estatísticos e probabilísticos para computar rapidamente a relevância dos documentos.

A estrutura de dados contendo o índice é armazenada no disco rígido, já que normalmente requer muito espaço. Ela ainda tem funcionalidades de atualização para adicionar um novo documento dentro da base. Vale a pena ressaltar que quando um documento é apagado da base o índice não é alterado e com o tempo ele pode conter muita informação desnecessária a ponto de se ter que construir um novo.

A arquitetura da solução implementada pela Microsoft (Figura 18) contém um serviço separado chamado MSSearch (*Microsoft Search service*) apenas para gerenciar o índice Full-text.

PUC-Rio - Certificação Digital Nº 0310437/CA

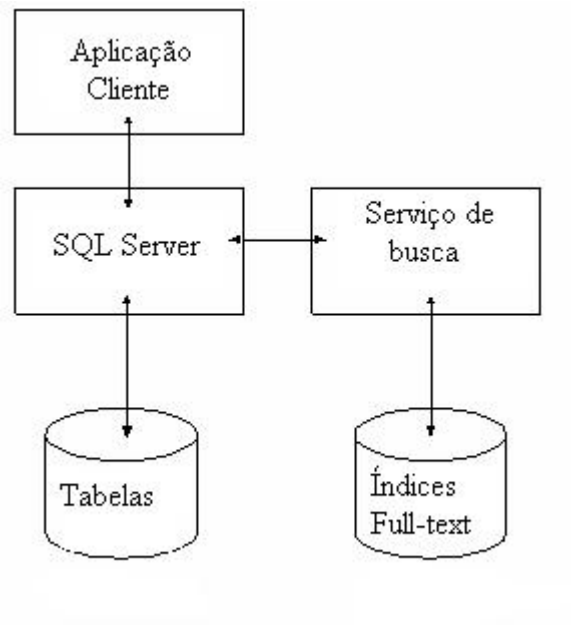

Figura 18 – Arquitetura da Busca tipo Full-text

PUC-Rio - Certificação Digital Nº 0310437/CA

Um exemplo de consulta usando a linguagem SQL (*Structured Query Language*) para pesquisa em banco de dados que tem como objetivo retornar todos os documentos que contenham a *string* "campeonato"

```
SELECT Texto
FROM Documentos
WHERE Texto LIKE '%campeonato%'
```

Essa consulta utiliza o operador de expressões regulares "LIKE" que pode ser aplicado para qualquer campo de caracteres alfa-numéricos. Com o mesmo objetivo que a consulta anterior, porém usando a estrutura de indexação Full-Text

```
SELECT Texto
FROM Documentos
WHERE CONTAINS(Texto, ' "campeonato" ')
```

Se a base de tamanho acima de 10.000 registros a diferença de velocidade na resposta é sensível.

3.3.6. Ordenação

Como em (Gawrysiak, P., 1999), praticamente todos os métodos de ranking caem em dois grupos. O primeiro deles inclui técnicas que exploram a estrutura de apresentação do conteúdo – na Internet, esta estrutura pode ser extraída dos hiperlinks; em bases acadêmicas (artigos e teses), pelas citações. Isso funciona porque a estrutura é criada pelos humanos e contém alguma informação semântica. O segundo grupo compreende ferramentas que lidam com o conteúdo em si operando de forma estatística ou ainda extraindo a estrutura do próprio texto usando técnicas de PLN.

Por exemplo, um mecanismo de recuperação de informação bastante conhecido é o Google. O algoritmo de busca presente no Google chama-se PageRank e está localizado principalmente no primeiro grupo descrito acima. O PageRank procura associar um peso a cada página de conteúdo que está relacionado com o número de páginas que apontam para ela. Estas páginas são chamadas de autoridades. A Figura 8 ilustra como esse problema pode ser complicado

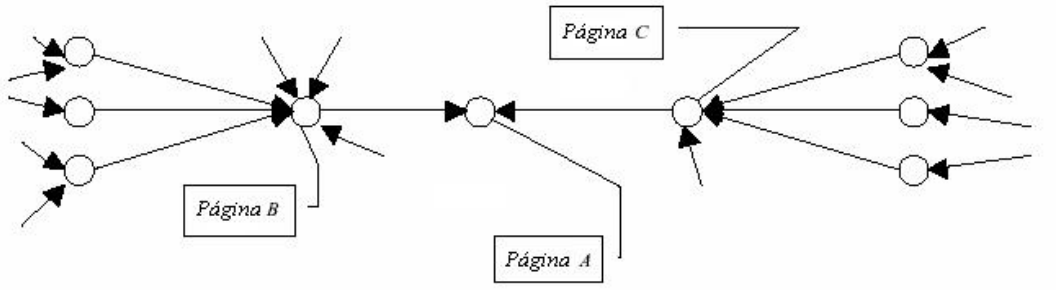

Figura 19 – Representação de uma estrutura de hiperlinks na Internet

Um algoritmo simples que considere apenas a estrutura ajustaria um peso para “Page A” menor do que para “Page B” e “Page C”. No entanto, intuitivamente, “Page A” deve ser a mais importante. Essa é a solução que o algoritmo de PageRank apresenta: é calculada a probabilidade de um navegador aleatório entrar em cada página e entregar o resultado com os valores ordenados da mais provável para a menos provável. A fórmula para este cálculo é dada pela seguinte fórmula

PUC-Rio - Certificação Digital Nº 0310437/CA

$$PR(A) = (1-d) + d(\frac{PR(T_1)}{C(T_1)} + ... + \frac{PR(T_n)}{C(T_n)})$$

onde *A* é um página, $T_{1..n}$ são páginas que contém links para *A*, *PR(A)* é o PageRank da página A, *C(T)* é o número de links de *T* para outras páginas, *d* é a probabilidade de o navegador sair da página.

O algoritmo tal como foi criado pode ser encontrado em (Brin, S. e Page, L., 1998) e (Gibson, D. et al, 1998). Hoje, no entanto, ele já se encontra com alterações, variações e extensões que não são divulgadas.

3.4. Mineração de Dados

A fase de mineração de dados (Goldschmidt, R. e Passos, E., 2005) envolve decidir quais algoritmos serão aplicados aos dados. Nessa fase, pode-se utilizar algoritmos provenientes de diversas áreas de conhecimento, tais como Aprendizado de Máquina, Estatística, Redes Neurais e Banco de Dados, alguns citados no Capítulo 2. Se o objetivo dessa fase é criar um modelo preditivo, então, decidir qual algoritmo é ótimo para o problema que está sendo analisado não é uma tarefa trivial. Esse fato ocorre pois é sabido que nenhum algoritmo é ótimo para todas as aplicações (Schaffer, C., 1994). Muitos estudos empíricos têm sido realizados a fim de relacionar o algoritmo de aprendizado com a natureza do problema a ser resolvido (Michie, D. et al, 1994). Entretanto, encontrar tal relacionamento parece ainda ser um problema em aberto. Uma possível solução, que ainda precisa ser analisada para grandes volumes de dados, é combinar os resultados de vários classificadores em vez de selecionar um único classificador. *Ensembles* (Wolpert, D. H., 1992) têm obtido muito sucesso em combinar o resultado de diferentes sistemas de aprendizado. Entretanto, a utilização de *ensembles* pode dificultar a fase de interpretação dos resultados.

PUC-Rio - Certificação Digital Nº 0310437/CA

3.5.
Análise da Informação

Após a fase de mineração de dados, o processo entra na fase de avaliação e interpretação dos resultados. Essa fase envolve todos os participantes. O analista de dados tenta descobrir se o classificador atingiu as expectativas, avaliando os resultados de acordo com algumas métricas tais como taxa de erro, tempo de CPU e complexidade do modelo. O especialista no domínio irá verificar a compatibilidade dos resultados com o conhecimento disponível do domínio. E, por fim, o usuário é responsável por dar julgamento final sobre a aplicabilidade dos resultados.

PUC-Rio - Certificação Digital Nº 0310437/CA