WEBERT TOMAZ

ALGORITMOS CLÁSSICOS DE BUSCA EM TEXTO E SUA APLICAÇÃO AO ENSINO A DISTÂNCIA: ESTUDO E AVALIAÇÃO

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências da Disciplina Projeto Orientado para obtenção do título de Bacharel em Ciência da Computação

Orientador
Prof. André Luiz Zambalde

LAVRAS
MINAS GERAIS – BRASIL
2001

WEBERT TOMAZ

ALGORITMOS CLÁSSICOS DE BUSCA EM TEXTO E SUA APLICAÇÃO AO ENSINO A DISTÂNCIA: ESTUDO E AVALIAÇÃO

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências da Disciplina Projeto Orientado para obtenção do título de Bacharel em Ciência da Computação

APROVADA em 26 de junho de 2001

_____
Prof. Jones Oliveira de Albuquerque

_____
Prof. André Luiz Zambalde
(Orientador)

LAVRAS
MINAS GERAIS – BRASIL

DEDICATÓRIA

Pelos esforços empreendidos
pelas noites de sono mal dormidas,
dedico a mim mesmo
o presente trabalho.
Webert Tomaz

RESUMO

Impulsionado pelo crescimento da internet, o ensino a distância cria novas oportunidades e novos padrões para a disseminação do conhecimento. Diante dessa realidade é necessário a utilização de ferramentas capazes de automatizar processos repetitivos, principalmente aqueles presentes na conversão dos textos de conteúdo, fundamentais em qualquer curso. A conversão em geral é realizada da seguinte forma: parte-se do modo texto para o modo *html*. O formato *html* introduz nos textos a capacidade de navegação entre seus tópicos. Para que ocorra a navegação é preciso que cada palavra seja previamente marcada como uma âncora e ligada ao seu destino.

É proposta, nesse trabalho, a criação de uma ferramenta de auxílio à geração do glossário de termos técnicos viabilizando as devidas ligações entre o texto e seu significado. A ferramenta buscará no texto as palavras que serão marcadas e ligadas ao glossário. Um breve estudo a cerca de algoritmos de busca em texto é apresentado para que se possa avaliar os benefícios introduzidos por cada algoritmo. A busca em texto é parte importante de muitos problemas que incluem editores de texto, recuperação de dados e manipulação de símbolos. Existem vários estudos sobre algoritmos de busca em texto e sua eficiência do tamanho *m* da palavra, do tamanho *n* do texto e do alfabeto utilizado, assim cada algoritmo possui um desempenho diferente para cada tipo de texto. Também são introduzidos os conceitos de índice e indexação por serem parte importante do estudo de recuperação de informação, quando se trabalha com grandes volumes de dados.

SUMÁRIO

LISTA DE FIGURAS

1. Introdução

Com o advento da computação o ato de escrever sofreu grandes alterações. Antigamente para se editar um livro eram necessárias várias horas de trabalho para se conseguir a forma desejada e a impressão de um livro era um processo repetitivo e demorado. As novas tecnologias eliminaram parte do tempo e dos trabalhos repetitivos o que provocou o aumento da velocidade com que a informação trafega.

Para a editoração de texto existem, hoje, ótimas ferramentas, mas quando se lida com a internet estas ferramentas, em geral, deixam a desejar pois não conseguem tirar proveito máximo das facilidades trazidas pela utilização da navegação, sem algum comprometimento ao documento.

Em especial, o crescimento da internet fez surgir uma nova modalidade de ensino a distância mais interativa do que as formas usuais, onde os alunos dificilmente se comunicavam com o professor. E quando tratamos de cursos via internet precisamos automatizar ao máximo as tarefas para acelerar o processo de disseminação de conhecimento.

Para acelerar tarefas, em computação, o mais comum é substituir as tarefas manuais por rotinas que as implemente. Neste trabalho, é proposta a automatização da criação de glossários, quando da geração de conteúdo para cursos de EAD via internet, por considerar este um dos passos mais repetitivos encontrados.

2. Referencial Teórico

2.1. O Ensino a Distância

2.1.1. Introdução

No ano de 1881, nos Estados Unidos, William Rainey Harper, primeiro reitor e fundador da Universidade de Chicago, ofereceu, com sucesso, um curso de Hebraico por correspondência. Em 1889, o Queen's College, do Canadá, deu início a uma série de cursos a distância, sempre registrando grande procura pelos mesmos, devido principalmente, a seu baixo custo e às grandes distâncias que separavam os centros urbanos daquele país[EDUCAÇÃO, 2001].

O termo, Educação a Distância, pressupõe uma modalidade de ensino aplicada com a distância física entre alunos e professores. Mas nem sempre se pensou assim, o conceito associado à separação física só começou a ser utilizado a partir de 1977.

Em [EDUCAÇÃO, 2001] diversos autores caracterizam o conceito central de Educação a Distância através das definições:

- . Perry & Rumble (1987) - afirmam que "a característica básica da Educação a Distância é o estabelecimento de uma *comunicação de dupla vi*a, na medida em que professor e aluno não se encontram juntos na mesma sala".
- . Keegan (1991) - os elementos fundamentais dos conceitos de Educação a Distância são: *"Separação física entre professor e aluno, que o distingue do presencial;* (...) *utilização de meios técnicos de comunicação, usualmente impressos, para unir o professor ao aluno e transmitir os conteúdos educativos; previsão de uma comunicação-diálogo, e da possibilidade de iniciativas de dupla via; possibilidade de encontros ocasionais com propósitos didáticos e de socialização; e participação de uma forma industrializada de educaçã*o".

Entende-se, de todos estes conceitos, que a Educação a Distância caracteriza-se pela não existência da presença de alunos e professores no mesmo espaço físico, e pode ser viabilizada através de diversas maneiras, bastando apenas existir alguma forma de comunicação entre os mesmos.

2.1.2. Mecanismos de comunicação aplicados à Educação a Distância

A implementação da Educação a Distância foi consagrada pelo chamado "curso por correspondência", ainda muito usado na atualidade.

À medida que novas tecnologias de comunicação surgiram, praticamente todas foram utilizadas na implementação da Educação a Distância. Isso aconteceu com o rádio, a televisão, o videocassete, e mais recentemente com a Internet, como ilustrado na figura 1.

Figura 1: Mecanismos de comunicação aplicados à Educação a Distância

Segundo Ferreira [FERREIRA, 2000], em seu artigo *"A viabilidade da Internet no Ensino a Distânci*a", o ensino por correspondência é estático e geralmente trata-se da simples transferência de informação, onde a única participação do educador é na elaboração do conteúdo que é enviado ao aluno. No rádio e na televisão, por outro lado, existe uma participação muito ativa do educador, que se apresenta de forma dinâmica, passando diretamente seus conhecimentos ao aluno. Mas o aluno tem uma participação muitas vezes de mero espectador.

A Internet oferece tecnologias de comunicação que permitem grande interatividade. Pode-se classificar a Internet como uma mediadora entre as outras tecnologias mais populares de Educação a Distância (o correio e a televisão, por exemplo), permitindo participação ativa tanto de alunos quanto de educadores.

2.1.3. Importância da Educação a Distância

Não se pode pensar na Educação a Distância como um concorrente do tradicional ensino presencial, nem tampouco classificá-la como a melhor solução para os problemas de acesso a educação em nosso país (no Brasil 74 milhões de trabalhadores possuem menos de quatro anos de estudo e cerca de 20% da população são analfabetos [IBGE, 2001]). A Educação a Distância é sim uma alternativa e, algumas vezes, um complemento que viabiliza a disseminação do ensino, mesmo quando a distância física interpõe-se como barreira.

Valer-se de formas alternativas de ensinar, que despertem o gosto pela pesquisa, pelo uso adequado das informações, pela canalização do saber em prol da melhoria da qualidade de vida é um direito do cidadão apoiado pela LDB - Lei de Diretrizes e Bases da Educação (Lei 9.394 de 20/12/96) em seu Art. 80, e do Decreto 2.492 de 10/02/98 que regulamenta o referido artigo de Lei [EDUCAÇÃO, 2001].

2.1.4. Educação a Distância através da Internet

A Internet é uma poderosa mídia de comunicação, que permite um grau de interatividade muito grande com o usuário.

A facilidade de transmissão da informação é outro ponto positivo da Internet, que possibilita a transferência de textos, imagens e sons com relativa velocidade e a baixo custo, já que a internet dispensa os custos das mídias tradicionais (livros, revistas, fitas k-7 e/ou VHS, etc).

A concentração de grandes quantidades de informações sobre os mais variados assuntos também é uma vantagem para o aluno que, além do material oferecido através do *site* de EAD, está separado apenas por alguns *cliques* do *mouse* de toda informação sobre o assunto que esteja publicado na Internet.

Marcelo Ferreira [FERREIRA, 2000] salienta que nos países desenvolvidos, como os Estados Unidos, o acesso a Internet é uma realidade para a maioria da população. Nas escolas americanas, as crianças do ensino básico já utilizam a Internet como principal fonte de pesquisa para seus trabalhos escolares. As universidades e empresas, a muito tempo oferecem os mais variados serviços a seus alunos e clientes através da rede mundial.

Dados retirados de um estudo de 1996, chamado Technology Strategies for Higher Education mostram que a China sozinha produz mais de 100.000 graduados por ano através de EAD, sendo que mais da metade dos 92.000 engenheiros formados adquiriram seus títulos por Educação a Distância [CASTRO, 2001].

No Brasil, a popularização da Educação a Distância através da Internet, depende da popularização da própria Internet, que ainda esbarra em problemas econômicos, falta de investimentos na área de educação e nos problemas gerados pela falta de legislação especifica quando se trata do assunto Internet. Mas existem perspectivas de mudanças.

No campo da educação, o governo brasileiro, apesar de ainda não investir o que é considerado o mínimo necessário, tem se preocupado com esta questão, criando novos currículos de ensino e novas normas que assegurem educação de qualidade no país [EDUCAÇÃO, 2001].

A Educação a Distância via Internet no Brasil já é uma realidade. Existem diversas instituições que oferecem os mais variados cursos através da rede. Mesmo não tendo a popularidade dos países do primeiro mundo, pode-se encontrar na Internet brasileira instituições oferecendo educação virtual, como

UNIVIR - Universidade Virtual (www.univir.br), UNIREDE - Universidade Virtual Pública do Brasil (www.unirede.br), NEAD/UFPR - Núcleo de Educação a Distância (www.nead.ufpr.br), UVB - Universidade Virtual Brasileira (www.uvb.com.br), @groescola - destinado a cursos agropecuários com sede na Universidade Federal de Lavras (www.agroescola.com.br), entre outras.

2.1.5. Os recursos de comunicação disponíveis na Internet

A Internet teve um desenvolvimento relâmpago na última década, passou de um canal quase exclusivo de disseminação de conhecimentos acadêmicos para um meio de comunicação de massa e uma alternativa de distribuição de informação que, a cada dia, conquista novos usuários. Os recursos para comunicação disponíveis na Internet podem ser classificados conforme o tipo de comunicação, que pode ser síncrona ou assíncrona com ilustrado na figura 2. A seguir exemplificamos alguns desses métodos de comunicação.

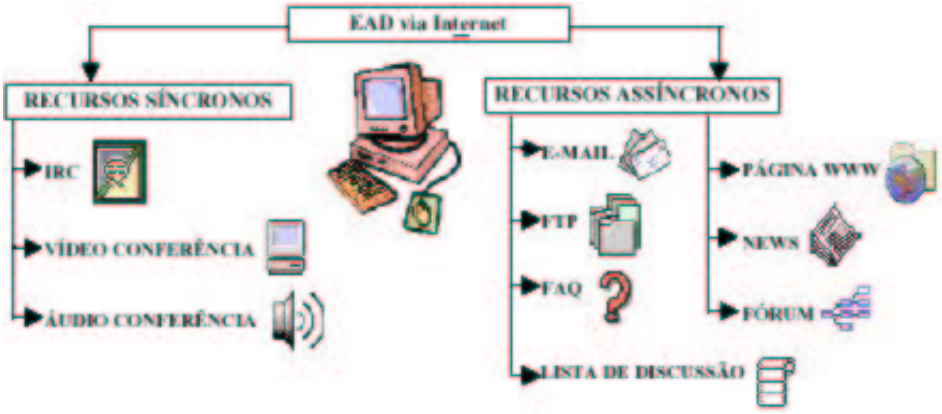

Figura 2: Serviços de comunicação na Internet disponíveis para EAD

2.1.5.1. Comunicação síncrona

É realizada em tempo real, os usuários devem estar ao mesmo tempo conectados na Internet, possibilitando a criação de espaços (salas) virtuais para o encontro de alunos e tutores, onde ocorrerão debates dos diversos assuntos relacionados com o curso em si. Desta categoria fazem parte os seguintes serviços:

- **IRC (Internet Relay Chat)**

  É um sistema que permite aos usuários da Internet se comunicarem em tempo real, também conhecido simplesmente por *cha*t. Existem diversos *softwares* que viabilizam este tipo de comunicação pela Internet, sendo o mais popular o ICQ® [1], distribuído gratuitamente através da própria Internet.

  Estes softwares normalmente possibilitam não só o envio e recebimento de mensagens mas também a transferência de arquivos diversos (som, texto e imagem). Também é possível implementar *chats* na Internet através da *WWW* (*World Wide Web*), que possibilita a utilização deste recurso sem que seja necessária a instalação de softwares adicionais.

  Na Educação a Distância o ICQ® é usado para a conversação escrita entre professores e alunos em tempo real, possibilitando esclarecimento de dúvidas, troca de idéias e troca de materiais de estudo entre os participantes de um mesmo curso. Há necessidade de um agendamento prévio dos encontros, já que os participantes devem estar conectados à Internet ao mesmo tempo.

- **Videoconferência e Audioconferência**

  São sistemas de conferência entre usuários em locais diferentes, usando a infra-estrutura de uma rede de computadores para a transmissão de dados, sinais de áudio e vídeo.

---

[1] ICQ® (*I seek yo*u) é marca registrada da Mirabilis Corporation (www.mirabilis.com).

São as formas mais sofisticadas de comunicação, pois envolve equipamentos especiais como câmeras de vídeo e microfones, permitindo uma interatividade maior entre alunos e professores, que podem expressar melhor suas idéias, bem como, conhecer os integrantes do grupo de uma forma mais pessoal.

Na Educação a Distância, geralmente são usadas para ministrar aulas, onde o professor está em uma sala-estúdio e os alunos espalhados em salas com recursos semelhantes. Assim, o aluno pode ver e/ou ouvir o professor, e vice-versa, mesmo estes estando e em vários lugares diferentes.

Estes são recursos muito utilizados em grandes companhias, no treinamento simultâneo de funcionários localizados em locais diferentes.

2.1.5.2. Comunicação assíncrona

Neste tipo de comunicação, o tempo é flexível e os alunos podem buscar as informações conforme suas disponibilidades. Fazem parte desta categoria:

- **E-mail (Eletronic Mail)**

  É um recurso bastante semelhante ao sistema de correspondência convencional que proporciona o envio e recebimento de mensagens pelo computador, onde cada usuário possui um endereço na Internet.

  O *e-mail* permite que os alunos enviem questionamentos aos professores, e estes podem respondê-los também através de *e-mail*. Os professores e administradores dos cursos a distância podem utilizar o *e-mail* para fazer comunicados gerais aos participantes dos cursos. O *e-mail* também possibilita a transferência de arquivos.

  É o recurso mais utilizado para a interação entre os participantes de cursos via Internet.

- **Lista de discussão**

É um serviço que permite intercâmbio de mensagens entre vários usuários de uma rede de computador.

Funciona como uma extensão do correio eletrônico. As mensagens são enviadas para um endereço único da lista que, posteriormente, retransmitirá as mesmas a todos os usuários que nela se cadastraram.

Recurso também muito utilizado na EAD, cria um canal permanente para alunos e professores comunicarem-se entre si.

- **FAQ (Frequently Asked Questions)**

É uma lista de perguntas e respostas relativas às dúvidas mais comuns sobre determinado assunto. Este conjunto de perguntas e respostas mais freqüentes é organizado em uma página da Internet, com acesso livre, onde os usuários podem consultar a qualquer momento.

O FAQ facilita a comunicação de uma maneira pró-ativa, antecipando as respostas aos questionamentos mais comuns sobre um determinado tema do curso, como também, antecipando informações sobre o funcionamento e estrutura do curso.

- **News**

São grupos de discussão organizados por temas, onde os usuários têm acesso a uma grande quantidade de informação. Servem como meio de distribuição de notícias, semelhante às listas de discussão, com o diferencial de que o usuário deve estar *on-line* no momento de ler as mensagens.

Não é muito comum o uso de *News* na Educação a Distância, porém permitem que alunos e professores tenham acesso a informações atuais relacionadas aos assuntos que estão sendo estudados, além de poderem também divulgar seus trabalhos, para que outras pessoas possam conhecê-los.

- **Fórum de discussão**

É uma ferramenta para grupos de discussão que, ao contrário das listas de discussão onde os usuários se comunicam através de *e-mail*, mantém uma

interface de comunicação na própria página do curso na Internet. As mensagens vão sendo acrescentadas automaticamente na própria página, com livre acesso.

Uma mensagem incluída no fórum abre um espaço de discussão, onde os comentários a respeito dessa primeira mensagem, feitos pelos outros usuários, irão sendo armazenados abaixo da mesma, criando assim uma ligação hierárquica entre as mensagens.

As contribuições ao fórum, podem ser de qualquer pessoa interessada e que tenha acesso à página, não necessariamente que seja um aluno ou professor do curso.

- **FTP (File Transfer Protocol)**

O FTP, ou protocolo de transferência de arquivos, é utilizado tanto para a transferência de arquivos do curso para o aluno, por exemplo apostilas a serem estudadas *off-lin*e; como também para a transferência de arquivos do aluno para o curso, como provas e trabalhos encaminhados para avaliação de aprendizagem.

- **Páginas *Web (Sit*e)**

Este é o recurso mais conhecido da Internet (*WWW*) e consiste em um endereço de Internet (*URL*), com páginas estáticas ou dinâmicas, contendo os mais variados tipos de informação que podem ser acessadas por qualquer pessoa conectada à rede.

As páginas *Web* são a expressão mais comum de distribuição de informações. Nos cursos a distância são usadas como portais de entrada e como via de acesso aos outros recursos utilizados no curso. Podem ainda, disponibilizar informações aos alunos e professores e também a outras pessoas interessadas em participar do curso.

É desta forma de disseminação da informação, páginas *web*, que o presente trabalho trata.

2.2. Recuperação de Informação

Recuperação de informação (RI) é [ZIVIANE, 2001] o processo de recuperar itens de informação armazenados em um meio que possa ser acessado por computador. Um item de informação é geralmente constituído de textos (tais como documentos diversos, páginas *Web*, livros, etc.), embora possa conter outros tipos de dados tais como fotografias, gráficos e figuras. RI lida com representação, armazenamento, organização e acesso a itens de informação. Representação e organização dos itens de informação devem prover ao usuário acesso facilitado à informação de seu interesse.

A área de RI cresceu muito em importância com a introdução da *Web*. No entanto a *Web* introduziu, por si mesma, uma série de problemas. De fato, encontrar informação útil na *Web* é freqüentemente uma tarefa tediosa e difícil. A maior dificuldade é a ausência de um modelo de dados claramente definido. Isto implica que a semântica e a estrutura dos dados é freqüentemente mal definida e de baixa qualidade. Tais dificuldades têm atraído a atenção para a área de RI e suas técnicas como soluções promissoras.

O principal objetivo dos pesquisadores ligados a Recuperação de Informação é estudar algoritmos e estruturas de dados para construir sistemas de RI. Um sistema de RI localiza documentos a partir das necessidades de informação colocadas pelo usuário, através de uma consulta formal. Entretanto, a caracterização da necessidade de informação do usuário é um problema não trivial. Em sistemas reais, essa necessidade de informação deve primeiro ser traduzida em termos de um conjunto de palavras-chaves que são então usadas na formulação de uma consulta. Dada a consulta do usuário, o maior objetivo de um sistema de RI é obter informação que é útil ou relevante para o usuário. A ênfase é na recuperação de informação e não, na simples, recuperação de dados.

Os objetivos específicos da RI é investigar o projeto e implementação de técnicas e ferramentas que possibilitem o desenvolvimento de sistemas de

informação dotados de técnicas eficientes de indexação, técnicas eficazes de ordenação de documentos por ordem de relevância para o usuário, e interfaces que auxiliem o usuário a formular consultas, em especial, para a *WWW*.

O estudo da Recuperação de Informação compreende áreas de indexação (arquivos invertidos e listas invertidas, compressão de textos, compressão de listas invertidas, arquivos de assinaturas, tipos de índices), consultas (consultas lógicas, consultas ordenadas por ordem de relevância, estruturas de acesso ao vocabulário, busca seqüencial no vocabulário, busca exata e aproximada), sistemas paralelos e distribuídos para RI (geração paralela de índices, consulta em bases de dados distribuídas, arquiteturas MIMD) e a busca na *Web*, propriamente dita, (explosão de informação, máquinas de busca, diretórios, linguagens de consulta, meta-ferramentas de busca, sistemas especializados de busca).

Embora os dados possam ser armazenados de diversas maneiras, textos continuam sendo a principal forma de troca de informação. Por exemplo, na Literatura e na Lingüística os dados são armazenados em uma vasta coleção de escritos e dicionários. Na Ciência da Computação, grande quantidade de dados são armazenados em arquivos de textos seqüenciais. Em Biologia Molecular, moléculas biológicas podem ser aproximadas por seqüências textuais de nucleotídeos e aminoácidos. Diante da grande importância dos arquivos textuais é interessante o estudo de algoritmos eficientes para manipulá-los. Em particular, os algoritmos de busca em textos.

Algoritmos de busca de padrões são relevantes em três aspectos diferentes: são componentes básicos utilizados na implementação de comandos existentes em muitos sistemas operacionais; enfatizam métodos de programação que servem de paradigma em outros campos da Computação; desempenham papel importante na Ciência da Computação fornecendo soluções para problemas desafiadores da RI.

2.3. Algoritmos de Busca em Texto

2.3.1. Introdução

Encontrar todas as ocorrências de um padrão em um texto [CORMEN, et al, 2001] é um problema que surge freqüentemente em programas editores de texto. Tipicamente, o texto é um documento que esta sendo editado, e o padrão procurado é uma palavra em particular fornecida pelo usuário. Algoritmos eficientes para este problema podem ajudar imensamente o programa de edição de texto. Algoritmos *String-matching* (casamento de padrões) também são usados, por exemplo, para procurar padrões particulares, seqüências que indicam determinada característica, em seqüências de DNA.

O problema, aparentemente simples, vem sendo estudado desde a década de 70. Vários algoritmos foram propostos ao longo das décadas de 70, 80 e 90, e o problema ainda, hoje, é objeto de pesquisa. Alguns desses algoritmos encontram-se implementados em aplicações de busca em sistemas operacionais, editores de texto, software de manipulação de imagens e de cadeias biológicas.

Várias das soluções propostas são sustentadas em resultados da Teoria dos Autômatos. Nessa área são estudadas propriedades combinatórias de palavras e formalismos para a manipulação de palavras. Em particular, há o formalismo de autômato finito, que serve de base para vários algoritmos de busca de padrão. Esse formalismo originou-se de estudos biológicos de redes de neurônios e circuitos de chaveamento. Atualmente, é utilizado, por exemplo, no desenvolvimento de compiladores, editores de texto e sistemas de manipulação de arquivos.

2.3.2. Estudo Teórico

A formalização do problema *string-matching* ou casamento de padrões ou, ainda como é freqüentemente chamado, busca em texto é dada a seguir:

Assuma que o texto é um vetor *T[1..n]* de tamanho *n* e que o padrão é um vetor *P[1..m]* de tamanho *m*. Assuma que os elementos de *P* e *T* são caracteres pertencentes a um alfabeto finito ∑. Por exemplo podemos ter ∑={0,1} ou ∑={a, b, c..., z}. Os caracteres dos vetores *P* e *T* são, freqüentemente, chamados *strings* de caracteres.

Diz-se que o padrão *P* ocorre com *s* deslocamentos no texto *T*, ou que o padrão *P* ocorre na posição *s+1* no texto *T*, se $0 \leq s \leq n-m$ e $T[s+1..s+m] = P[1..m]$ (isto é, se $T[s+j] = P[j]$, para $1 \leq j \leq m$). Se *P* ocorre com *s* deslocamentos em *T*, chama-se *s* de um deslocamento válido, caso contrário diz-se que *s* é um deslocamento inválido. O problema da busca em texto é encontrar todas as posições em que o padrão *P* ocorre em um dado texto *T*. A figura 3 ilustra esta definição.

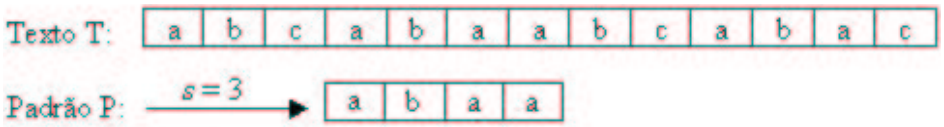

Figura 3: O problema da busca em texto. O exemplo encontra todas as ocorrências do padrão P = abaa no texto T = abcabaabcabac. O padrão aparece somente uma vez no texto, na posição *s* = 3.

Existem diversos algoritmos de busca em texto. Alguns são melhores quando tratam classes especiais de textos e alfabetos, como é o caso de seqüências de DNA onde ∑={C, G, A, T}, outros conseguem melhores resultados quando tratam textos formados por alfabetos maiores, como é o caso da língua portuguesa. Atente para o caso mais comum onde os textos podem conter letras, números, acentuação e caracteres especiais.

De modo geral os estudos [CORMEN, et al, 2001] demonstram que o algoritmo *string-matching* mais simples conhecido como algoritmo força-bruta possui para o pior caso complexidade de tempo $O((n-m+1)m)$. O algoritmo Rabin e Karp também tem complexidade de tempo $O((n-m+1)m)$ para o pior caso, mas trabalha muito melhor na prática. Um outro algoritmo mais eficiente

com complexidade de tempo $O(n+m|\Sigma|)$ começa o processamento pela construção de um autômato finito especialmente desenvolvido para procurar pela ocorrência do padrão *P* no texto. Um similar, mas muito mais hábil, é o algoritmo Knuth-Morris-Pratt (ou simplesmente KMP) com complexidade de tempo $O(n+m)$. E finalmente o algoritmo de Boyer e Moore com seu pior caso com complexidade de tempo igual ao algoritmo Rabin-karp.

2.3.2.1. Notação e Terminologias

A seguir apresentaremos algumas notações comumente usadas no estudo de algoritmos de busca em texto.

Denotaremos por Σ^* o conjunto de todas as palavras de tamanho finito formadas usando os caracteres do alfabeto Σ. A palavra de tamanho zero, palavra vazia, denotada por ε, também pertence a Σ^*. O tamanho da palavra x é denotada por $|x|$. A concatenação de duas palavras *x* e *y*, denotada por *xy*, tem tamanho $|x|+|y|$ e consiste dos caracteres de *x* seguidos pelos caracteres de *y*.

A palavra *w* é prefixo da palavra *x*, quando $w \subset x$ e $x = wy$ para alguma palavra $y \in \Sigma^*$. Note que se $w \subset x$, então $|w| \leq |x|$. Similarmente, uma palavra é sufixo de uma palavra *x*, quando $w \supset x$ e $x = yw$ para alguma palavra $y \in \Sigma^*$. Isto significa que $w \supset x$ e $|w| \leq |x|$. A palavra vazia ε é ambos sufixo e prefixo de todas as palavras.

2.3.2.2. O algoritmo *string-matching* Força-Bruta

O algoritmo força-bruta encontra todas as posições válidas usando um loop que checa a condição P[1..m] = T[s+1..s-m] para cada n-m+1 possíveis valores de s.

```
Algoritmo-string-matching-força-bruta
1 n ← tamanho[T]
2 m ← tamanho[P]
```

```
3 para s ← 0 até n-m faça
4     se P[1..m] = T[s+1..s+m]
5           então imprima "O padrão ocorre na posição" s
```

O procedimento do algoritmo força-bruta pode ser interpretado graficamente como um vetor contendo o padrão procurando pela posição em que todos os seus caracteres correspondem aos caracteres do texto (figura 4). O loop começando na linha 3 considera a possibilidade de cada posição do texto ser a posição inicial em que ocorre o casamento do padrão com o texto. O teste na linha 4 determina se a posição é valida ou não. Neste teste ficou implícito o loop para checar a correspondência entre as posições dos caracteres do padrão com o do texto ou indicar sua falha. A linha 5 imprime cada posição valida de *s*.

O procedimento do algoritmo força-bruta gasta tempo $O((n-m+1)m)$ no pior caso. Por exemplo, considere um texto formado pela palavra a^n (uma palavra formado por *n* a's) e o padrão a^m. Para cada uma das $n-m+1$ possíveis posições de *s*, o loop implícito na linha 4 para comparar os caracteres correspondentes deve ser executado *m* vezes para validar a posição. O pior caso rodando no tempo $O((n-m+1)m)$, é $O(n^2)$ se $m = [n/2]$.

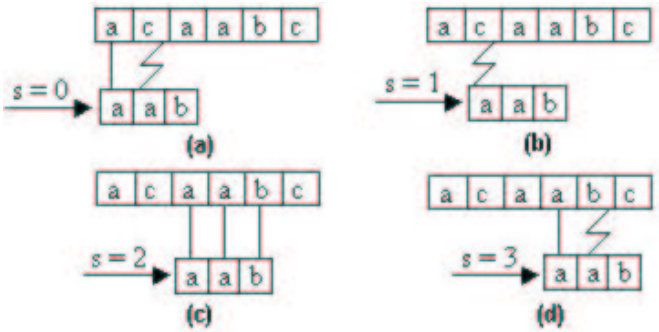

Figura 4: passos do algoritmo força-bruta para um padrão P=aab e um texto T=acaabc. Em cada passo as linhas verticais conectam as correspondências e as linhas quebradas as não correspondências. Um ocorrência do padrão é encontrada em s = 3, mostrada em (c).

Como visto o algoritmo de busca em texto usando força-bruta não é uma solução ótima para o problema. Há algoritmos que rodam no seu pior caso

usando $O(n+m)$. O algoritmo força-bruta é ineficiente porque as informações geradas para uma posição de *s* são totalmente ignoradas para outro valor de *s*. As informações geradas para uma posição de *s* podem ser muito úteis. Por exemplo, se tivessemos *P* = aaab e encontrássemos *s* = 0 válido, então nenhuma das posições 1, 2 ou 3 são válidas pois T[4] = b. Os demais algoritmos de busca em texto fazem uso desse tipo de informação.

2.3.2.3. O algoritmo Rabin-Karp

Rabin e Karp propuseram um algoritmo que funciona bem na prática e generaliza outros algoritmos para problemas relacionados com busca em duas dimensões. O pior caso do algoritmo Rabin-Karp tem complexidade de tempo $O((n-m+1)m)$, mas tem um bom caso médio $O(n+m)$.

Este algoritmo usa noções da teoria elementar dos números para isso assumimos que $\sum = \{0, 1, 2, .., 9\}$ e que cada caracter é um digito decimal. Assuma que cada caracter é um digito em notação radix-*d* (ver [CORMEN, et al, 2001] seção 33.1), onde $d = |\sum|$. Podemos então representar uma palavra de *k* caracteres como um número decimal de tamanho *k*.

O algoritmo que segue ilustra o funcionamento do algoritmo Rabin-Karp com texto de entrada *T*, padrão *P*, o radix *d* (que em geral é $|\sum|$) e um primo *q* para uso.

```
Algoritmo string-matching-Rabin-Karp(T,P,d,q)
1 n ← tamanho[T]
2 m ← tamanho[P]
3 h ← d^(m-1) mod q
4 p ← 0
5 t_o ← 0
6 para i ← 0 até m faça
7     p ← (dp + P[i]) mod q
8     t_o ← (dt_o + T[i]) mod q
9 para s ← 0 até n-m faça
10    se p = t_s então
```

```
11          se P[1..m] = T[s+1..s+m]
12             então "padrão ocorre na posição" s
13          se s<n-m
14             então t_{s+1} ← (d(t_s – T[s+1]h) + T[s+m-1]) mod q
```

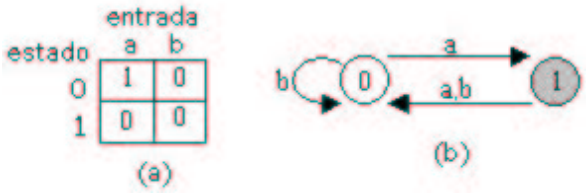

Figura 5: Um autômato finito com conjunto de estados Q = {0, 1}, estado inicial q_0 = 0, e alfabeto de estrada Σ = {a, b}. (a) representação tabular da função de transição δ. (b) diagrama dos estados de transição equivalente.

A função ϕ de um autômato finito *M*, chamada função do estado final, de Σ^* para *Q* tal que $\phi(w)$ é o estado final de *M* após processar a palavra *w*. Isto é, *M* aceita uma palavra *w* se, e somente se, $\phi(w) \in A$. A função ϕ é definida pela relação recursiva $\phi(\varepsilon) = q_0$, $\phi(wa) = \delta(\phi(w),a)$ para $w \in \Sigma^*$, $a \in \Sigma$.

Há um autômato *string-matching* para cada padrão; este autômato deve ser construído para o padrão em um pré-processamento antes de ser usado no algoritmo de busca.

Para exemplificar o funcionamento do algoritmo baseado em autômato finito, apresentaremos um programa simples e eficiente para simulação de um autômato em busca de um padrão *P* de tamanho *m* em um texto *T*[1..*n*]. Para um autômato de busca de em padrão de tamanho *m*, o conjunto de estados Q é {0, 1, .., *m*}, o estado inicial é 0 e o estado aceito é o estado *m*.

```
Algoritmo-de-busca-com-autômato-finito
1 n ← tamanho[T]
2 q ← 0
3 para i ← 1 até n faça
4     q ← δ(q,T[i])
5     se q = m
6           então s ← i-m
```

```
7                   imprima “Padrão ocorre na posição” s
```

Na linha 4 é computada as transições e na linha 5 “$q = m$” se, e somente se, encontrou-se o padrão P no texto T.

O loop simples do algoritmo usando autômato finito implica que está rodando no tempo do tamanho n do texto, isto é $O(n)$. Este tempo não inclui o tempo requerido para computar a função de transição δ. O tempo gasto para computar a função de transição é de $O(m^3|\Sigma|)$ e o tempo total gasto pelo algoritmo de busca usando autômato finito é $O(n+m|\Sigma|)$.

2.3.2.5. O algoritmo Knuth-Morris-Pratt

O algoritmo apresentado por Knuth, Morris e Pratt trabalha em tempo linear. Seu algoritmo realiza sua computação em tempo $O(n+m)$ pois evita o computação da função de transição δ, e faz a busca do padrão usando uma função auxiliar $\pi[1..m]$ que pré-computa o padrão em tempo $O(m)$.

A função prefixo usada no algoritmo KMP para o padrão extrai informações sobre seu próprio padrão. A função prefixo π trabalha da seguinte forma: dado um padrão P = ababaca e um texto T = bacbababababcbab podemos notar que com s = 4 os primeiros q = 5 caracteres casam. Se usarmos nosso conhecimento sobre estes 5 caracteres, podemos deduzir que a posição s+1 (6) é inválida, mas que a posição s+2 (7) é consistente com o que sabemos sobre o texto e é uma posição potencialmente válida. A informação para esta dedução pode ser pré-computada comparando-se o padrão com ele mesmo. Neste caso, vemos que o maior prefixo de P = [ababaca] que também é um sufixo de P_5 é P_3. Dado que q caracteres tem casado sucessivamente com a posição s, a próxima potecial posição é $s' = s + (q - \pi[q])$. Veja a figura 6.

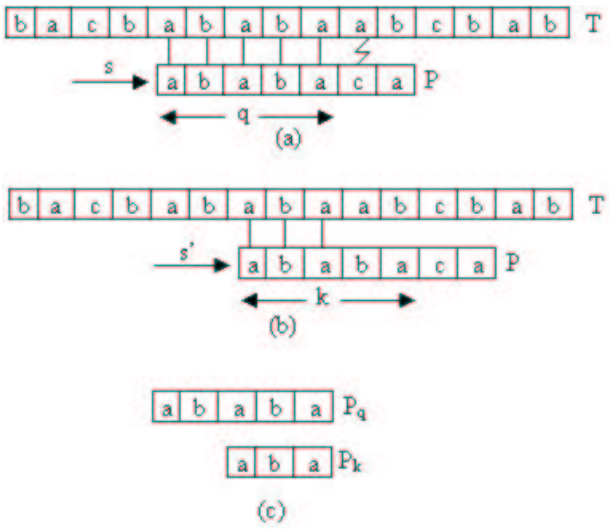

Figura 6: A função prefixo π. Em (a) casamos 5 caracteres do padrão na posição s. Em (b) passamos para a posição s+2, pois já sabemos que a posição s+1 é inválida. Em (c) deduzimos que aba é o maior prefixo que também é sufixo do padrão.

Segue a formalização da pré-computação.

Dado um padrão $P[1..m]$, a *função prefixo* para o padrão P é a função π: $\{1, 2, .., m\} \rightarrow \{0, 1, .., m\text{-}1\}$ tal que $\pi[q] = \max\{k : k<q \text{ e } P_k \supset P_q\}$.

Isto é, $\pi[q]$ é o maior prefixo de P que é um sufixo de P_q.

O algoritmo KMP é dado abaixo para um algoritmo usando busca com autômato finito. O algoritmo KMP chama a função auxiliar compute-função-pefixo para computar π.

```
1 n ← tamanho[T]
2 m ← tamanho[P]
3 π ← compute-função-prefixo(P)
4 q ← 0
5 para i ← 1 até n faça
6     enquanto q>0 e P[q+1]≠T[i] faça
7           q ← π[q]
8     se P[q+1]=T[i]
9           então q ← q+1
10    se q=m
11          então imprima “O padrão ocorre na posição” i-m
```

```
12              q ← π[q]

compute-função-prefixo(P)
1 m ← tamanho[P]
2 π[1] ← 1
3 k ← 0
4 para q ← 2 até m faca
5     enquanto k>0 e P[k+1]≠P[q] faca
6           k ← π[k]
7     if P[k+1]=P[q]
8           então k ← k+1
9     π[q] ← k
10 retorne π
```

O algoritmo KMP tem complexidade de tempo $O(m+n)$. A função prefixo toma tempo $O(m)$, usando o valor de q como uma função potencial mostra-se que o algoritmo KMP toma tempo $O(n)$ no melhor caso.

Comparando-se ao algoritmo usando autômato finito, se usarmos π no lugar de δ reduzimos o tempo de pré-processamento do padrão de $O(m|\Sigma|)$ para $O(m)$, enquanto mantemos o tempo limite de busca em $O(m+n)$.

2.3.2.6. O algoritmo Boyer-Moore

Se o padrão P é relativamente grande e o alfabeto Σ é rasoavelmente grande, como acontece na maioria dos textos comuns, então o algoritmo proposto por Robert S. Boyer e J. Strother Moore é considerado o algoritmo mais eficiente.

O algoritmo é essencialmente o mesmo algoritmo força-bruta realizando sua comparações da direita para a esquerda, mas com duas novas variáveis que são usadas para calcular o novo deslocamento.

```
1 n ← tamanho[T]
2 m ← tamanho[P]
3 λ ← função-ultima-ocorrência(P,m, Σ)
4 γ ← função-bom-sufixo(P, m)
5 s ← 0
```

```
6 equanto s ≤ n-m faça
7     j ← m
8     enquanto j>0 e P[j]=T[s+1] faça
9           j ← j-1
10    se j=0
11          então imprima “O padrão ocorre na posição” s
12                s ← s + γ[0]
13          senão s ← s + max(γ[j], j-λ[T[s+j]])
```

O algoritmo de Boyer-Moore incorpora duas novas heurísticas conhecidas como heurística do mau-caracter e heurística do bom-sufixo. Elas operam independentemente e em paralelo. O algoritmo incrementa *s* na linha 13 com o maior deslocamento, sendo que a heurística do mau-caracter propõe incrementar *s* com $j-\lambda[T[s+j]]$ e a heurística do bom sufixo propõe o incremento de *s* com $\gamma[j]$.

A heurística do mau-caracter

Quando não há um casamento entre os caracteres, a heurística do mau caracter usa informações sobre onde o mau-caracter do texto $T[s+j]$ ocorre no padrão para propor uma nova posição. No melhor caso, não há casamento na primeira comparação ($P[m] \neq T[s+m]$) e mau-caracter não ocorre no padrão. Nesnte caso nós podemos incrementar a posição *s* por *m*, dando o maior incremento possível $s+m$. Se o melhor caso ocorre repetidamente, o algoritmo Boyer-Moore examina somente uma fração $1/m$ do texto. Este melhor caso ilustra o poder do casamento da direita para a esquerda ao invés de da esquerda para a direita.

Em geral a heurística do mau-caracter trabalha da seguinte forma. Suponha que encontramos um não casamento: $P[j] \neq T[s+j]$, onde $1 \leq j \leq m$. Nós tomamos o maior índice k, $1 \leq k \leq m$, tal que $T[s+j]=P[k]$, se algum k existe. Caso contrário tomamos $k = 0$. Consideramos três casos:

- $k = 0$: o mau-caracter $T[s+j]$ não ocorre no padrão, então incrementamos *s* com *j* sem perder nenhuma posição válida;

- $k < j$: a ocorrência do mau-caracter no padrão está a esquerda da posição j, portanto $j - k > 0$ e o padrão deve ser movido $j - k$ caracteres para a direita sem que o mau-caracter case novamente com algum caracter do padrão. Podemos incrementar s com $j - k$ sem perda de alguma posição válida.
- $k < j$: então $j - k < 0$, e a heurística do mau-caracter propõe que se decremente s. Esta recomendação será ignorada, já que a heurística do bom sufixo sempre proporá incrementos positivos.

Chamaremos de função da ultima ocorrência para o padrão a função que calcula a mau-caracter.

```
função-ultima-ocorrência
1 para cada caracter a ∈ ∑ faça
2     λ[a] = 0
3 para j ← 1 até m faça
4     λ[P[j]] ← j
5 retorne λ
```

O tempo de processamento da função ultima-ocorrência é $O(|\sum|+m)$.

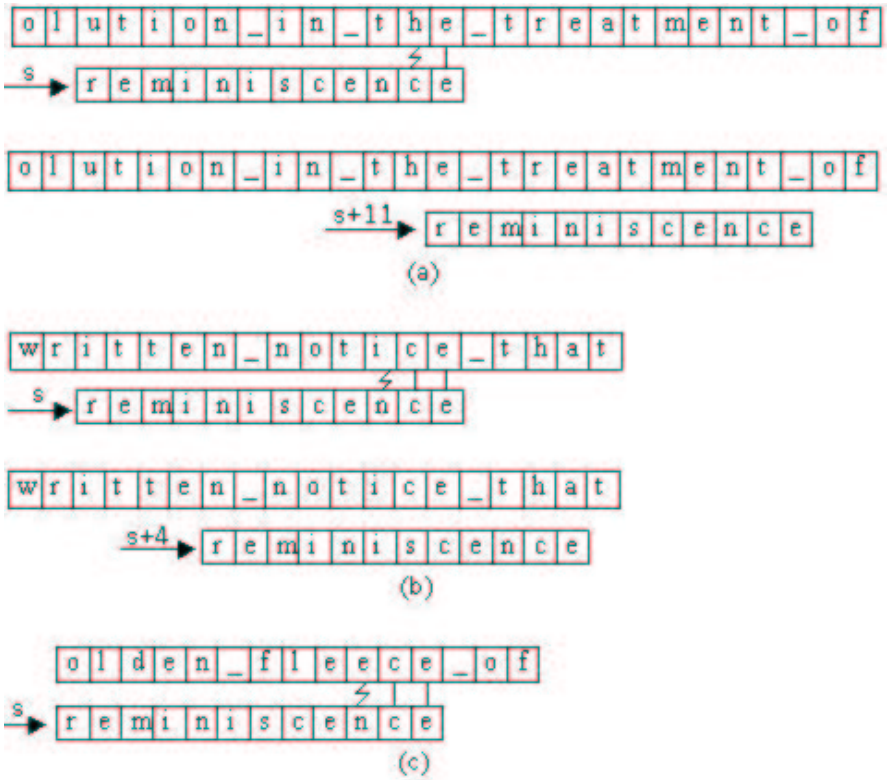

Figura 7: Os casos da heurística do mau-carater. (a) O mau-caracter h não ocorre no padrão e o padrão pode avançar 11 posições até passar pelo mau-caracter. (b) A ocorrência mais a direita do mau-caracter no padrão está na posição k < j, então o padrão avança j – k caracteres. Dado j = 10 e k = 6 para o mau-caracter *i*, o padrão avança 4 posições. (c) A ocorrência mais a direita do mau-caracter está na posição K > j. Neste caso, j = 10 e k = 12 para o mau-caracter *e*. A heurística do mau-caracter propõe um avanço de posição negativo, que é ignorado.

A heurística do bom sufixo

Seja a relação Q ~ R (Q é similar a R) que significa que Q ⊃ R ou R ⊃ Q. A relação é simétrica: Q ~ R se, e somente se, R ~ Q.

Se encontrarmos $P[j] \neq T[s+j]$, onde $j < m$, então a heurística do bom-sufixo nos diz que podemos avançar *s* por:

$$\gamma[j] = m - \max\{k: 0 \leq k \leq m \text{ e } P[j+1..m] \sim P_k\}$$

Isto é, $\gamma[j]$ é o menor valor que podemos avançar s e não casar nenhum caracter do bom-sufixo $T[s+j+1..s+m]$ no novo alinhamento do padrão. A função γ é bem definida para todo j, dado $P[j+1..m] \sim P_0$ para todo j: a palavra vazia é similar a todas as palavras. Chamamos γ de função bom-sufixo para o padrão P.

A função γ é dada por:

$\gamma[j] = m - \max(\{\pi[m]\} \cup \{m - l + \pi'[l]: 1 \leq l \leq m \text{ e } j = m - \pi'[l] \})$

$= \min(\{m - \pi'[m]\} \cup \{ l - \pi'[l]: 1 \leq l \leq m \text{ e } j = m - \pi'[l] \})$

A prova matemática para a expressão acima encontra-se em [CORMEN, et al, 2001].

A função que computa o bom-sufixo tem complexidade de tempo O(m) e é ilustrada abaixo.

```
função-bom-sufixo
1 π ← função-prefixo(P)
2 P' ← reverso(P)
3 π' ← função-prefixo(P')
4 para j ← 0 até m faça
5     γ[j] ← m - π[m]
6 para l ← 1 até m faça
7     j ← m - π'[l]
8     se γ[j] > l - π'[l]
9           então γ[j] ← l - π'[l]
10 retorne γ
```

O pior caso para o algoritmo de Boyer-Moore é $O((n - m + 1)\ m + |\Sigma|)$, dado que a função última-ocorrência toma tempo $O(m+|\Sigma|)$, a função bom-sufixo toma tempo $O(m)$ e o algoritmo Boyer-Moore (como o Rabin-karp) gasta tempo $O(m)$ validando cada posição válida s. Na prática, entretanto, o algoritmo Boyer-Moore é freqüentemente o melhor algoritmo.

3. Definição do problema

Suponha que um professor esteja preparando uma apostila para um curso que será fornecido via internet através de um portal qualquer. Quando finalizada a preparação do material o professor se depara com a seguinte tarefa: criar um glossário de termos de forma que o aluno que estiver consultando o material possa consultar o significado de determinados termos sempre que achar necessário e de uma forma rápida e intuitiva. Para resolver este problema, em geral, o professor cria uma página chamada *glossario.html* e nela insere os termos e seus significados. Mas quando termina esta ultima tarefa surge um novo problema: ligar todos os termos que aparecem no texto ao seu respectivo significado no glossário. Esta ligação recebe o nome de âncora e é a responsável pela inserção da capacidade de navegação entre as páginas. A facilidade de navegação entre páginas é um das principais vantagens introduzidas pela *web*.

Para avaliarmos melhor este problema observe o seguinte exemplo.

Vamos criar uma página *html* contendo um texto, de preferência com alguns termos não usuais, podem ser palavras estrangeiras, nomes técnicos, abreviaturas, etc. Veja o exemplo da figura 8. Agora suponha que queiramos selecionar as palavras da figura 9 para colocá-las no glossário. Selecionados os termos vamos montar o glossário da maneira usual: insere-se o termo e seu significado no glossário, vide exemplo na figura 10, e em seguida procura-se no texto cada ocorrência do termo e ali é criado uma âncora para o seu significado no glossário. Após todo este processo ser realizado sobre o texto, este ficará com a aparência da figura 11.

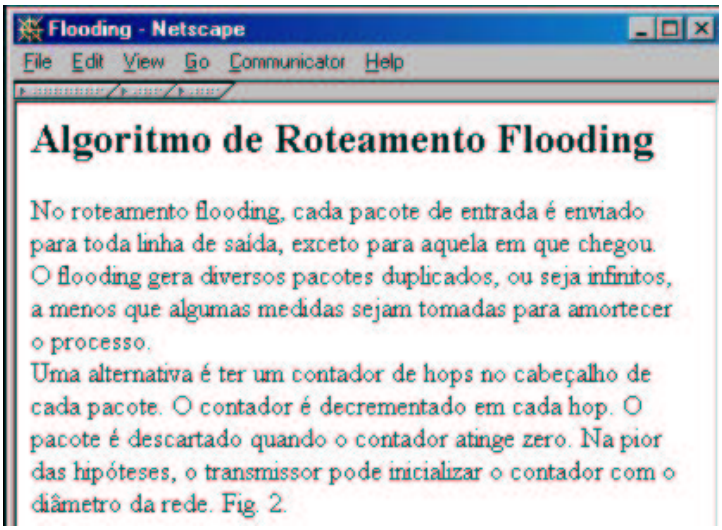

Figura 8: Exemplo de uma página html contendo termos técnicos.

flooding
pacotes
hops
transmissor

Figura 9: Exemplo de palavras a serem colocadas no glossário.

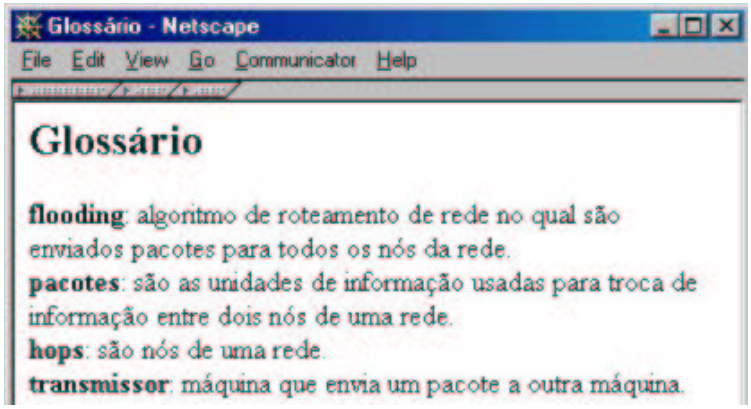

Figura 10: Exemplo de um glossário.

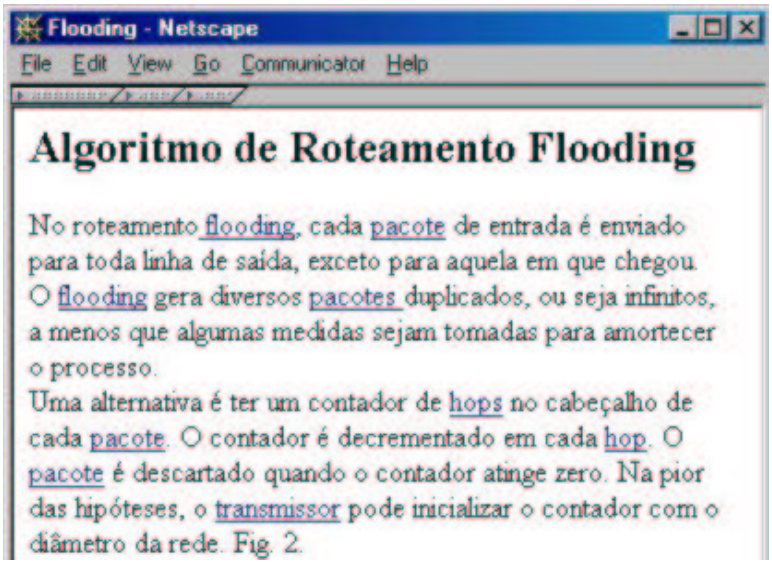

Figura 11: Exemplo de um texto onde todas as ocorrências das palavras técnicas foram ligadas ao seu significado na página do glossário.

Este trabalho pode ser facilitado pelas ferramentas usuais que fazem a substituição de palavras por outras, neste caso é preciso substituir a palavra desejada pela âncora correspondente. Mesmo com esta ajuda o trabalho ainda é repetitivo em especial se o glossário for formado por muitos termos.

4. Metodologia

Para facilitar o trabalho de geração de glossários propomos a construção de uma ferramenta que automatize o processo acima descrito. A ferramenta é uma aplicação prática proveniente do estudo teórico de algoritmos de busca em texto, *string-matching*.

A ferramenta funciona de acordo com esquema apresentado na figura 12: são passados dois arquivos – um arquivo *html* contendo o texto onde será feita a inserção das âncoras, veja o exemplo da figura 8, e outro contendo a lista de palavras que se deseja incluir no glossário, este pode ser tanto um arquivo texto ou um arquivo *html*. A figura 13 apresenta o exemplo de um arquivo texto contendo as palavras a serem incluídas no glossário. A ferramenta lê as palavras do segundo arquivo e cria uma página chamada *glossario.html* a partir do conteúdo lido. Como mostrado na figura 14, o arquivo *glossario.html* possui a lista de palavras e cada uma destas é marcada como destino de uma âncora.

Figura 12: Esquema de funcionamento da ferramenta

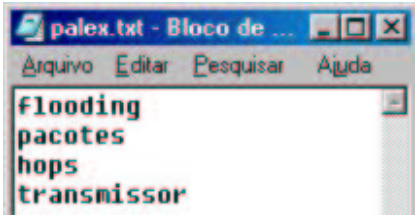

Figura 13: Exemplo de um arquivo texto contendo as palavras a serem incluídas pela ferramenta no glossário

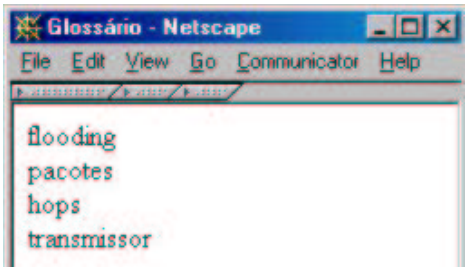

Figura 14: Arquivo glossario.html criado pela ferramenta

O glossário gerado está pronto para receber os conceitos dos termos. Poderia ter sido criado uma base de dados ou, ainda, uma busca numa base disponível na internet para a inserção do significado dos termos. Veja que o procedimento de inserção de significados não foi automatizado, visto que este foge ao nosso contexto.

Após a geração do glossário é realizada a busca de todas as ocorrências das palavras no arquivo texto e a sua substituição pelas âncoras que as ligam ao seu significado no arquivo *glossario.html*. O arquivo depois de processado ficará com a mesma aparência mostrada na figura 11.

O processo apesar de aparentemente simples é por vezes demorado e enfadonho quando feito manualmente.

Para melhor entender os motivos da escolha de um bom algoritmo faremos alguns testes com alguns dos algoritmos apresentados acima para exemplificar as diferenças entre os algoritmos.

Os detalhes da implementação da ferramenta podem ser encontrados na próxima seção desta monografia.

5. Resultados e Discuções

5.1. O gerador de glossário

A seguir apresentaremos o código dos algoritmos utilizados na implementação da ferramenta de geração automática de glossário bem como comentários a respeito dos mesmos.

Antes, porém, devem ser ressaltadas as ferramentas utilizadas para a implementação: os algoritmos estão na linguagem PHP4 [PHP, 2001] que é muito similar a linguagem C/C++ ANSI, mas é mais adequada para se trabalhar com o padrão de arquivos *html* além de ser facilmente alterada para sua utilização remota, neste caso os arquivos devem ser transportados para algum diretório da máquina servidora de páginas porque a linguagem PHP roda na máquina servidora e não na máquina cliente (como é o caso do Java script), no caso o servidor utilizado foi o Apache®[1] 1.3 [APACHE, 2001] rodando na plataforma Windows®[2] mas o algoritmo é independente de plataforma. Para fazer a copia dos arquivos para o servidor consulte a documentação do servidor de páginas para maiores informações. Os arquivos têm de possuir permissão de leitura e escrita, caso não possua o algoritmo tentará alterar estas permissões, caso não consiga o programa abortará com uma mensagem de erro.

Para executar o programa altere o nome dos arquivos do script *criarglossario.php* e o execute o na linha de comando (php *criarglossario.php*) ou crie uma página *html* com formulário de onde será chamado o script acima com variáveis contendo os nomes dos arquivos.

```
criarglossario.php
<?
$filename = "texto.html";
```

[1] Apache® é marca registrada do Apache Group e é distribuído sobre Licensa Publica (www.apache.org)
[2] Windows® é marca registrada da Microsoft Corporation (www.microsoft.com)

```
$filenamesub = "listadepalavras.txt";

include("function.php");
include("substring.php");
include("string.php");
$n = $j -1;
echo "Arquivos criados com sucesso. Renomeie o arquivo
newhtml$n.html para $filename";

//Para substituir o arquivo automaticamente insira o código
//delete($filename);
//copy ("newhtml$n",$filename);
?>
```

Note que o script chama outros scripts o primeiro *function.php* contém as funções usadas pelos demais scripts, como a função para abrir arquivos que tenta alterar as permissões dos arquivos caso estes estejam sem permissão para leitura e/ou escrita e as funções de busca em texto usadas nos testes.

Em seguida é feita a chamada do script *substring.php* que faz o processamento do arquivo que contém a lista de palavras e a construção do arquivo *glossario.html*. Uma observação deve ser feita a respeito do arquivo que contém a lista de palavras do glossário: cada palavra deve estar em uma linha, mas alterações simples podem ser feitas de modo a usar algum separador. Por exemplo o espaço poderia ser utilizado mas isto restringiria o algoritmo a palavras simples e quando nos referimos a palavras, entenda qualquer formação: palavra simples, frase, siglas, palavras compostas separadas por espaço, etc.

```
substring.php
<?
$filesizesub = filesize($filenamesub) - 1;

//extração da string do arquivo
$fileopensub = openfile($filenamesub,"r+");
while ( !feof($fileopensub) ){

      $filesemtagsub.=fgetss($fileopensub,$filesizesub);
      }
fclose($fileopensub);

//criando arquivo temporário
$fileopentmp=openfile("tmpgl","w+");
```

```
//escrevendo no arquivo temporário
$write = fputs ($fileopentmp, $filesemtagsub);
fclose($fileopentmp);

//array com as palavras do glossário.
$filesubs = file("tmpgl");

//script que cria o glossário
include("glossario.php");
?>
```

No algoritmo acima é incluído em seu processamento um script a parte que é responsável pela criação do arquivo *glossario.html* utilizando-se das palavras extraídas do arquivo que contém a lista de palavras. Pode-se querer incluir as palavras do glossário diretamente no fim da página de texto. Isto é aconselhável quando o todo o texto se encontra numa única página. Neste caso no script que substitui a palavra pela âncora retire da ancora o nome do arquivo do glossário, no caso *glossario.html* .

```
glossario.php
<?
//criando o arquivo glossario.html
$fileopeng=openfile("glossario.html","w+");

//escrevendo no arquivo glossario.html
fputs ($fileopeng,"<html><title>Gloss&aacute;rio</title>");
for ($i=0 ; $i<(sizeof($filesubs)) ; $i++){
      $write = fputs ($fileopeng, "<a NAME=\"$i\">
                        </a>$filesubs[$i]<br>");
}
fputs ($fileopeng,"</html>");
fclose($fileopeng);
?>
```

Abaixo apresentamos o script *string.php* responsável pela busca das palavras no arquivo de entrada e subseqüente substituição pela devida âncora para o termo no arquivo *glossario.html*.

```
string.php
<?
$substring = $filesubs;
$j = 0;
do{
```

```
   $filesizestr = filesize($filename) - 1;

   //extraindo a string do arquivo
   $fileopenstr = openfile($filename,"r+");

   $filetagstr ='';
   while ( !feof($fileopenstr) ) {
        $filetagstr .= fgets($fileopenstr,$filesizestr);
      }
   fclose($fileopenstr);

   //o codigo abaixo realiza a busca
   $newhtml = openfile("newhtml$j.html","w+");

   $posicaostr = 0;
   $busca = 1;
   while ($busca != 0){
        $novastring ='';
        for ($i=$posicaostr ;$i<($filesizestr+1); $i++){
            $novastring .= $filetagstr[$i];
        }

        //posição em que substring ocorre na string
        $busca = strpos ($novastring, $substring[$j]);

        if ($busca){
            //sustituicao no arquivo html
           for ($i=$posição ;$i<($posicao+$busca-1); $i++){
                  fputs ($newhtml, $novastring[$i]);
           }
           fputs ($newhtml, "<a href=\"glossario.html#$j\">
                  $substring[$j]</a>");

           $posicaostr+=($busca-1+ strlen($substring[$j]));
         }
         else{
               for ($i=$posicaostr;($i<$filesizestr);$i++){
                  fputs ($newhtml, $filetagstr[$i]);
              }
         }
   }
   fclose($newhtml);
   $filename="newhtml$j.html";
   $j++; //muda de palavra

}while ($j<sizeof($substring));
?>
```

5.2. Testes algoritmicos

Para demonstrar a utilização dos algoritmos em textos da língua portuguesa, $\sum = 26$, fizemos testes com o algoritmo força-bruta, o algoritmo KMP e com uma função nativa da linguagem em alguns textos e para exemplificar os resultados abaixo segue o resultado de dois destes testes.

Tempo médio dos algoritmos de Força-bruta, KMP e função nativa da linguagem.

Algoritmo	**Tempo de processamento**	
	Arquivo 1	**Arquivo 2**
Força-bruta	1.085.973	1.446.689
KMP	653.954	1.072.588
Função Nativa	720.347	923.084

Figura 15: Quadro comparativo dos tempos médios dos algoritmos de Força-bruta, KMP e função nativa da linguagem.

Como se pode observar pelo tempo de processamento mostrado acima o segundo arquivo e maior que o primeiro e note que o algoritmo KMP no primeiro arquivo obteve tempo de processamento menor do que a função nativa, que provavelmente implementa o algoritmo Boyer-Moore visto que a maioria das linguagens faz isto, mas no segundo arquivo obteve tempo maior, o que confirma a teoria apresentada sobre os algoritmos: o algoritmo KMP tende a ser pior quando não há prefixos e sufixos iguais no padrão e o texto e é formado por um alfabeto grande, nestes casos o algoritmo KMP tende ao algoritmo força-bruta.

5.3. Geração de âncoras

As âncoras por introduzirem a capacidade de navegação nos documentos podem ser consideradas uma das principais inovações trazida pela *web*, pois até então os documentos eram muito estáticos, ou seja, não apresentavam muita mobilidade ao leitor.

Apesar de auxiliar a leitura e a busca de informação o processo de geração de ancoras para textos em *html* pode ser bastante repetitiva e trabalhosa, mas existem ferramentas que podem nos auxiliar. Uma das ferramentas mais usadas é a ferramenta de substituição que vasculha todo o texto e substitui cada termo por um ancora. A desvantagem deste método é: se o número de palavras do glossário for extenso caímos no problema da repetição. Para isso foi proposta e implementada uma ferramenta de geração automática de glossário onde o problema de repetição de processos foi totalmente eliminado.

5.4. Considerações finais

O Ensino a Distância através da Internet oferece grandes facilidades, dentre as quais estão o custo reduzido para o aluno e a flexibilidade de tempo. Destaca-se também a interatividade entre professor-aluno e aluno-aluno, o que facilita a troca de experiências e proporciona um ambiente colaborativo mesmo sem a presença física dos integrantes do grupo. Porém nem tudo são vantagens no EAD via Internet. Esta modalidade exige disponibilidade de equipamentos e tecnologias que estão distantes da maioria da população brasileira.

Está claro que o Ensino a Distância via internet tem inúmeras vantagens e desvantagens. Além do problema citado acima, quando se trabalha com a internet, há problemas relacionados com trabalhos repetitivos que foi o alvo principal deste trabalho, o qual propôs a implementação de uma ferramenta de auxílio à geração de glossário por considerar este um dos trabalhos repetitivos mais comuns na geração de conteúdo para algum curso a distância.

Na EAD via *web* existem pontos fortes e também pontos negativos a serem considerados, mas de forma alguma se pode desprezar a disseminação de informações através de EAD via Internet, que pela sua própria característica difusora de conhecimento, apresenta-se como o principal portal de acesso ao mundo atual, pois acompanha a velocidade de seu progresso.

Referências Bibliográficas

[1] EDUCAÇÃO a Distância via Internet: uma introdução aos seus recursos. Artigo submetido a revista de iniciação científica da Sociedade Brasileira de Computação. Maio de 2001.

[2] FERREIRA, M. "A viabilidade da utilização da Internet como ferramenta no Ensino a Distância" Ensino a Distância na Internet, 1998, http://www.geocities.com/WallStreet/7939/index.html. (17/08/2000).

[3] IBGE (Rio de Janeiro, RJ). http://www.ibge.gov.br. (Junho, 2001).

[4] LOBO NETO, F.J.S. "Educação a distância: regulamentação, condições de êxito e perspectivas" 1998, http://www.intelecto.net/ead/lobo1.htm (09/09/2000).

[5] ZIVIANE, N. Página da disciplina “Recuperação de Informação A”. UFMG, MG. Abril, 2001. http://www.dcc.ufmg.br (Abril, 2001).

[6] CORMEN, T.; LEISERSON, C.; e RIVEST, R. 1997. “Introduction To Algoritms”, em String-Matching, p. 853-885.

[7] BRASIL. Governo Federal. Lei de Diretrizes e Bases da Educação Nacional, Lei nº 9394/96. Brasília: Governo Federal, 1996.

[8] CASTRO, C.L. de. Desenvolvimento de um servidor de Avaliações Web para redes de Aprendizado. DCC-UFLA. Junho de 2001. (Monografia – Bacharelado em Ciência da Computação).

[9] BAEZA-YATES, R.; RIBEIRO-NETO, B. Modern Information Retrieval, Addison-Wesley, 1999. http://www.dcc.uchile.cl/Erbaeza/index.html. (Maio, 2001).

[10] CARVALHO, P.S. de; OLIVEIRA, D. de; GUARACY, A. et al. Um Estudo Teórico e Experimental em Algoritmos Clássicos de Busca em Texto. INFOCOMP Revista de Computação da UFLA, Ano II – V1, Lavras, novembro de 2000.

[11] NEUBERT, M. S. Recuperação Aproximada de Informações em Textos Comprimidos e Indexados. DCC-UFMG. Dezembro de 2000. (Dissertação – Mestrado em Ciência da Computação).

[12] FISCHER, H.G. PHP Guia de Consulta Rápida. Novatec Editora Ltda. 2001. 128p.

[13] TOMAZ, W. Legislação de Informática. Lavras, novembro de 2000. Artigo apresentado na III Semana de Ciência da Computação da Universidade Federal de Lavras.

[14] GASTON, H.; BAEZA-YATES, R. Handbook of Algorithms and Data Structures. Gonnet Informatik, ETH Zürich. Dept. of Computer Science, Univ. of Chile. http://www.dcc.uchile.cl/~rbaeza/handbook/hbook.html (Maio, 2001).

[15] SOUZA, R.N.P.M. de. Projeto de Iniciação Científica: Autômatos e Algoritmos de Busca de Padrões. DCC-USP. Dezembro de 2000. http://www.linux.ime.usp.br/~rsouza/pmatching.html. (Abril, 2001). (Monografia - Bacharelado em Ciência da Computação).

[16] SILVA, A.H.R.; AYALA-RINCÓNn, M. Departamento de Matemática Universidade de Brasília. Abril, 2001. http://www.mat.unb.br/~ayala/TCgroup/PatternMatching/kmpbm.html . (Abril, 2001).

[17] PHP. Site oficial do projeto PHP. http://www.php.net e site oficial em português http://www.br.php.net. Junho de 2001

[18] APACHE. Site oficial do servidor web Apache. http://www.apache.org. Junho de 2001.

Apêndice A: Recuperação Aproximada de Informação em Textos Comprimidos e Indexados

O volume de dados armazenados na forma de textos tem acompanhado o rápido crescimento das bibliotecas digitais modernas, como coleções médicas e jurídicas, enciclopédias e a *World Wide Web (WWW*). Para se recuperar de forma eficiente a informação armazenada nesse tipo de texto, várias técnicas especializadas de indexação foram propostas na literatura. Entre elas encontram-se as árvores PATRICIA, arranjos de sufixos, arranjos PAT, arquivos de assinatura e arquivos invertidos, cada uma apresentando pontos fortes e fracos.

Entre tantas técnicas, porém os arquivos invertidos têm sido tradicionalmente a estrutura mais utilizada. Essa popularidade se deve ao seu custo de construção (proporcional ao tamanho do texto) e a simplicidade de sua estrutura. Além disso, a técnica de busca utilizada por arquivos invertidos é bastante rápida, pois se baseia no vocabulário (conjunto de termos distintos da coleção), que geralmente pode ser acomodado na memória principal.

Entretanto, apesar de sua estrutura simples, a construção de arquivos invertidos para grandes coleções textuais não é uma tarefa trivial. As listas de documentos podem ser muito longas e exceder a memória principal disponível. Torna-se, portanto, necessário o uso de espaço em disco para a indexação e o projeto de algoritmos eficientes passa a ser mais complexo. O espaço ocupado pelo índice pode ser bastante significativo (até 100% do tamanho do texto original).

A técnica de compressão pode ser aplicada a arquivos invertidos, reduzindo significativamente o tamanho do índice. A técnica mais comum é ordenar os números dos documentos de cada lista em ordem crescente e substituir cada um pela diferença entre ele e seu antecessor. Essa nova seqüência de valores pode ser eficientemente codificada através de métodos que

representam números pequenos por poucos bits. Para coleções típicas, o espaço ocupado pelo índice invertido comprimido representa entre 6 e 10% do tamanho do texto original.

Índices também podem ser usados para permitir uma busca mais rápida de padrões de caracteres diretamente dentro do texto, seja informando a posição exata da ocorrência ou o endereço do bloco de texto no qual ele ocorre. Nestes casos, as listas invertidas não contêm mais identificadores de documentos, mas posições físicas das ocorrências do padrão dentro do texto.

O grande porte dos bancos de dados textuais pode ainda representar, juntamente com o índice, um volume muito grande de informação. A compressão do texto pode ser implementada como uma camada transparente dentro de um sistema completo de recuperação de informação, possibilitando tanto a redução dos requisitos de espaço como a diminuição dos tempos de busca, em uma situação de duplo ganho.

Finalmente, um banco de dados com grandes volumes de texto apresenta uma alta probabilidade de conter erros, seja por digitação incorreta, erros no reconhecimento ótico de caracteres ou ainda falhas de transmissão. Outra possível situação é o desconhecimento da grafia exata do termo a ser procurado. Outra situação ocorre devido às várias flexões que uma palavra pode conter. Em cenários como esses, é desejável que o sistema se informação permita buscas aproximadas, isto é, que se recuperem padrões com até certo número de erros em relação ao termo pesquisado.

Um bom sistema de recuperação de informações deve realizar a indexação eficiente de grandes bases textuais, possibilitar a compressão tanto do índice como do próprio texto e a realização de buscas em dois níveis (acesso tanto aos índices como partes do texto), além do reconhecimento automático de alguns tipos de arquivos texto como HTML, PostScript, PDF, entre outros.