Criação de um Arquivo Invertido para a Recuperação de Informação em Grandes Volumes de Texto

Rafael Rodrigo da Silva[1], Wender Magno Cota[1]

[1]Universidade Presidente Antônio Carlos (UNIPAC)
Faculdade Regional de Ciências Exatas e Sociais de Barbacena (FACEC)
Campus Magnus – Rua Palma Bageto Viol, s/nº – Campolide
CEP: 36.200-108 – Barbacena – MG – Brasil

rafaeleafb@yahoo.com.br, wmcota@terra.com.br

Resumo. *Este trabalho apresenta uma técnica usada na recuperação de informação, chamada Arquivo Invertido, a qual é usada para que se possa criar um índice contendo as palavras de uma determinada coleção de documentos do tipo texto, a fim de possibilitar uma recuperação mais rápida e eficiente de informações nessa coleção. Assim, o objetivo deste trabalho é apresentar e implementar um algoritmo que gera um arquivo invertido, a fim de que este possa ser utilizado na maximização da eficiência no processo de recuperação de informação.*

Palavras-chave: *arquivo invertido, recuperação de informação.*

1. Introdução

"Por aproximadamente 4000 anos o homem tem organizado informações para serem recuperadas e usadas posteriormente. Por exemplo, uma tabela de conteúdo de um livro. Como o acervo de livros cresceu, uma estrutura teve de ser criada para acessar de forma mais rápida as informações armazenadas nos livros" [1].

Assim como cresce o acervo de livros, cresce também proporcionalmente a quantidade de informações disponíveis, principalmente em forma de texto, na web. Bibliotecas digitais estão cada vez mais em uso, assim como *sites*, que crescem em número e quantidade de informação textuais, e este crescimento causa uma certa sobrecarga se não for gerenciado.

Da mesma maneira que foi necessária a criação de uma estrutura para acessar de forma mais rápida as informações armazenadas em livros, faz-se também necessária a criação de alguma estrutura para que se possa recuperar eficientemente informações textuais armazenadas de forma digital, o que leva ao surgimento de diversas técnicas dedicadas à Recuperação de Informação.

"Uma estrutura antiga e muito utilizada para a rápida Recuperação de Informação, é uma coleção de palavras selecionadas, ou conceitos, com os quais estão ligados, através de apontadores, às informações relacionadas – o chamado índice" [2].

Na literatura, é possível encontrar várias técnicas que podem ser utilizadas para a criação destes índices, como pode ser visto em [3], dentre as quais, encontra-se a técnica de indexação denominada Arquivo Invertido (objeto de estudo do presente trabalho), a

qual “tem sido tradicionalmente a técnica de indexação mais utilizada, devido ao seu custo (proporcional ao tamanho do texto) e à simplicidade de sua estrutura”[3] .

Assim sendo, esta técnica será discutida e implementada neste trabalho, visto a necessidade de se utilizar de uma estrutura para recuperação de informação, que torne este processo mais rápido e eficaz. Na seção 2, será apresentado o conceito de um arquivo invertido, bem como a forma que este pode ser implementado.Na seção 3, será apresentado o algoritmo destinado à geração de arquivos invertidos, destinados à recuperação eficiente de informações.Na seção 4 serão apresentados dois tipos de consultas que podem ser utilizadas no processo de recuperação de informação a partir do arquivo invertido gerado. Finalmente, na seção 5, serão apresentadas as considerações finais, bem como as propostas para trabalhos futuros.

2. Arquivos Invertidos

Como citado na seção 1, nas aplicações envolvendo textos (bibliotecas digitais, manuais, *sites*, dentre outros), o método mais ajustável para a recuperação rápida e eficiente da informação tem sido o arquivo invertido [5].

Um arquivo invertido é um arquivo de índices, que contém duas partes: um **Vocabulário** ou **Dicionário,** contendo um conjunto de palavras distintas do texto, e uma **Lista de Ocorrências,** indicando para cada termo do vocabulário, em quais documentos de uma determinada coleção este termo (palavra) ocorre, indicando, ainda, qual a freqüência do termo em cada um destes documentos [5]. Para um melhor entendimento do que vem a ser um arquivo invertido, será utilizada, como exemplo, a coleção de documentos textuais apresentada a seguir:

Documento	Texto
Doc1	a casa da mãe
Doc2	casa da mãe
Doc3	mãe da casa
Doc4	a da casa da mãe
Doc5	a casa mãe da casa a mãe

TABELA 2.1. Exemplo de uma coleção textual, contendo os documentos e seus conteúdos.

Sendo a Tabela 2.1, a representação de uma coleção de documentos com seus respectivos conteúdos, pode-se criar índices para uma eficiente recuperação de informação desses documentos, usando para a indexação, a técnica de arquivos invertidos. A Figura 2.1 representa esta técnica, destacando cada palavra existente na coleção de documentos da Tabela 2.1, seguida de seu respectivo número de ocorrência em cada documento.

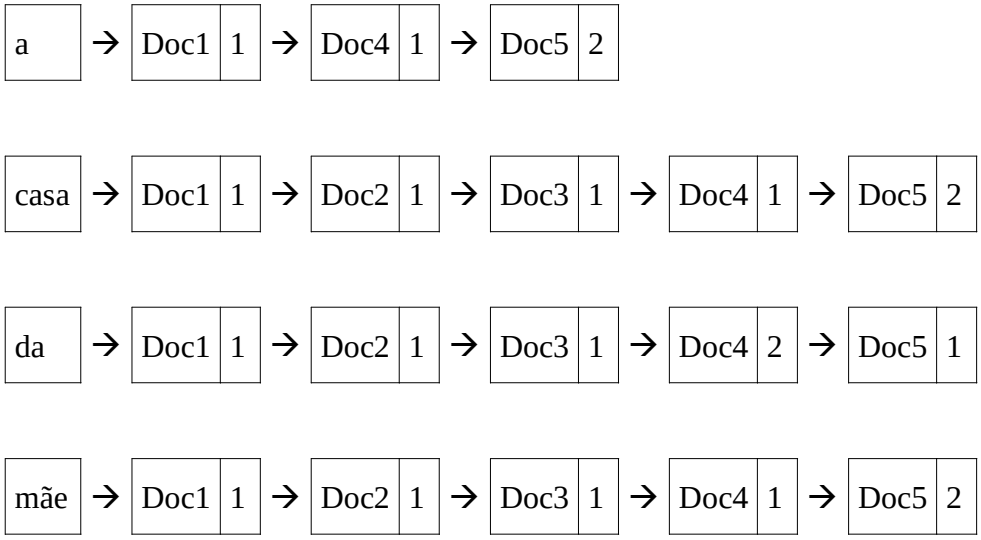

FIGURA 2.1. Arquivo invertido tradicional criado para a coleção de documentos representada na Tabela 2.1.

Segundo [3], "a eficácia de arquivos invertidos na recuperação da informação é totalmente dependente da definição e da seleção dos termos (palavras) a serem indexados ". Sendo assim, deve-se indexar as palavras que realmente podem ser úteis na recuperação de informações, as quais possam permitir o retorno apenas de resultados relevantes.

Como nos documentos, normalmente, os substantivos apresentam maior relevância se comparados a artigos, preposições e conjunções, visto que estas últimas classes de palavras aparecem na maior parte dos documentos de coleções textuais, uma prática aconselhável e comum seria descartar estas palavras, as quais na literatura são chamadas *StopWords* [3].

Tomando-se, então, a Tabela 2.1 como exemplo e considerando-se a eliminação de stopwords, o arquivo invertido gerado diante a coleção de documentos desta referida tabela, ficaria conforme mostra a Figura 2.2.

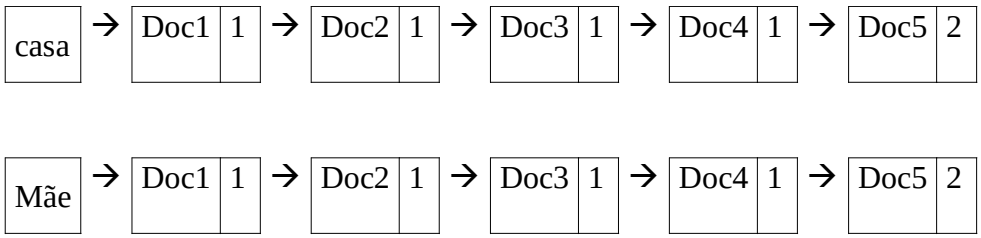

FIGURA 2.2. Arquivo invertido criado para a coleção de documentos representada na Tabela 2.1, eliminando-se as stopwords.

2.1. Granularidade do Índice

A granularidade de um índice corresponde à precisão com que uma palavra é localizada por ele. Normalmente, arquivos invertidos são aplicados para a geração de índices que se referem a documentos completos, e neste caso a granularidade do índice é dita moderada [10].

Existe também a opção de se utilizar apenas blocos de textos pertencentes aos

documentos, o que caracteriza uma granularidade de índice grossa.[4]

Neste trabalho, a granularidade do índice considerada será a denominada granularidade de índice fina, sendo que esta identifica cada palavra dos documentos completos, inclusive as *StopWords* e, em alguns casos, identifica cada byte [10]. A partir desta opção de granularidade, o arquivo invertido representado na Figura 2.3, terá o índice ordenado alfabeticamente pelas palavras dos documentos, juntamente com a freqüência que estas palavras ocorrem, ordenada de forma decrescente. Deve-se notar que a eliminação de Stopwords não está sendo considerada.

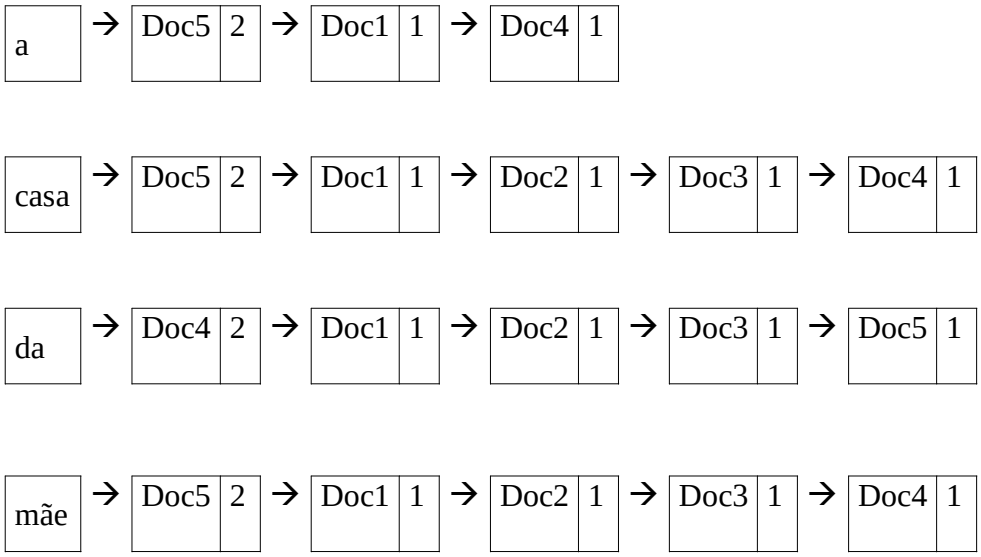

FIGURA 2.3. Arquivo invertido ordenado pela freqüência para a coleção de documentos representada na Tabela 2.1.

3. Construção de um Arquivo invertido

"A construção de um índice utilizando a técnica de arquivo invertido pode ser dita relativamente simples se a coleção de documentos for pequena, visto que estruturas como árvores binárias ou tabelas *hash* podem ser implementadas e carregadas na memória principal" [2]. Porém, quando a coleção de documentos cresce, torna-se impossível o uso da memória principal, exigindo-se assim o uso de memória secundária, normalmente, um disco rígido.

A implementação do arquivo invertido apresentada neste trabalho contemplará o uso de memória secundária, visto que o número de informações textuais a serem aqui tratadas é muito grande, sendo impossível o uso apenas da memória principal [6].

O **Algoritmo 1**(encontrado na seção 3.1), considerado e implementado para este trabalho funciona, de forma geral, da seguinte maneira:

- À medida que os textos de uma determinada coleção vão sendo processados (lido), as ocorrências de cada termo (palavra) vão sendo armazenadas em uma área pré-definida da memória principal.

- Quando esta área da memória principal estiver totalmente preenchida, as ocorrências contidas nesta área são ordenadas e gravadas em memória secundária, gerando arquivos temporários.

- Quando termina o processamento de todos os textos da coleção a ser tratada, faz-se a intercalação dos arquivos temporários armazenados anteriormente, gerando-se um arquivo invertido final.

A Figura 3.1 ilustra o funcionamento do algoritmo para a criação do arquivo invertido e, seu funcionamento, de forma mais detalhada, será apresentado na seção 3.1.

3.1. Algoritmo de Criação do Arquivo Invertido

O algoritmo utilizado no processo de indexação que gera o arquivo invertido, representado na Figura 3.1, pode ser visualizado abaixo. O significado dos símbolos nele utilizados são mostrados na tabela 3.1.

para cada $d_j \in C$ faça

  leia(d_j)

  para cada $t_i \in d_j$ faça

    $B \leftarrow B \cup <t_i,d_j,f_{i,j}>$

    se $|B| \geq M$ then

      ordene(B)

      grave(B,A')

      $R \leftarrow R + 1$

      $B \leftarrow \Phi$

    fim se

  fim para

fim para

$A \leftarrow$ merge(A',R)

Algoritmo 1 - Destinado à geração seqüencial de arquivo invertido ordenado pela freqüência.

Símbolo	Significado
d	Documento
C	Cadeia de Documentos

t	Termo
B	Buffer de Triplas
f	Frequência do Termo
M	Memória Disponível
A’	Arquivo Temporário
R	Número de Arquivos Temporários
A	Arquivo Invertido

TABELA 3.1. Significado dos símbolos utilizados no algoritmo.

O funcionamento do **Algoritmo 1** é representado na **figura 3.1,** sendo que esta é melhor detalhada a seguir:

No primeiro passo, objetiva-se selecionar o vocabulário da coleção de documentos a ser tratada. Neste passo é realizado o ***Parsing*** (indicado na Figura 3.1), onde cada texto é lido e as palavras são identificadas. As palavras distintas encontradas são armazenadas em um **Dicionário** (também indicado na Figura 3.1). Este dicionário é implementado por meio de uma tabela hash, que é uma estrutura de dados especial que associa chaves de pesquisa (hash) a valores. A utilização da tabela hash para representar o dicionário, se baseia no fato desta estrutura ser capaz de apresentar um melhor desempenho em pesquisas e recuperações, se comparadas a outras estruturas de dados, uma vez que o acesso a seus dados é da ordem de O(1). Assim sendo, se existir uma tabela com 100 termos, o acesso ao último termo é feito de forma direta, gastando apenas um acesso [7].

Faz-se então a leitura dos documentos identificando, além dos termos (palavras), os documentos onde estes termos ocorrem e sua respectiva freqüência em cada um deles. São então formadas **Triplas** (também indicado na Figura 3.1) contendo o termo em questão, a sua freqüência no documento *d* e o identificador do documento *d*. Tais triplas são então copiadas para um **Buffer de Triplas**(também indicado na Figura 3.1)**,** que é uma área da memória de tamanho pré-determinado, para uso na indexação.

Quando o **Buffer de Triplas** está totalmente preenchido, faz-se sua ordenação, utilizando o método de ordenação QuickSort (visto que este é um método de ordenação mais rápido e eficiente se comparado aos demais [8]), sendo esta ordenação feita em relação ao termo juntamente com sua freqüência. Após feita a ordenação, o conteúdo do buffer de triplas é gravado na memória secundária, gerando um arquivo temporário. Após feita esta gravação, o buffer é esvaziado para que se possa repetir este procedimento até que toda a coleção de documentos tenha sido processada.

Ao final deste processamento, passa-se à tarefa de se realizar a intercalação (**Merge**) dos arquivos temporários gerados. Para se fazer esta intercalação é utilizada a estrutura de dados denominada fila de prioridades, por ser uma estrutura muito útil na resolução de problemas nos quais se necessita encontrar rápida e repetidamente o maior elemento de uma coleção de valores e removê-lo desta coleção [9]. O número de posições disponíveis nesta fila de prioridades deve ser proporcional ao número de arquivos temporários produzidos anteriormente. Sendo assim, se *n* arquivos temporários foram gerados, a fila deve possuir *n* posições, a fim de que se possa intercalar todos os arquivos temporários e gerar um arquivo final.

Após se ter a fila de prioridades preenchida com todos os arquivos temporários produzidos, é definido um vetor contendo um elemento de cada arquivo temporário e o nome do respectivo arquivo. Logo depois é feita a ordenação deste vetor, utilizando novamente o método QuickSort. Após se obter o vetor ordenado, retira-se o seu maior elemento e o armazena em um buffer. Para cada elemento retirado do vetor, acessa-se na fila de prioridades, arquivo temporário correspondente ao elemento que acaba de ser retirado, transferindo para o vetor o seu próximo elemento e o seu nome. O preenchimento do buffer é sempre realizado a partir de sua última posição preenchida e quando este buffer se torna cheio, ele é então gravado em memória secundária.

Após gravado o conteúdo do buffer em memória secundária, este é esvaziado para que se possa repetir as operações descritas acima, até que todos os arquivos temporários tenham sido processados até o seu fim.

Ao fim da intercalação dos arquivos temporários, tem-se gerado o arquivo invertido final.

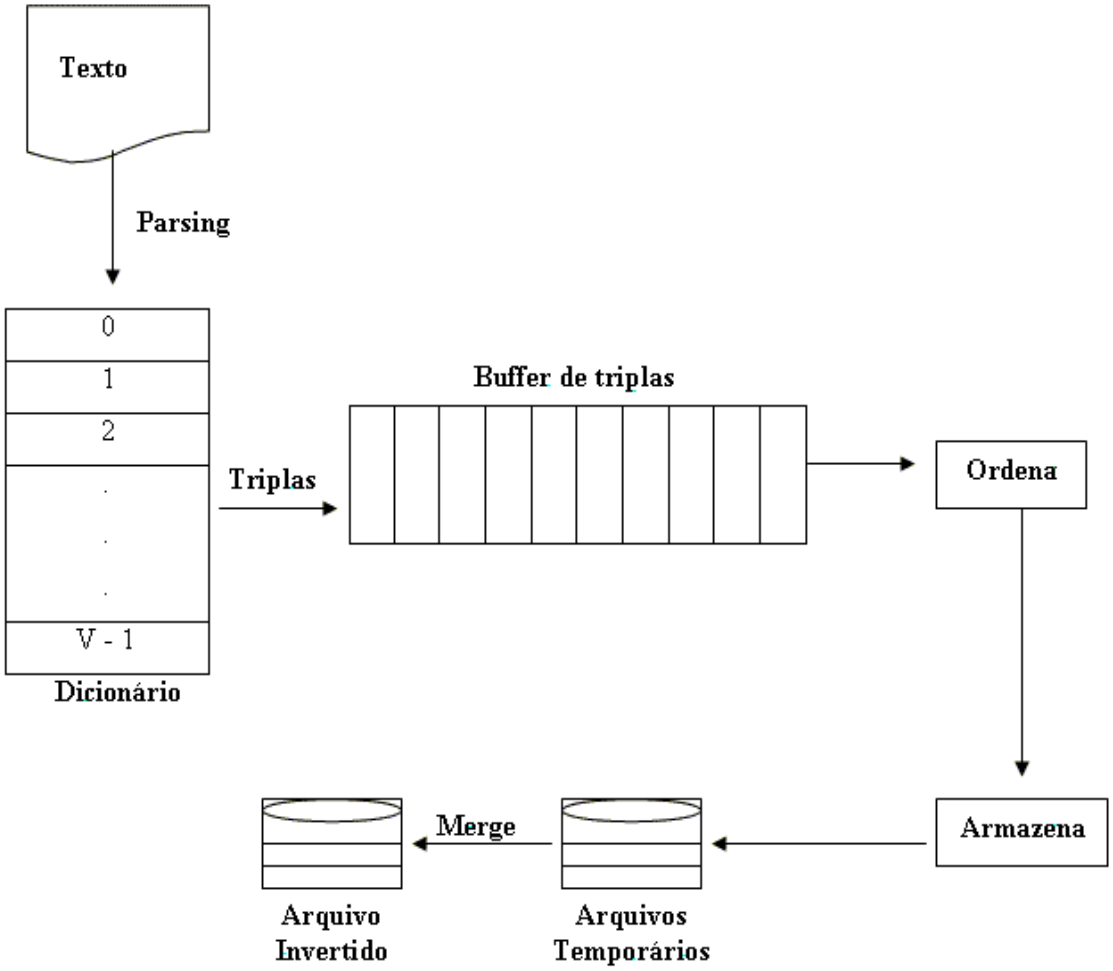

FIGURA 3.1. Esquema de passos a serem seguidos para a criação de um arquivo invertido.

4. O Processamento de Consultas

Depois de explicitado o algoritmo destinado à obtenção de um arquivo invertido, serão consideradas dois tipos de consultas que podem utilizar este tipo de índice na

recuperação eficaz de informações em uma determinada coleção de documentos, a saber: **Consultas Lógicas** e **Consultas Ordenadas**.

Em uma consulta lógica os termos fornecidos são conectados pelas expressões AND, OR e NOT, e as respostas à consulta serão os documentos que satisfazem às condições estipuladas pelos termos juntamente com os conectivos lógicos. A consulta lógica é relativamente simples de ser feita e pode apresentar resultados satisfatórios. Porém, o usuário que deseja fazer a pesquisa deve ter um bom conhecimento sobre os documentos no banco de dados a ser pesquisado. Outro fator a ser observado é que as respostas podem variar muito de acordo com a utilização dos conectivos, pois ao se combinar termos com o conectivo AND, por exemplo, se terá uma resposta um tanto quanto restrita, enquanto que se o conectivo utilizado for o OR, muitos documentos não relevantes à consulta poderão ser retornados.

Uma forma de solucionar a deficiência das consultas lógicas seria a utilização da consulta ordenada, visto que neste tipo de consulta, o usuário informa somente o termo ou um conjunto destes, sem a utilização de conectivos. A partir daí, o sistema de recuperação de informações determina uma medida de semelhança entre o(s) termo(s) da consulta e os documentos indexados. Essa medida de semelhança é um indicador numérico que determina quais documentos mais se similarizam à consulta, retornando-os para o usuário. Para a determinação desta medida de semelhança, pode-se levar em conta o peso de um termo em relação ao documento avaliado no momento, ou seja a freqüência do termo naquele documento. Pode-se também considerar a freqüência do termo na coleção, onde os termos mais raros tem maior relevância e, ainda pode ser considerado o tamanho do documento, visto que quanto mais curto o documento, maior privilégio este documento terá.

Fica, então, a cargo do usuário do arquivo de índice gerado, a escolha de qual tipo de consulta utilizar, visto que este trabalho se atém à geração do arquivo invertido, estando a implementação de mecanismos de consultas que utilizem este arquivo, fora do escopo do mesmo.

5. Considerações Finais

Neste trabalho é apresentada e implementada a técnica de Arquivos Invertidos, muito utilizada no processo de recuperação de informação. Com a implementação e análise do funcionamento deste algoritmo (sendo que a figura 5.1 representa um teste feito para a coleção de documentos da tabela 2.1, localizada na seção 2 deste trabalho), detalhado na Seção 3, pôde-se observar que a técnica em questão é eficiente, rápida e relativamente fácil de ser implementada, gerando um arquivo final de fácil utilização para consultas, o que possibilita uma recuperação de informações concisa e eficiente. O algoritmo implementado contribui na facilidade de se usar uma consulta a grandes volumes de texto, podendo-se recuperar informação nesses textos.

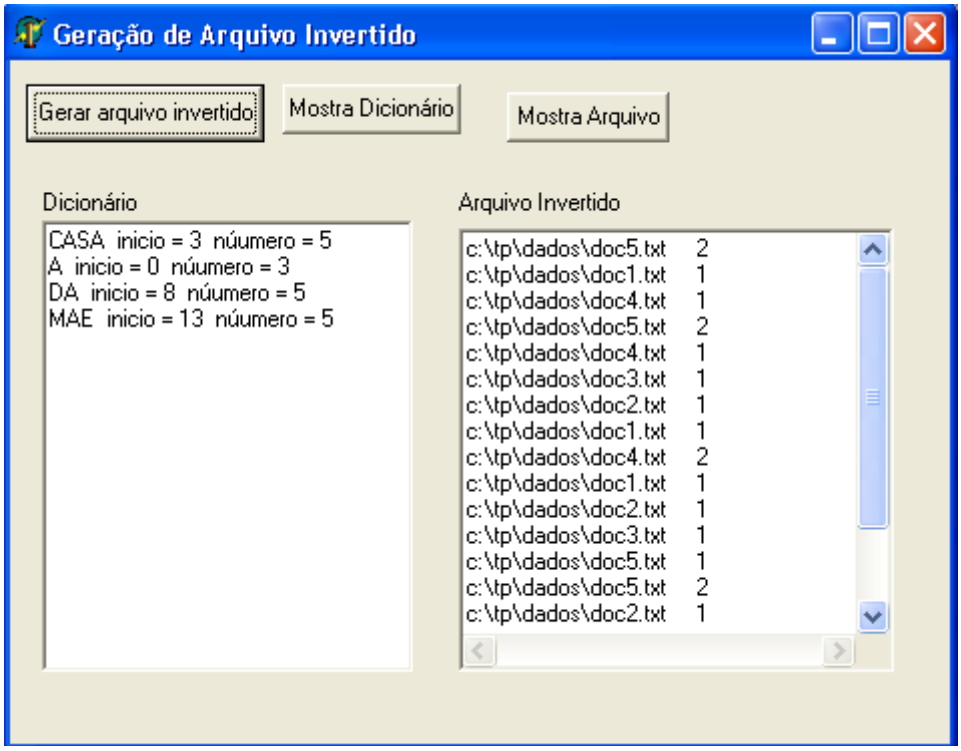

FIGURA 5.1. Tela de teste do algoritmo implementado, tendo como base a coleção de documentos da tabela 2.1, apresentada na seção 2.

No intuito de se aperfeiçoar o algoritmo implementado, pode-se sugerir como trabalho futuro, a compressão dos dados armazenados, substituindo os termos (palavras) por outros símbolos que usem um número menor de bits ou bytes, possibilitando assim um maior aproveitamento na transferência de dados da memória principal para a memória secundária e vice-versa.

Além disto, pode-se desenvolver um sistema dedicado à recuperação de informação que utilize tal algoritmo tanto para consultas lógicas quanto para consultas ordenadas, para que se possa observar em termos práticos, a sua eficiência.

6. Referências Bibliográficas

[1] FIGUEIREDO, C. X., FRANCICANI, J. F. e CARDOSO, O. N. P. (2003). *Recuperação de Informação e Bibliotecas Digitais*. Publicação: 5ª Semana de Ciência da Computação, Universidade Federal de Lavras. Disponível em: <http://www.dcc.ufla.br/~olinda/arquivos/apostila_RI.pdf>. Acesso em: 10 de outubro de 2006.

[2] LOPES, V. J. (2004). *Técnicas de análise para a usabilidade de diferentes máquinas de busca*. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação pela Universidade Presidente Antonio Carlos.

[3] MOFFAT, A.; WITTEN, I.; BELL, T. Managing Gigabytes: Compressing and Indexing Documents and Images. Series in Multimedia Information and Systems. Morgan Kaufmann, San Francisco, California, 1999.

[4] ZIVIANI, N. (2005). *Arquivos Invertidos*. Disponível em: <http://homepages.dcc.ufmg.br/~nivio/cursos/ri05/tp1/ri05tp1.html>. Acesso em: 10

de outubro de 2006.

[5] NETO, M. C. T. (2001). *Estrutura de dados para Indexação de Grandes Volumes de Dados*. Trabalho de Graduação Em Ciência da Computação, Universidade Federal de Pernambuco.

[6] NEUBERT, M. (2000). *Algoritmos distribuídos para a construção de arquivos invertidos*. Dissertação de Mestrado, Universidade Federal de Minas Gerais.

[7] WIKIPEDIA. (2006). *Tabela Hash*. Site Wikipedia. Disponível em <http://pt.wikipedia.org/wiki/Tabela_hash>. Acesso: 20 de outubro de 2006.

[8] WIKIPEDIA. (2006). *Tabela Hash*. Site Wikipedia. Disponível em <http://pt.wikipedia.org/wiki/Quicksort>. Acesso em: 25 de outubro de 2006.

[9] FUNDÃO DA COMPUTAÇÃO. (2002). *Fila de Prioridades*. Site Fundão da Computação. Disponível em: <http://www2.fundao.pro.br/articles.asp?cod=65>. Acesso em: 20 de outubro de 2006.

[10] ZIVIANI, N. (2005). *Tópicos em Recuperação de Informação*. Disponível em: <http://homepages.dcc.ufmg.br/~nivio/cursos/ri05/transp/indexing.ps>. Acesso em: 15 de outubro de 2006.

6. Leitura Complementar

CANTU, M. (2002). *Dominando o Delphi TM 6: a biblia*. Tradução de João Eduardo Nobrega Tortello. São Paulo: Makron Books, 2002. 934 p. il. ISBN 85-346-1408-3.

NASCIMENTO, D. (2004). *Manipulando arquivos texto*. Site Crie Seu Web Site. Disponível em: <http://www.crieseuwebsite.com/tutoriais/abretutorial.php?tutorial=32> Acesso em: 20 de outubro de 2006.

PASIN, M. (2004). *Compressão de dados*. Disponível em: <http://www-usr.inf.ufsm.br/~marcia/pesquisa/transpalunos/comprdados.pdf>. Acesso em: 28 de outubro de 2006.