

Orfeu, um sistema para busca e manipulação de arquivos de música

Juliana Lucas de Rezende, Vinícios Pereira, Vinicius VonHeld, Amanda Varella, Guilherme Figueiredo, Marcelo Mayworm, Rógea Rocha, Geraldo Xexeo

Coordenação dos Programas de Pós Graduação de Engenharia da Universidade Federal do Rio de Janeiro (COPPE/UFRJ)
Ilha do Fundão – Rio de Janeiro – RJ – Brazil

{vinicios, vonheld, varella, g.coelho, mayworm, rogeasr, xexeo}@cos.ufrj.br, (ju_lucas)@yahoo.com.br

Abstract. *This paper presents a system for Internet files searching, indexing and retrieval. The first module is a framework based on the web crawling technology that is able to handle any type of file of specific subjects. This framework was instantiated to search song files in formats MIDI, MusicXML and MP3 in music web sites. The system uses greedy search to visit web pages and sorts them according to their relevance. The other module converts the found files into a specific format, index them into a spatial structure (kd-tree), allowing content-based retrieval. Given a sequence of eight notes, the system retrieve songs containing one or more pieces identical or similar to the sequence of entrance.*

Resumo. *Esse artigo apresenta um sistema para obtenção e indexação de arquivos da Internet. O primeiro módulo é um framework baseado na tecnologia de web crawling que busca e baixa arquivos de assuntos específicos. Esse framework foi instanciado para buscar arquivos nos formatos MIDI, MusicXML e MP3 em páginas relacionadas à música. O sistema utiliza busca gulosa para visitar páginas e as ordena de acordo com a sua relevância. O segundo módulo converte os arquivos encontrados para um formato específico e os indexa em uma estrutura espacial chamada kd-tree, permitindo que se realize buscas por conteúdo. Dada uma seqüência de oito notas, o sistema recupera as músicas que contenham um ou mais trechos idênticos ou semelhantes ao trecho de entrada.*

1. Introdução

Com o surgimento e a disponibilidade de arquivos de música digitalizados, o número de usuários interessados em aumentar seu acervo musical cresceu em grandes proporções. No entanto, o volume de informação disponível da *Internet* também aumenta absurdamente a cada ano, tornando muito difícil encontrar arquivos de música navegando por sites. Muitos programas foram criados para facilitar a troca de arquivos de música entre usuários localizados em máquinas diferentes. Essa arquitetura, denominada *peer-to-peer*, depende da contribuição de usuários com arquivos que tenham permissão para serem baixados dos seus computadores. Além disso, normalmente as ferramentas não permitem que sejam baixados todos os arquivos disponibilizados que possuam uma certa extensão.

Por outro lado, o aumento do acervo musical dificulta a localização de uma música específica e de informações relativas a ela em um grande universo de arquivos de músicas. Muitas vezes a única informação com a qual o usuário pode contar é uma vaga lembrança da melodia da música. A proposta deste trabalho é apresentar um sistema composto por dois módulos integrados. O primeiro módulo é responsável por localizar e baixar arquivos e o segundo é responsável por indexar e possibilitar ao usuário a consulta aos arquivos encontrados.

O primeiro módulo é um *framework*, baseado na tecnologia de *web crawling*, para localizar arquivos de qualquer extensão definida pelo usuário. No entanto, a busca é direcionada por um conjunto de palavras previamente estabelecidas que irão definir o grau de relevância de uma página. É preciso deixar clara a necessidade de uma certa margem de erro ao baixar arquivos de formatos comuns para assuntos específicos. Nesse trabalho foi criada uma instância deste *framework* para arquivos musicais. Outras instâncias permitiriam que o processo fosse aplicado, de maneira semelhante, para outros tipos de arquivos e assuntos.

O objetivo do módulo de indexação e consulta é indexar arquivos musicais, de forma que seja permitida a recuperação de músicas por conteúdo, ou seja, tendo como entrada uma seqüência de notas, o sistema consiga recuperar as músicas que contenham trechos idênticos ou semelhantes ao trecho de entrada. É importante que o *link* do *site* que contém a referência para o arquivo de música seja também armazenado no banco de dados, já que o objetivo do sistema é fazer com que o usuário encontre informações sobre músicas sobre as quais ele se recorda somente da melodia. É provável que neste *site* tais informações possam ser encontradas.

Este artigo está organizado de forma que na seção 2 é dada uma breve introdução ao formato MIDI, da mesma forma que é mostrado o formato MusicXML na seção 3. Na seção 4 mostramos como funciona a tecnologia de *web crawling*. Na seção 5 apresentamos a arquitetura do módulo de obtenção de arquivos na web. Na seção 6 mostramos o funcionamento do *crawler*, na seção 7 como as páginas são mineradas e como é estabelecido um peso para seus *links*. Na seção 8. A conclusão e os trabalhos futuros serão mostrados nas seções 100 e 111, respectivamente.

2. Formato MIDI

As formas de armazenamento de música em meio digital vêm caminhando para uma padronização. Armazenar dados de áudio em formato analógico discretizado consome muito espaço. O padrão MIDI (“*Musical Instrument Digital Interface*”) [Heckroth, 95] de armazenamento gera arquivos extremamente pequenos quando comparados a arquivos gerados por amostragem de ondas sonoras. Isto se dá em virtude do arquivo MIDI não armazenar o sinal sonoro propriamente dito, e sim os comandos necessários para gerá-lo. Os sons são gerados posteriormente pelos sintetizadores. Este formato facilita a análise da música, tornando mais simples a extração de padrões e a identificação de características referentes à seqüência de notas. O padrão MIDI é hoje um padrão de fato da indústria de instrumentos musicais eletrônicos e de programas musicais (editores e seqüenciadores). Essas características tornam arquivos MIDI a forma mais aconselhável para o registro musical em computador e, conseqüentemente, o alvo mais adequado para sistemas de indexação musical.

3. Formato MusicXML

O MusicXML foi criado baseado nos principais formatos musicais existentes comercial e academicamente, mas sua principal influência foi de formatos extremamente utilizados na área acadêmica, como o MuseData [Selfridge, 1993] e o Humdrum [Selfridge, 1997]. Segundo os próprios autores, MusicXML é uma atualização do MuseData para trabalhar com XML, utilizando alguns conceitos-chaves do Humdrum. Sendo um formato aberto, inúmeros softwares já estão utilizando o MusicXML para troca de informações.

Em MusicXML, assim como em XML, a informação é representada de maneira hierárquica. Desta maneira, é feita uma separação entre os elementos sonoros e de notação.

Um grande problema de formatos como MIDI é que a música transcrita (partitura) é diferente da execução desta música, pois o músico executor interpreta os símbolos da partitura de maneira particular, fazendo variações rítmicas e de intensidade. Em MusicXML, há uma separação dos elementos sonoros dos elementos de notação, permitindo que um mesmo arquivo represente paralelamente uma informação sonora fiel, e sua notação mais clara. Como um exemplo, os nós *duration* (duração), e *type* (tipo), apesar de possuírem uma dependência semântica, são definidos separadamente e independentemente, conforme o exemplo dado.

4. Web Crawling

O objetivo dos web crawlers é a coleta automática e sistemática de documentos da Web a serem indexados e consultados pela máquina de busca. Eles iniciam sua coleta a partir de uma lista inicial de URLs para indexar seu conteúdo em um repositório central. A ordenação da lista de URLs armazenada nesse repositório, segundo uma heurística, é importante para direcionar o crawler para sites de maior relevância para o assunto desejado.

Em geral, o web crawling é o gargalo no processo de indexação de páginas para sistemas de busca na web. Isso porque são exigidos banda e processamento para analisar, processar e indexar as páginas da Internet.

O web crawling pode ser genérico, indexando páginas de todo e qualquer tipo para agilizar buscas de usuários de diferentes interesses. Por outro lado o desempenho do crawler pode ser melhorado uma vez que ele seja focado em um determinado assunto, concentrando recursos na busca por um grupo seletivo de páginas ou em um determinado domínio. Nesse trabalho o web crawling é focado na busca de arquivos de música nos formatos MIDI, MusicXML e MP3.

O Web crawler apresentado nesse trabalho possui todas as funcionalidades de uma arquitetura típica, porém com adaptações de funcionalidades consideradas pertinentes ao problema específico.

5. Arquitetura do Módulo de Obtenção de Arquivos

O sistema deve ser inicializado com o banco de parâmetros devidamente preenchido com os dados necessários para o reconhecimento das palavras-chave que devem dar maior peso às URLs próximas a elas e com as stop-words existentes nas línguas que

serão pesquisadas. Além disso, deve haver, pelo menos, uma URL na fila. Essa(s) URL(s) será(ão) chamada(s) inicial(is) e serão as primeiras a serem visitadas e avaliadas pelo crawler.

A arquitetura externa do sistema é mostrada na figura 3. O crawler, formado por um único coletor, inicia o processo de busca retirando da lista de URLs a próxima a ser consultada. Nesse trabalho chamamos de banco de armazenamento do crawler o que, na arquitetura geral, é chamado de escalonador. Isso porque nesse sistema os arquivos, objetivo da busca, são armazenados em diretórios do sistema operacional, não havendo assim a necessidade de uma base para armazenamentos de páginas encontradas. Dessa forma, a base de armazenamento passa a ser responsável por armazenar a fila e a memória do crawler, ou seja, as páginas já visitadas e os arquivos já encontrados. A memória é responsável por evitar que páginas sejam re-visitadas e que arquivos sejam duplicados, evitando assim desperdício de tempo, banda, processamento, e espaço em disco.

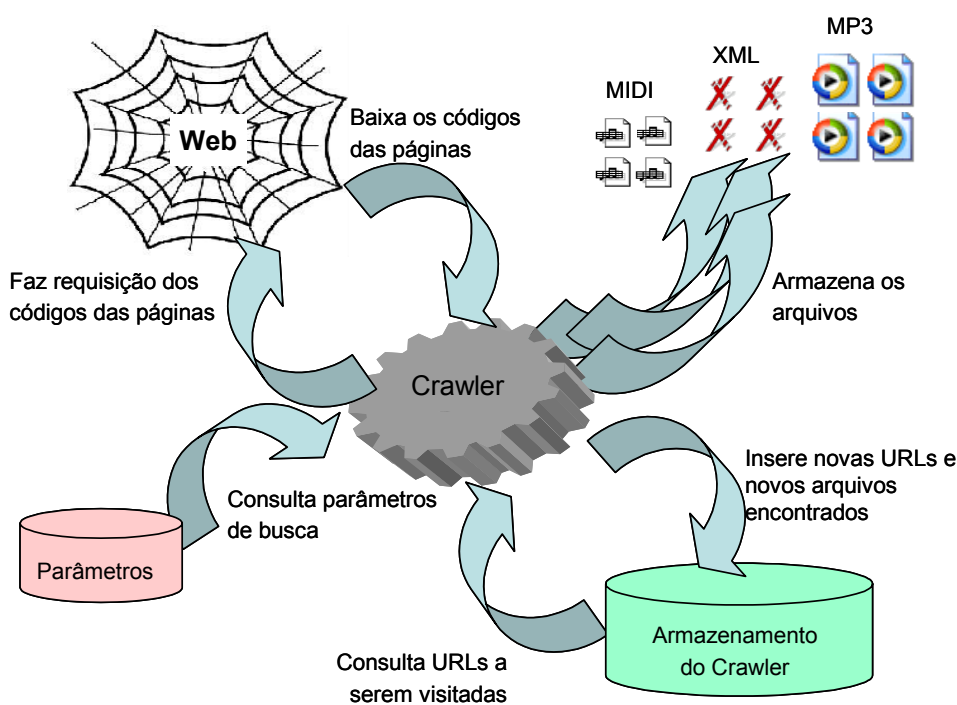


Figura 1. Arquitetura do módulo de obtenção de arquivos.

De posse da próxima URL a ser visitada, o crawler faz uma requisição HTTP do código da página, faz uma análise do seu texto e extrai todos os links (URLs) existentes. Caso haja alguma nova URL que aponte para um arquivo MIDI ou MP3 o crawler verifica se o arquivo já foi baixado, caso não tenha sido ele cria uma nova thread que baixa o arquivo e o armazena em um diretório do sistema operacional. Além disso, guarda uma referência no banco de dados para que seja consultada nas próximas vezes que um arquivo for baixado. Caso a URL referencie um arquivo XML [W3] a nova thread do crawler verifica se esse arquivo é um arquivo MusicXML e, se for o caso, baixa o arquivo usando o mesmo procedimento adotado para os outros tipos de arquivo, com o adendo de que, no caso do MusicXML, é extraído um número maior de informações sobre a música, como título da música, compositor, etc.

A vantagem de se criar novas threads para baixar os arquivos é permitir que o crawler possa continuar a buscar novos arquivos enquanto outros estão sendo baixados. Se muitos arquivos forem encontrados em um curto espaço de tempo, uma thread será inicializada para cada arquivo, assim, os arquivos menores não terão que aguardar que arquivos maiores, como MP3, sejam baixados. Paralelamente, a thread principal continua sua busca pelos links das páginas HTML.

6. O Crawler

O crawler foi concebido de forma que o assunto e os arquivos baixados sejam parametrizados. Dessa forma ele pode ser re-configurado para baixar arquivos com qualquer extensão (desde que haja uma implementação para a interface Advisor para a respectiva extensão). Para baixar arquivos XML o sistema valida o seu formato através da definição pública (DTD) do musicXML disponível em <http://www.musicxml.org/dtds/partwise.dtd>. Além da extensão, o crawler pode ser configurado para dar prioridade a palavras próximas de palavras-chave de outros assuntos diferentes de música.

O fluxo interno do crawler e sua interação com as bases de dados é mostrado na figura 5. Nela podemos observar que ele inicia seu processo recuperando a URL com maior peso na tabela FILA_URL. Isso é possível através da ordenação dos registros da tabela em relação ao atributo “peso”. Caso a URL referencie um arquivo com alguma extensão configurada nos parâmetros, o crawler baixa o arquivo, armazena esta URL na tabela MEMORIA e volta a pegar a próxima URL a ser visitada. Caso a URL referencie uma página o crawler baixa o seu conteúdo HTML, remove as stop-words e busca pelas URLs existentes no site e os textos próximos a elas. O número de palavras-chave encontradas próximas a uma URL define qual será o seu peso. O crawler verifica a existência dessa URL na memória, caso ela não tenha sido visitada anteriormente, ela é armazenada na lista de páginas a serem visitadas. A URL da página que acabou de ser analisada é armazenada na memória e o crawler retorna ao início do processo.

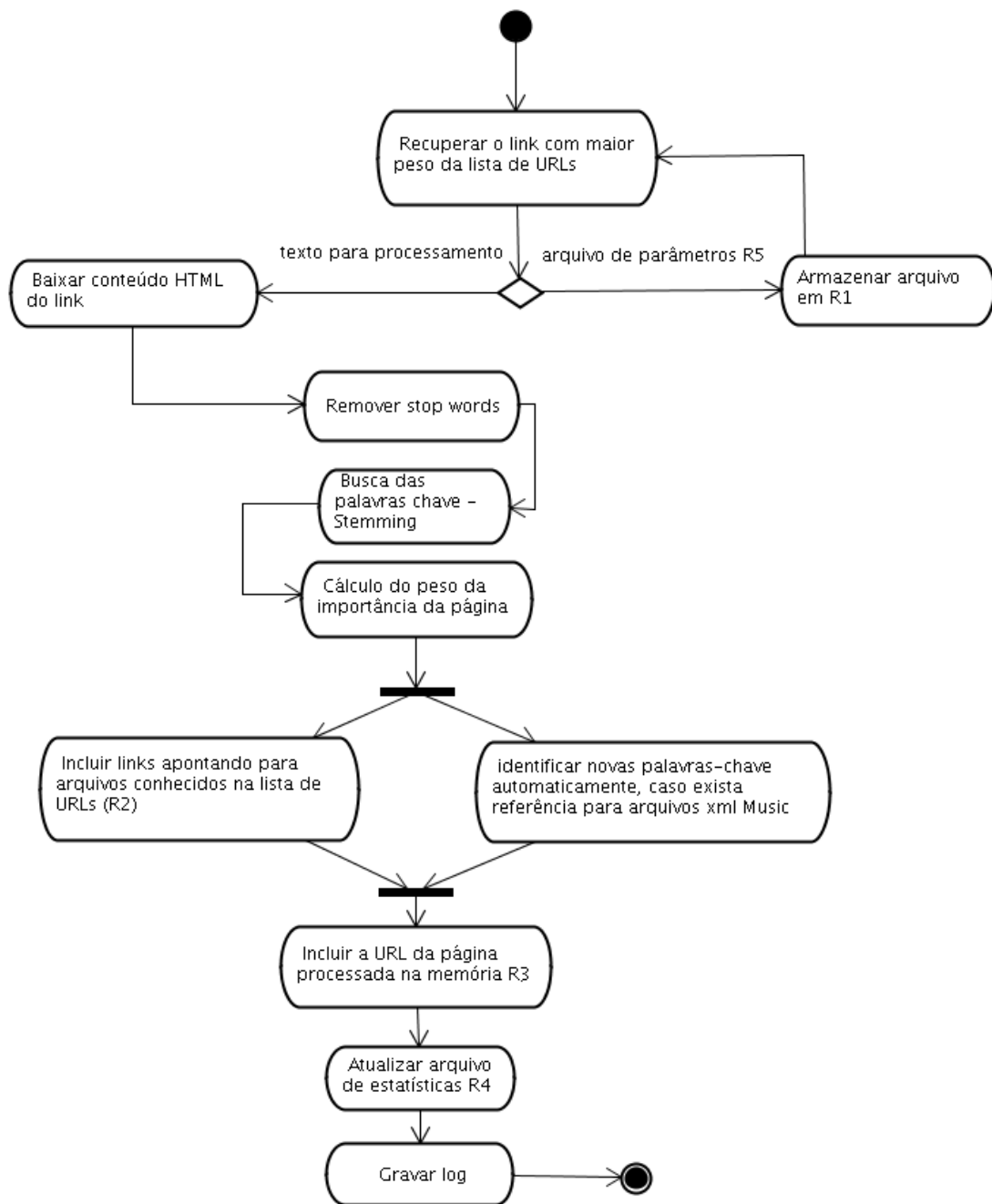


Figura 2. Fluxo interno do crawler.

A figura 6 apresenta a interface gráfica do módulo de obtenção de arquivos, onde podem ser observadas as URLs que estão sendo analisadas pelo Crawler no momento.

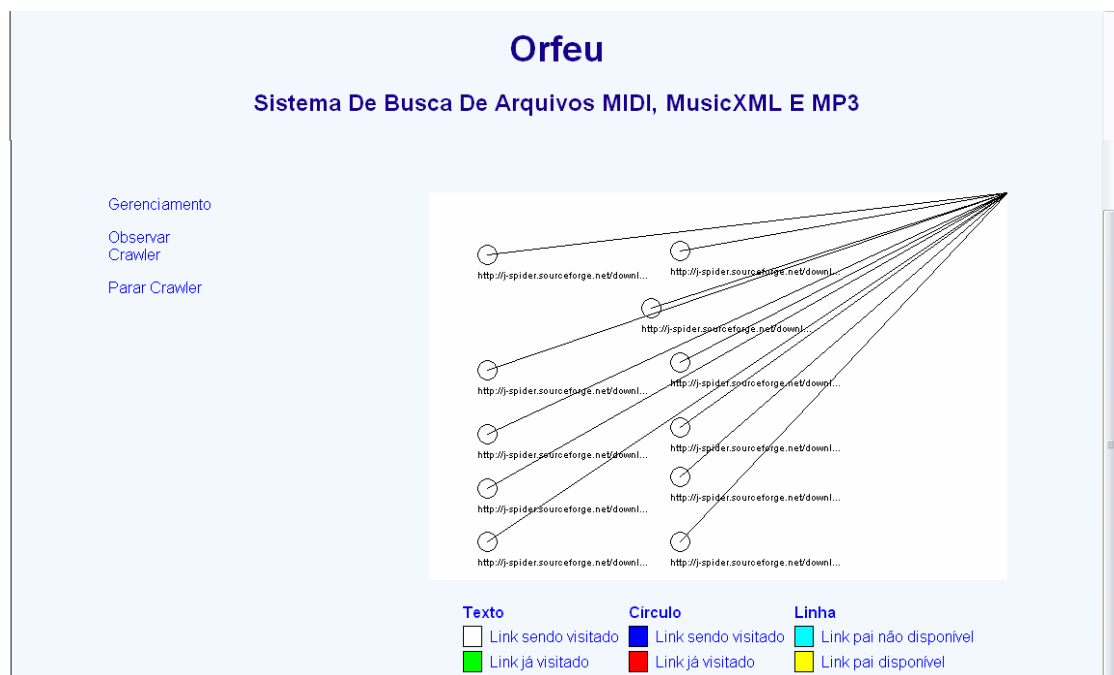


Figura 3. Interface gráfica do módulo de obtenção de arquivos.

7. A Definição do “Peso” das URLs

Para fazer com que o Crawler dê prioridade a links (URLs) “interessantes”, de forma a direcionar sua pesquisa ao assunto determinado pelo usuário a partir de um conjunto de palavras-chave configuráveis, atribuímos um peso a cada link encontrado em uma página.

O peso de um link é proporcional a sua relevância em relação ao assunto pesquisado pelo Crawler. O assunto a ser pesquisado é definido por um conjunto de palavras-chave armazenadas em uma tabela no banco de dados do Crawler (KEYWORDS). Cada registro de palavra-chave possui além do termo em si, um peso, de forma a diferenciarmos a importância das próprias palavras em relação ao assunto. Como o Crawler foi desenvolvido para buscar arquivos de música na Web, a palavra-chave “música” terá uma importância, ou um peso, maior do que a palavra-chave “som”, por exemplo.

A partir das palavras-chave e de seus pesos, podemos calcular o peso de um link em uma página HTML. Para isso, buscamos palavras-chave próximas ao link, e calculamos um peso para este link a partir do número de palavras-chave encontradas e de seus pesos.

O peso de um link será o somatório dos pesos das palavras-chave próximas ao link (N palavras antes e N palavras depois do link, sendo N variável – em nossa implementação N = 10) dividido pelo número total de palavras-chave no banco somado com o número de ocorrências múltiplas de uma palavra-chave.

Como os pesos de uma palavra-chave variam de 0 a 10, teremos o peso de um link também variando entre 0 e 10. Os links com maior número de palavras-chave

próximas a eles terão um peso mais alto e serão colocados no topo da fila de URLs a serem visitadas.

Além do peso pelas palavras-chave próximas ao link, somamos a este um peso relativo ao conteúdo do link. Se o link aponta para um arquivo de interesse do Crawler (MIDI, MP3 ou musicXML), somaremos ao peso deste link um valor (15) maior do que o peso máximo atribuído pelas palavras-chave (10) de forma a colocá-lo em primeiro lugar na fila de URLs a serem visitadas.

Definido o peso das URLs da página elas são inseridas na tabela `FILA_URL`. O crawler utiliza busca gulosa pra definir a ordem de visita das páginas a partir da página com maior peso, ou seja, maior grau de relevância. Essa estratégia de busca informada é conhecida como busca gulosa, pois o caminho seguido é sempre o com maior heurística. Essa busca é completa, mas não ótima, para um número finito de nós. No entanto, para o tipo de busca desse trabalho a “solução” de encontrar arquivos não é única. Como o crawler não irá parar ao encontrar um arquivo, não é necessário que a estratégia de busca seja ótima.

8. Módulo de indexação e consulta

Para o servidor de indexação de músicas, foi utilizada a abordagem de conversores, ou seja, os arquivos musicais devem ser convertidos para um formato único que é reconhecido pelo sistema. O formato utilizado nesse projeto foi um formato semelhante ao *ringtone*, muito utilizado para campanhas de telefones celulares. Esse formato permite descrever uma melodia (seqüência de notas) monofônica (sem notas tocando simultaneamente): Cada nota é representada por dois números, um representando a nota propriamente dita e o outro, a duração da mesma. Normalmente, uma música é polifônica, onde diferentes instrumentos compõem diferentes seqüências melódicas, o que chamamos, nesse projeto, de canais. Cada canal da música, então, é convertida para o formato *ringtone*. Nesse projeto, foi implementado o conversor para arquivos midi. Para que os demais tipos de arquivos pesquisados pelo *crawler* sejam indexados por esse módulo do sistema, é necessário construir um conversor apropriado para cada formato de arquivo musical recuperado.

A unidade indexada nas músicas (o que chamaremos de ‘palavra’ musical) é uma seqüência de oito notas: uma ‘janela’ contendo oito notas ‘desliza’ (a segunda nota da primeira janela será a primeira da próxima janela, e assim por diante) por cada seqüência de cada canal, aplicando uma transformada *wavelet* [Faloutsos, 1996], o que transforma cada grupo de 8 notas em um ponto 7 dimensional. Para indexar esse ponto, utilizamos uma estrutura de dados espaciais chamada *Kd-Tree* [Bentley, 1975].

O que define uma melodia musical não é o valor absoluto das notas, mas sim a distância relativa entre elas. Assim, cada palavra musical é normalizada (o valor de cada nota é subtraído do valor da primeira nota da seqüência), e, dessa forma, o que é indexado não é a própria seqüência de notas, mas sim valores das posições relativas entre as mesmas. As seqüências de durações de notas têm a particularidade de possuir distribuição exponencial em um intervalo [Cavalcanti et al, 1999] (as notas têm distribuição linear). Dessa forma, aplicamos a função logaritmo (base 2) nos valores das durações para tornar linear a distribuição dos mesmos. Assim, a duração das notas pode

receber tratamento semelhante às notas propriamente dita, tanto para normalização, quanto na indexação e em consultas.

9. Arquitetura do Módulo de Indexação e Consulta

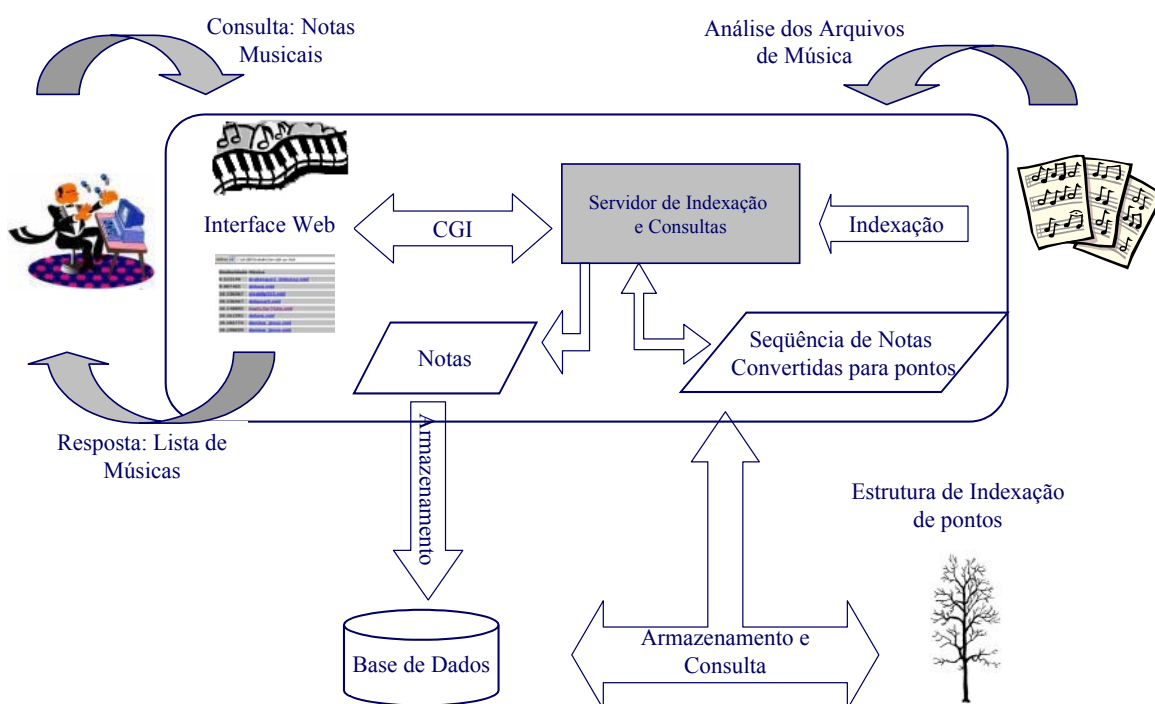


Figura 4. Arquitetura do módulo de indexação e consulta de arquivos.

9.1. Interface

A interface de consultas, acessível via *web*, é composta de um teclado musical (contendo duas oitavas completas). O usuário deve informar uma seqüência de oito notas musicais, clicando no teclado virtual (a duração de cada nota é obtida a partir do tempo que o usuário fica clicando sobre cada nota com o mouse). As notas selecionadas pelo usuário são listadas e podem ser manualmente alteradas. O usuário pode também informar também o grau de similaridade da seqüência informada com as seqüências que poderão ser retornadas na consulta, além de indicar os pesos que nota e duração terão na ordenação nos resultados da busca. Após o processamento da consulta, é exibida na interface uma lista de músicas que contém seqüência(s) similar(es) àquela informada, além dos canais e trechos onde tais seqüências ocorrem nas músicas.

9.2. Indexação de Músicas

Após a aplicação da transformação do arquivo de música original para o formato *ringtone*, o sistema extrai notas e respectivas durações das notas da música. Após a extração, é aplicada uma transformada *wavelet* na seqüência de notas (nas notas apenas, não de duração) de cada canal de cada música, gerando um conjunto de pontos n-

dimensionais (neste trabalho, n é igual a sete). As notas e os pontos são armazenados no banco de dados, mas somente os pontos são indexados em uma *kd-tree*.

9.3. Processamento de consultas

O servidor de consultas recebe como entrada uma seqüência de 8 notas e outra seqüência com as 8 respectivas durações das notas, além dos pesos de nota e duração e o erro (todos informados pelo usuário). Somente a seqüência de notas será utilizada para recuperação de pontos. A seqüência de durações será utilizada para ordenação dos dados recuperados.

A seqüência de notas e respectivas durações de entrada são então normalizadas e nelas é aplicada a transformada *wavelet*, gerando dois pontos 7-dimensional. O sistema recupera da estrutura de indexação (*kd-tree*) os pontos similares ao ponto fornecido, ou seja, aqueles pontos que se encontram dentro da hiper-esfera cujo de raio igual ao erro fornecido pelo usuário, e de centro cujas coordenadas são as mesmas do ponto resultante da transformada *wavelet* aplicada a seqüência de notas de entrada.

Os pontos retornados são referentes a notas, mas possuem durações correspondentes na base de dados. O sistema recupera tais pontos e calcula a similaridade de cada par com a seqüência de entrada, de acordo com a fórmula:

$$\text{coef_similaridade} = \sqrt{\text{peso_nota} * (\text{distância_pt_nota})^2 + \text{peso_duração} * (\text{distância_pt_duração})^2}$$

e ordena os pares de acordo com esse coeficiente: quanto menor o coeficiente, mais próximo é o ponto encontrado ao ponto da consulta (*distância_pt_duração* e *distancia_pt_nota* são as distância euclidianas entre os pontos encontrados e o pontos da consulta).

O sistema então recupera o nome das músicas que contêm pelo menos um dos pares nota/duração recuperados e retorna os resultados para a interface.

10. Conclusão

Nesse artigo apresentamos o sistema Orfeu, um sistema baseado na tecnologia de web crawling para busca e recuperação de arquivos de música. Mostramos uma pequena descrição dos padrões MIDI e MusicXML. Apresentamos aqui a parametrização desse crawler para alterar as extensões de arquivos buscados e assuntos abordados pelas páginas percorridas.

Proporcionamos uma breve descrição dos padrões de projetos utilizados no desenvolvimento do sistema e mostramos os diagramas de classes referentes a cada um deles. Optamos pela utilização desses padrões por serem soluções de qualidades previamente testadas e validadas. Além disso, a utilização desses padrões permite a outros desenvolvedores entender melhor o código do sistema, facilitando sua manutenção ou o desenvolvimento de novas funcionalidades.

Esta implementação mostrou que, com a utilização de algumas técnicas de mineração de texto, é possível criar um crawler direcionado para um determinado assunto a partir de um conjunto de palavras-chave.

Embora *crawlers* genéricos (não direcionados) tenham sido utilizados para indexação do conteúdo da Internet, a utilização de *crawlers* direcionados pode ser útil

para a busca de informação, ou arquivos, na Internet ou em Intranets sobre um determinado assunto específico.

Nos testes realizados, observamos a eficiência do direcionamento do crawler para páginas com conteúdos relativos a música. Embora simples, o algoritmo de atribuição de pesos aos links de uma página se mostrou eficiente para a aplicação em música.

11. Trabalhos Futuros

Uma das melhorias possíveis a serem feitas futuramente é dotar o crawler de capacidade de aprendizado. Dessa forma seria possível a inclusão automática de palavras-chave e a alteração dos seus pesos na tabela KEYWORD. Isso ocorreria quando o crawler encontrasse arquivos em uma página. Uma vez que uma página tenha referências para arquivos com extensões solicitadas nas configurações do sistema, pressupõe-se que as palavras que aparecem com maior frequência nessa página têm algum relacionamento com o tema abordado na busca (nesse trabalho, o tema é música).

Ao encontrar arquivos, o crawler irá retirar as stop-words, irá contar as ocorrências das demais palavras e dividir essa ocorrência pelo número de palavras no texto. Dessa forma uma palavra com ocorrência dois em um texto de cinco palavras tem maior “peso” que uma palavra com a mesma ocorrência em um texto com quinhentas palavras. Para palavras-chave encontradas o sistema irá fazer uma média ponderada do novo peso encontrado com o já cadastrado no banco de dados, como mostrado na fórmula abaixo.

$$Peso = \frac{PesoAntigo + PesoNaPagina * NumeroArquivos}{NumeroArquivos + 1}$$

Para palavras que não estão na lista de palavras-chave, porém possuem peso superior a um *threshold* previamente determinado o sistema irá cadastrá-las com o peso encontrado.

No módulo de indexação e consultas, uma melhoria que compactaria significativamente a base de dados indexada seria a análise do padrão de repetição de pontos em uma música, para considerar (e armazenar) apenas trechos repetitivos da música, que normalmente são refrões ou algumas partes da música que o usuário se lembrará com maior facilidade. Trechos repetitivos de uma música podem ser facilmente obtidos através do relacionamento das tabelas de pontos e da tabela de ocorrência dos mesmos (posição e canal onde tais pontos ocorrem) na música.

A *Kd-tree* se mostrou bastante rápida na recuperação de pontos das músicas (que representavam seqüências de notas musicais). A grande quantidade de pontos próximos, ou seja, similares, na árvore de indexação faz com que muitas vezes muitos resultados sejam recuperados. Porém, a ordenação dos mesmos, levando-se em conta a duração das notas da melodia, minimiza esse problema, já que os resultados mais relevantes (mais similares à consulta) aparecem no topo da lista de músicas recuperadas.

12. Referências

- Bentley, J.L. (1975) "Multidimensional binary search trees used for associative searching", In: Communications of the ACM, V. 18, I. 9, p. 509-517, September 1975.
- Cavalcanti, M. C., Machado, M. T., Cerqueira, A. & Araújo Jr., N. and Xexéo, G. (1999) "MIDIZ: Indexação e Recuperação Baseada em Conteúdo de Arquivos MIDI", In: Simpósio Brasileiro de Sistemas Multimídia e Hipermídia, SBMIDIA'99 - Goiânia, GO, Junho/1999.
- Faloutsos, C. (1996) "Searching Multimedia Databases by Content", Kluwer Academic, 1996
- Foskett D. J. (1997) "Readings in Information Retrieval", Morgan Kaufmann Publishers, Thesaurus. pag. 111-134
- Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995). "Design Patterns - Elements of Reusable Object Oriented Software", Addison-Wesley
- Heckroth J. (1995) "Tutorial on MIDI and Music Synthesis" - The MIDI Manufacturers Association
- Nievergelt, J. and Widmayer, P. (1997) "Spatial Data Structures Concepts and Design Choices", em "Algorithms Foundations of GIS", eds. van Kreveld, Nievergelt, Roos, Widmayer, C. 6, p. 153-198, LNCS 1340, Springer Verlag
- Selfridge E. (1993) "The MuseData Universe: A System of Musical Information". Computing in Musicology, V. 9, p. 9-30
- Selfridge, E. (1997) "Beyond MIDI The Handbook of Musical Codes". The MIT Press, Cambridge, Massachusetts
- Sourceforge: <http://hsqldb.sourceforge.net/>
- T. Chou, A. Chen and C. Liu (1996) "Music Databases: Indexing Techniques and Implementation", In: Anais do International Workshop on Multimedia Database Systems, p. 46-53
- W3: <http://www.w3.org/XML/>