Recuperação de informação com realimentação de relevância apoiada em visualização

Diogo Oliveira de Melo

Recuperação de informação com realimentação de relevância apoiada em visualização

Diogo Oliveira de Melo

Orientador: **Prof. Dr. Alneu de Andrade Lopes**

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências - Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*.

USP – São Carlos
Junho de 2014

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

M528r

Melo, Diogo Oliveira de
Recuperação de informação com realimentação de relevância apoiada em visualização / Diogo Oliveira de Melo; orientador Alneu de Andrade Lopes. -- São Carlos, 2014.
95 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2014.

1. Recuperação de Informação. 2. Exploração Visual. I. Lopes, Alneu de Andrade, orient. II. Título.

Agradecimentos

Agradeço aos meus pais Hermes e Sueli e ao meu irmão Denner, que sempre me apoiaram e foram pacientes comigo. Também à minha namorada, Simone, a quem eu tanto admiro, que sempre me ajudou, aconselhou e esteve ao meu lado.

Também sou muito grato ao meu Orientador Prof. Dr. Alneu Lopes, que confiou a mim este projeto de mestrado e sempre me guiou da melhor forma possível.

Agradeço ao Prof. Dr. Lucas Antiqueira por participar da banca de qualificação de mestrado e contribuir com importantes críticas para o amadurecimento deste trabalho. Também à Profa. Dra. Maria Cristina Ferreira de Oliviera por colaborar com este trabalho nas bancas de qualificação e de defesa e também por auxiliar na elaboração dos testes com usuários.

Gostaria de agradecer também ao Dr. Roberto Dantas de Pinho, pela prontidão em esclarecer dúvidas acerca do IncBoard e também pela valiosa contribuição na banca de defesa de mestrado.

Muito obrigado também ao Prof. Dr. Mário de Castro Andrade Filho por me ajudar a encontrar o caminho para a solução dos problemas de estatística relacionados a este trabalho.

Agradeço também ao Instituto de Ciências Matemáticas e de Computação pela oportunidade de partifipar do programa de mestrado e por financiarem minha participação no CIKM 2013.

Tenho muito a agradecer também à Seção de Pós-Graduação do ICMC pela extrema competência e boa vontade em tratar os assuntos relacionados ao meu mestrado.

Sou muito grato também à Profa. Dra. Roseli A. F. Romero por ter me apoiado durante a graduação e me dado a base do pensamento científico que uso desde então.

Não posso esquecer de agradecer meus colegas do Centro de Pesquisas Avançadas Wernher von Braun que, entendendo minha ambição pelo aprimoramento profissional e pessoal, permitiram que eu tivesse um horário de trabalho mais flexível afim de possibilitar o desenvolvimento deste mestrado.

Também sou muito grato ao pessoal da comunidade de *software* livre, que se envolveu ativamente no projeto, testando o sistema, traduzindo-o,

corrigindo-o, criando *layouts* novos e aproveitando partes do Amuzi em outros projetos. Desenvolvedores de software aberto e pesquisadores são os grupos para os quais tenho o mais alto respeito e admiração, pois são compostos de pessoas que desenvolvem o trabalho como uma forma de arte.

Por fim, agradeço à Universidade de São Paulo, que me formou profissionalmente por oito anos e me deu muitos amigos.

We build too many walls and not enough bridges.

Isaac Newton

Resumo

A mineração de grandes coleções de textos, imagens e outros tipos de documentos tem se mostrado uma forma efetiva para exploração e interação com grandes quantidades de informações disponíveis, principalmente na *World Wide Web*. Neste contexto, diversos trabalhos têm tratado de mineração tanto de coleções estáticas quanto de coleções dinâmicas de objetos. Adicionalmente, técnicas de visualização têm sido propostas para auxiliar o processo de entendimento e de exploração dessas coleções, permitindo que a interação do usuário melhore o processo de mineração (*user in the loop*). No caso específico de dados dinâmicos, foi desenvolvido por Roberto Pinho e colegas uma técnica incremental (IncBoard) com o objetivo de visualizar coleções dinâmicas de elementos. Tal técnica posiciona os elementos em um *grid* bidimensional baseado na similaridade de conteúdo entre os elementos. Procura-se manter elementos similares próximos no *grid*. A técnica foi avaliada em um processo que simulava a chegada de novos dados, apresentando iterativamente novos elementos a serem posicionados no mapa corrente. Observa-se, entretanto, que um aspecto importante de tal ferramenta seria a possibilidade de novos elementos – a serem exibidos no mapa, mantendo coerência com o mapa corrente – serem selecionados a partir do interesse demonstrado pelo usuário. Realimentação de relevância tem se mostrado muito efetiva na melhoria da acurácia do processo de recuperação. Entretanto, um problema ainda em aberto é como utilizar técnicas de realimentação de relevância em conjunto com exploração visual no processo de recuperação de informação. Neste trabalho, é investigado o desenvolvimento de técnicas de exploração visual utilizando realimentação de relevância para sistemas de recuperação de informação de domínio específico. O Amuzi, um sistema de busca de músicas, foi desenvolvido como uma prova de conceito para a abordagem investigada. Dados coletados da utilização do Amuzi, por usuários, sugerem que a combinação de tais técnicas oferece vantagens, quando utilizadas em determinados domínios. Nesta dissertação, a recuperação de informação com realimentação de relevância apoiada em visualização,

bem como o sistema Amuzi são descritos. Também são analisados os registros de utilização dos usuários.

Abstract

The mining of large text collections, images and other types of digital objects has shown to be a very effective way to explore and interact with big data, specially on the World Wide Web. On that subject, many researchers have been done on data mining of static and dynamic collections. Moreover, data visualization techniques have been proposed to aid on the understanding and exploration of such data collections, also allowing users to interact with data, user in the loop. On the specific subject of dynamic data, Roberto Pinho and colleagues have developed an incremental technique, called IncBoard, which aims to visualize dynamic data collections. IncBoard displays the documents on a two dimensional grid in a way that similar elements tends to be close to each other. This technique was evaluated in a process that simulated the arrival of new data elements, iteratively inserting new elements on the grid. Nonetheless, it would be useful if the user could interact with such documents to point out which are relevant and which are not relevant to his/her search. Relevance Feedback has also shown to be effective on improving the accuracy of Information Retrieval techniques. An issue that still open is how to combine data visualization and Relevance Feedback to improve Information Retrieval. On this dissertation, the development of techniques with data visualization and Relevance Feedback are investigated to aid on the Information Retrieval task, for specific domains. Amuzi is an Information Retrieval system, built to be a proof of concept for the investigated approach. Data collected from the usage of the system suggests that combining such techniques may outperform traditional Information Retrieval systems when applied for specifc domains. This dissertation has the description the information retrieval process with feedback relevance supported by visualization and the Amuzi system. Usage log are processed and analyzed to evaluate the investigated approach.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo

1

Introdução

Com o avanço tecnológico, a humanidade tornou-se capaz de acumular quantidades cada vez maiores de dados. Objetos de todos os tipos, como livros, textos, músicas, dados de georreferência, são gerados diariamente no formato digital. Tal aumento implica que além de ter métodos efetivos para armazenar estes dados, também são necessárias técnicas eficazes para recuperá-los. Boa parte destes dados estão disponíveis na Web. Mecanismos de busca catalogam a maior parte possível da Web e se esforçam para apresentá-los da forma mais eficiente e amigável possível, para o usuário.

Mecanismos de busca na Web são utilizados diariamente por mais de um bilhão de pessoas. De acordo com o *site* searchengineland (McGee, 2010), os três maiores mecanismos de busca – Google, Yahoo e Bing – respondem a aproximadamente três bilhões de pesquisas por dia. Melhorias no processo de busca podem ter impacto sobre todas estas pessoas.

1.1 Motivação

Os três motores de busca citados anteriormente são de domínio genérico, ou seja, tem o propósito de responder a qualquer requisição independente do assunto. Um outro tipo de mecanismo de busca é o de domínio específico. Como o nome já sugere, este tipo de mecanismo é limitado a determinado contexto. Imobiliárias, por exemplo, mantêm uma base de dados dos imóveis que estão abertos a negociação. A busca oferecida por estes

sites abrange apenas a base de dados local. De modo similar, existem livrarias, produtoras musicais, *sites* de vendas, de relacionamentos, etc.

Apesar de serem limitados pelo contexto, mecanismos de busca de domínio específico podem oferecer recursos que buscadores de domínio geral não podem, pois dispõem de mais informações sobre os dados que manipulam. Além disso, estes mecanismos já possuem um conhecimento à priori sobre qual é o alvo da busca do usuário. Em um sistema de busca de filmes, por exemplo, é possível presumir que o usuário está buscando por um filme, ator/atriz ou diretor.

De modo geral, em uma busca, o usuário não tem total conhecimento sobre o assunto que está pesquisando. À medida que o usuário interage com o mecanismo de busca, ele ganha informações sobre o assunto. Consequentemente, o alvo da busca é refinado à medida que o usuário ganha mais informações sobre o domínio. Outra característica das buscas é que o usuário potencialmente se interessa por resultados similares. Por exemplo, ao buscar por Cleópatra, o usuário também pode estar interessado em informações sobre Marco Antônio ou o Antigo Egito.

Buscadores tradicionais permitem que o usuário realize a busca através de palavras-chave. Entretanto, nem sempre é trivial expressar o conteúdo desejado deste modo. Além disso, duas buscas por conteúdos distintos podem ser expressas pelas mesmas palavras. Utilização de sistemas visuais pode ser uma alternativa para a resolução desta ambiguidade. Exploração visual de grandes coleções de documentos tem se mostrado uma forma efetiva para a mineração e interação com grandes quantidades de objetos. Diversos trabalhos têm tratado de mineração, tanto de coleções estáticas (Honorato, 2008; Lopes et al., 2007; Minghim et al., 2006; Paulovich et al., 2007) quanto de coleções dinâmicas de objetos (Chen, 2006; Pinho e de Oliveira, 2009; Robertson e Jones, 1976; Salton e Buckley, 1990; Silberschatz e Tuzhilin, 1995).

Adicionalmente, técnicas de visualização têm sido propostas para auxiliar o processo de entendimento e de mineração dessas coleções, permitindo que a interação do usuário melhore o processo de mineração (*user in the loop*) (Chen, 2006; Felizardo et al., 2010; Lopes et al., 2007; Lv e Zhai, 2009; Pinho e de Oliveira, 2009; Zhai e Lafferty, 2001). É necessário salientar que não existe mecanismo automático para determinar se o resultado retornado, pelo sistema de Recuperação de Informação (*Information Retrieval* – IR), corresponde às expectativas do usuário.

No caso específico de dados dinâmicos, Pinho et al. (2009) desenvolveram uma técnica incremental, IncBoard, com o objetivo de visualizar coleções dinâmicas de objetos. Tal técnica posiciona os elementos em um *grid* bidimensional baseado na similaridade de conteúdo entre os objetos. Nesta técnica, a disposição dos elementos obedece regras similares às de um tabuleiro de xadrez, em que cada célula é ocupada por não mais do

que um elemento. Procura-se manter próximos, no tabuleiro, documentos similares. Em um outro trabalho, Pinho et al. (2010) propôs o IncSpace. Esta técnica é similar ao IncBoard, mas os elementos não têm restrições relacionadas às células.

As técnicas propostas (Pinho et al., 2010; Pinho e de Oliveira, 2009) foram avaliadas em um processo que simulava a chegada de novos conjuntos de dados, apresentando iterativamente novos documentos a serem posicionados no mapa corrente. Observa-se, entretanto, que um aspecto importante de tal ferramenta é a possibilidade de novos documentos – a serem exibidos no mapa, mantendo coerência com a disposição atual dos objetos – serem selecionados a partir do interesse demonstrado pelo usuário. Tal interesse pode ser capturado via seleções de textos ou áreas no mapa, que mais refletem os objetivos do usuário utilizando técnicas de Realimentação de Relevância (*Relevance Feedback* – RF).

1.2 Objetivos, Métodos e Resultados Esperados

Este trabalho de mestrado tem o objetivo de desenvolver técnicas para mecanismos de busca de domínio específico. Mais precisamente, utilizar RF em conjunto com exploração visual para melhorar o processo de IR. Uma hipótese deste trabalho de mestrado é que a utilização de exploração visual com RF em um sistema de IR permite que o usuário explore melhor as informações fornecidas pelo sistema de IR. Portanto, é realizado um estudo de caso no contexto de busca de músicas. Foi construída uma ferramenta de busca que possui tanto IR com RF e exploração visual quanto IR tradicional. Desta forma é possível comparar as duas abordagens e entender a forma como exploração visual e IR podem auxiliar usuários a explorarem resultados.

Também faz parte do escopo deste trabalho de mestrado identificar as limitações da combinação das técnicas IR, RF e exploração visual. Desta forma é possível saber em quais situações este conjunto de técnicas é adequado para realizar a tarefa de IR.

Com o objetivo de medir a eficácia dos mecanismos de busca foram realizado testes com usuários. Através destes testes, é possível medir vantagens e desvantagens desta abordagem. Existem diferentes tipos de testes, desde testes realizados em laboratório até testes remotos e assíncronos. Cada tipo de teste endereça aspectos diferentes do projeto. No trabalho de Andreasen et al. (2007) é feito um estudo sobre os principais tipos de teste remotos de usabilidade, suas características e propósitos .

Por fim, este trabalho de mestrado também busca encontrar as condições que os dados devem satisfazer para que a utilização destas técnicas culmine em um mecanismo de IR eficiente.

1.3 Organização da Dissertação

Esta dissertação de mestrado está organizada da seguinte forma. O Capítulo 2 contém a revisão bibliográfica que aborda os temas de exploração visual, *crawlers* inteligentes, IR, RF e testes de usabilidade, que são assuntos importantes para este trabalho de mestrado. O Capítulo 3 contém a descrição dos métodos utilizados e as questões técnicas e científicas encontradas ao longo do desenvolvimento deste trabalho. O Capítulo 4 expõe e analisa os resultados obtidos. Por último, o Capítulo 5 contém as conclusões obtidas e as considerações finais.

Capítulo

2

Fundamentação: Conceitos Básicos

Exploração visual de dados, utilizando realimentação de relevância, envolve diversas áreas. Para tal, é necessário que todas as partes funcionem em sinergia, para que o sistema resultante responda em tempo real à interação do usuário. A mineração visual de dados está diretamente relacionada à interação do usuário com os objetos ou mapas gerados pela ferramenta de visualização. Ou seja, com a participação do usuário no processo de exploração e mineração em um cenário dinâmico. Neste cenário, a realimentação de relevância é responsável por melhorar o processo de recuperação de informação e apresentar, ao usuário, os dados disponibilizados por um *crawler* inteligente.

Nesta investigação de mestrado, algumas técnicas específicas são utilizadas para melhorar o processo de IR. A Figura 2.1 ilustra as interações entre os diferentes módulos que compõem o sistema proposto. O projeto do sistema depende do domínio dos dados e da fonte de informação utilizada. Caso já exista uma fonte de informações eficiente que disponibilize os dados de modo estruturado, o módulo de *crawler* pode ser desnecessário.

O módulo de exploração visual tem comunicação direta com o módulo de realimentação de relevância. Este é responsável por otimizar/refinar a requisição do usuário e enviá-la ao módulo de recuperação de informação. O módulo de recuperação de informação envia os objetos que casam com a requisição de volta para a exploração visual. O módulo de exploração visual atualiza a interface de acordo com os objetos retornados.

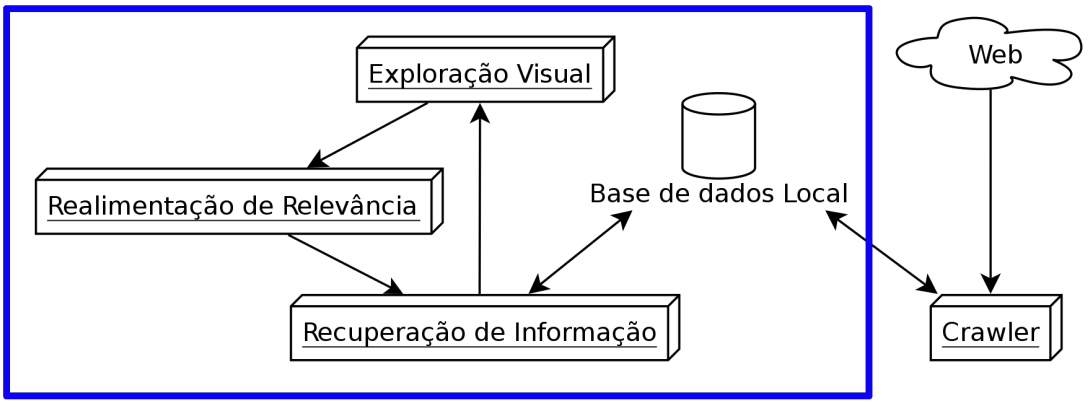

Figura 2.1: Módulos que compõem um sistema de recuperação de informação, com o foco de investigação deste mestrado dentro da área azul.

De forma assíncrona ao processo de interação com o usuário, o *crawler* rastreia a Web em busca de informações que sejam relevantes ao domínio. Esta informação é sumarizada e guardada na base de dados.

As seções seguintes descrevem em maiores detalhes os módulos que compõem o sistema. Na Seção 2.1 são abordadas técnicas de exploração visual. *Crawlers* inteligentes são tratados na Seção 2.2. Na Seção 2.3 é apresentado o funcionamento e as técnicas utilizadas em sistemas de recuperação de informação. Por fim, na Seção 2.4 são tratadas as principais técnicas utilizadas na construção de sistemas de realimentação de relevância.

2.1 Exploração Visual

Técnicas de visualização de informação têm um papel importante em aumentar o desempenho do processo de análise de documentos. A exploração visual tem o objetivo de inserir o usuário no processo de construção de um modelo mental adequado para um conjunto de dados em particular (Chen, 2006). Na exploração ou mineração visual de textos, abordagens multidisciplinares são reunidas para permitir que o usuário entenda a estrutura geral e tendências locais em conjuntos complexos de documentos.

Chen (2006) sugere que navegar por um espaço de informação visual, como pode ser o caso com exploração visual, requer que o usuário construa um mapa cognitivo interno, o que é similar a caminhar pelo mundo real. Então, é desejado ter visualizações onde o usuário seja capaz de estabelecer uma conexão com o mapa cognitivo, ao mesmo tempo evitando a complexidade inerente ao espaço de informação. Não é necessário que o mapa seja a representação de uma região da Terra. Este mapa de informação e espaço de navegação cognitivo é evidente em algumas técnicas de visualização do domínio de conhecimento, *Knowledge Domain Visualization* (KDViz). Um exemplo claro é usar mapas para visua-

lizar milhares de resumos de congressos. A Figura 2.2 ilustra um exemplo de um mapa formado a partir de um *corpus* contendo cerca de 600 artigos de três áreas de ciências da computação, *Case-based Reasoning*, *Inductive Logic Programming* e IR. O mapa foi gerado usando a ferramenta PEx (Paulovich et al., 2007), em que cada documento é representado por um pequeno círculo, as cores representam as três áreas e a distância entre os círculos reflete a similaridade entre os textos.

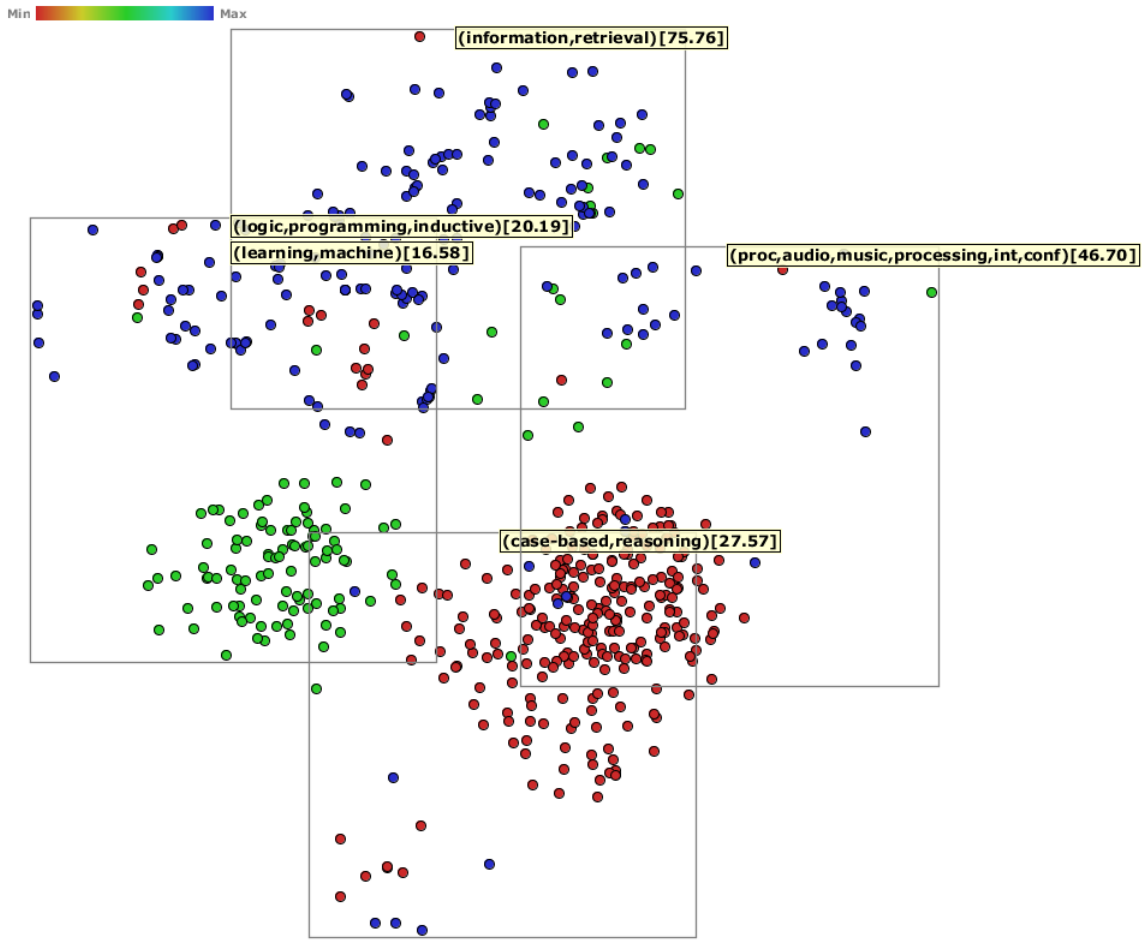

Figura 2.2: Exemplo de um mapa baseado no *corpus* CBR-ILP-IR.

Em um mapa como o da Figura 2.2, é importante que seja possível descobrir rapidamente o conteúdo dos grupos de documentos apresentados. Com esse objetivo, no trabalho de Lopes et al. (2007) é apresentada uma técnica para utilizar regras de associação para detectar e mostrar tópicos ou assuntos associado às diferentes regiões do mapa. Neste trabalho, são utilizados mapas topológicos – *Topic Maps* – uma técnica de visualização de dados apropriada para representar redes semânticas (Garshol e Moore, 2005).

Em trabalhos prévios do grupo no qual esta investigação será realizada, é feito mapeamento de textos através da redução de dimensionalidade combinado com técnicas de visualização para prover um sistema interativo que mapeie documentos científicos de áreas

diferentes (Lopes et al., 2006; Paulovich et al., 2007). Como pode-se observar na Figura 2.2, o mapa produzido foi capaz de separar os artigos pela área principal e também de aproximar artigos similares, demonstrando a eficiência da técnica utilizada.

Outra pesquisa propõe uma ferramenta para a representação de dados de alta dimensionalidade, o PEx (Paulovich et al., 2007). Esta ferramenta gera projeções 1D, 2D e 3D. As projeções são feitas através do mapeamento de espaço de alta dimensionalidade em espaços de poucas dimensões. Este mapeamento geralmente pode ser alcançado com redução de dimensão ou *clustering*. Projeções podem ser mostradas para usuários através de representações visuais, que podem variar de pontos em um plano até grafos, superfícies e volumes. O PEx trata grande quantidade de dados a um custo computacional adequado.

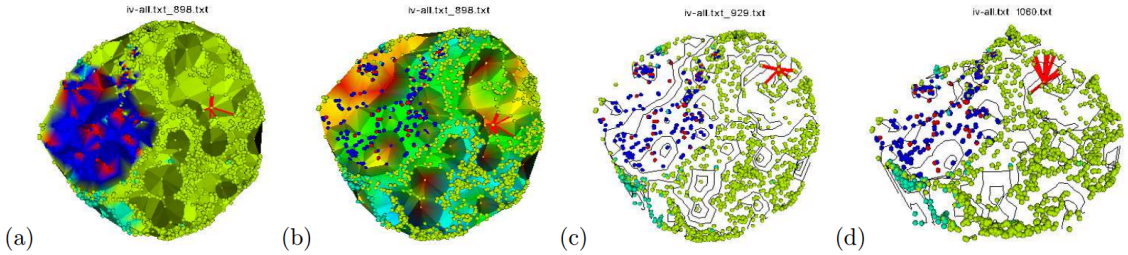

Figura 2.3: Imagens da técnica de visualização posposta por Minghim et al. (2006)

Em um outro trabalho são utilizadas técnicas de projeções de dados multidimensionais para prover um sistema que permite a visualização e a exploração das relações entre documentos (Minghim et al., 2006). A Figura 2.3 contém representações de tais documentos, que são agrupados por similaridade. Além disso, sistema permite que os usuários interajam com os dados a fim de descobrir relações e agrupamentos.

Figura 2.4: IncBoard – O mapa de imagens se adapta à chegada de novos elementos (Pinho et al., 2009).

Pinho et al. (2010, 2009) propuseram o IncBoard e o IncSpace que são técnicas que extraem características de um conjunto de documentos e posicionam suas representações

em um espaço bidimensional. Isto é feito de forma tal que elementos similares são mantidos próximos no mapa, à medida do possível. Estas técnicas permitem a mineração visual dinâmica de textos, pois incluem algoritmos para o reposicionamento dos documentos mediante chegada de novos elementos. O algoritmo também possibilita a remoção de documentos do *corpus* exibido.

As técnicas propostas por Pinho et al. (2009) podem ser utilizadas em um sistema de recuperação de informação. Não é necessário que o sistema de IR tenha todos os documentos a priori, este pode enviar novos documentos encontrados ao IncBoard, ou IncSpace, que rearranja o mapa para comportar os novos elementos. A Figura 2.4 é uma imagem gerada por uma implementação do algoritmo IncBoard.

O IncBoard mantém elementos similares próximos e seu algoritmo tenta minimizar a quantidade de elementos remanejados durante a inserção de um novo objeto. Nesta técnica de visualização, o espaço é organizado de forma similar a um tabuleiro de xadrez. Inicialmente, o tabuleiro possui zero elementos. Os elementos são inseridos um a um. O primeiro elemento é inserido no centro do tabuleiro. A partir do segundo elemento, novos elementos são posicionados na mesma célula que o elemento mais similar ao elemento a ser inserido.

O IncBoard possui o conceito de estabilidade. O tabuleiro está no estado estável quando não há mais de um elemento em nenhuma das células. O tabuleiro está no estado instável quando há uma célula com dois elementos. O modo como o IncBoard reage ao estado instável torna impossível que haja mais de uma célula com mais de um elemento cada ou uma célula com mais de dois elementos.

Com exceção do primeiro elemento, a inserção de um novo elemento inevitavelmente leva o tabuleiro ao estado instável. Para tratar o estado instável e trazer o tabuleiro de volta à estabilidade uma ação deve ser tomada. Esta ação consiste em decidir qual dos dois elemento da célula que causa a instabilidade deve ser movido para uma das oito células vizinhas. Se, após mover um dos elementos para a célula vizinha, o tabuleiro continuar instável então a mesma ação deve ser executada, recursivamente, até que o tabuleiro esteja no estado estável.

A medida em que o sistema caminha em sua recursão, é mantida uma lista de células que já foram visitadas. Com o objetivo de evitar recursão infinita, elementos não podem ser movidos para células que já foram visitadas.

$$W_{err}(E_i, L) = \sum_{E_j \in L} |R_{ci}(E_j) - R_{ni}(E_j)| \times (|L| - R_{ni}(E_j)) \tag{2.1}$$

Para decidir qual dos dois elementos deve ser movido para qual das oito células vizinhas, é necessários considerar o erro W_{err} obtido através da Equação 2.1. Esta equação baseia-se na diferença entre dois *rankings*.

$$d_c(E_i, E_j) = max\{|x_i - x_j|, |y_i - y_j|\} \tag{2.2}$$

O *ranking* R_{ci} considera a distância de Chebychev, dada pela Equação 2.2, consiste no número de movimentos que são necessários para caminhar de uma célula a outra. O *ranking* R_{ci} é tal que elementos próximos à E_i estão no topo do *ranking* enquanto elementos distantes ocupam posições mais baixas no *ranking*.

O *ranking* R_{ni}, por outro lado, não leva em consideração as limitações de representação da visualização. Este *ranking* é baseado em uma medida de similaridade que deve ser intuitiva ao usuário. Quanto mais similares os elementos E_i e E_j, maior será a posição do elemento E_j no *ranking* R_{ni}.

Na Equação 2.1, L representa o conjunto de elementos presentes no tabuleiro. O somatório calcula a distância entre este dois *rankings*. Desta forma, quanto maior W_{err}, mais distante a visualização está de representar a similaridade entre os elementos.

Desta forma, o valor de W_{err} é calculado para avaliar cada uma das 16 possibilidades, mover o novo elemento para uma das oito células vizinhas, ou mover o antigo ocupante da célula para uma das oito células vizinhas. A solução que apresentar o menor W_{err} será a escolhida.

2.2 Crawlers Inteligentes

Crawlers são robôs que fazem acessos sistematizados à Web (Brin e Page, 1998a,b). Esses robôs acessam o conteúdo de uma página e são utilizados para dois propósitos. O primeiro é a recuperação da informação em si, em que a página é armazenada integralmente ou processada primeiro e o conteúdo sintetizado é armazenado, para que possa ser acessada facilmente por outros módulos no sistema. O segundo é a extração de *links* que apontam para outras páginas. Os *links* encontrados são os alvos armazenados para que o *crawler* possa acessá-los futuramente.

Com o crescente avanço da informatização e digitalização de documentos, mais e mais conhecimento se torna disponível na Web. Ao mesmo tempo, com a popularização da Internet e ascensão de novas tecnologias, a rede mundial de computadores fica maior, mais complexa e, por consequência, mais difícil de ser rastreada (Baeza-Yates e Ribeiro-Neto, 2011).

Muitos esforços já foram empregados em pesquisas para indexar a Web. Estes esforços geraram excelentes frutos como os buscadores providos pelo Google[1], Yahoo[2] e Microsoft[3], dentre outros. A utilização de *crawlers* se tornou um processo tão bem difundido na indústria de sistemas Web que as entidades que provêem as páginas e os sistemas Web disponibilizam informações que facilitam o processamento pelos *crawlers*. Para muitos casos, a implementação de um novo *crawler* pode ser evitada. Mecanismos de busca, como o oferecido pelo Google[4], oferecem APIs (*Application Platform Interface*) para realização de buscas. Também há implementações em código livre de *crawlers*, como o Sphinx[5]. Além disso, é possível encontrar documentação na Internet sobre como implementar *crawlers*[6].

Segundo Castillo (2005), projetos de *crawlers* se diferenciam basicamente em quatro pontos:

- uma política que determina a ordem com a qual as páginas devem ser examinadas;
- política de atualização das páginas já visitadas;
- evitar que sistemas Web sejam sobrecarregados pela ação do *crawler*;
- política de paralelização para coordenar *crawlers* distribuídos.

Tais tópicos são tratados nas subseções 2.2.1, 2.2.2, 2.2.3 e 2.2.4.

Devido ao grande volume de dados espalhado pela Internet, não é viável armazenar todos os arquivos. É necessário processá-los e armazenar apenas o necessário para o funcionamento do módulo de IR. Com o objetivo de diminuir a quantidade de dados a serem mantidos em disco rígido, é possível utilizar técnicas de preprocessamento de textos, tal utilização é abordada na Subseção 2.2.5.

2.2.1 Política de visitação de páginas

Existem dois algoritmos principais que determinam a ordem em que as páginas são acessadas pelo *crawler*. A primeira é a busca em largura e a segunda, busca em profundidade. Na primeira estratégia, o *crawler* lista todos os *hyperlinks* da páginas e os coloca em uma fila. A página seguinte a ser processada será a primeira página da fila. Na busca

[1] `http://google.com`

[2] `http://search.yahoo.com/`

[3] `http://bing.com`

[4] JSON/Atom Custom Search API - JSON/Atom Custom Search API - Google Code – `http://code.google.com/apis/customsearch/v1/overview.html`, Online; acessado em 9-Março-2011

[5] WebSPHINX: A Personal, Customizable Web Crawler – `http://www-2.cs.cmu.edu/~rcm/websphinx/`, Online; acessado em 9-Março-2011

[6] Writing a Web Crawler in the Java Programming Language – `http://java.sun.com/developer/technicalArticles/ThirdParty/WebCrawler/`, Online; acessado em 9-Março-2011

em profundidade, é utilizada uma estrutura de pilha para obter a próxima página a ser visitada.

Outra tarefa do *crawler* é a de organizar o conteúdo que foi obtido através da Web e armazená-lo na base de dados. Entretanto, devido ao enorme volume de dados, não é possível armazenar este conteúdo integralmente. É necessário utilizar técnicas de mineração de texto para extrair o que for mais relevante sobre o conteúdo e armazená-lo na base de dados. A Subseção 2.2.5 traz técnicas de mineração de textos que podem ser empregadas nesta tarefa.

2.2.2 Política de atualização das páginas

Ainda para sistemas de grande porte, rastrear toda a Web é algo que leva algo na ordem de semanas ou meses. Neste período, a Web em si já mudou bastante. Assim, é necessário ter uma política de atualização das páginas que possibilite que o *crawler* esteja com uma visão atualizada da Web. Uma opção é revisitar todos os *links* com igual prioridade sem considerar a taxa com a qual costumam ter seu conteúdo alterado ou sem considerar a importância.

A política proporcional envolve atualizar os links de acordo com as estimativas sobre a taxa com a qual sofrem mudanças. O *crawler* busca atualizar primeiro os links cujo conteúdo muda com maior frequência. O problema com esta opção é que o sistema desperdiça muito esforço tentando atualizar páginas que mudam com muita frequência e, apesar dos esforços, estas páginas continuarão desatualizadas na base de dados.

2.2.3 Política de bons modos

Em um estudo feito por Koster (1995), foi apontado que apesar de *crawlers* serem úteis, isto consome muito processamento por parte dos sistemas que são rastreados:

- recursos de rede. Um robô requer uma quantidade considerável de largura de banda para visitar as páginas;
- sobrecarga de sistema. *Crawlers* podem acabar em páginas que apontam para outras páginas no mesmo sistema e visitar uma grande quantidade de páginas que está hospedada no mesmo servidor. Isso pode levar a sobrecarga do sistema;
- robôs pessoais que, se forem utilizados por muitos usuários, podem sobrecarregar o sistema.

O mesmo autor que identificou estes problemas em 1995 propôs uma solução, que é a criação de um protocolo que os *webmasters* poderiam utilizar para indicar aos robôs quais partes do sistema não deveriam ser visitadas (Koster, 1996).

Entretanto, prover um modo para administradores excluírem parte do sistema é apenas parte da solução. Outra parte é os robôs estabelecerem um intervalo entre duas conexões consecutivas ao mesmo servidor. Mas estabelecer um tempo fixo entre os acessos a um sistema pode não ser a melhor opção. Da mesma forma que os robôs rastreiam sistemas pequenos, com apenas algumas dezenas de páginas, também rastreiam grandes *sites*, com conteúdo dinâmico que produzem milhares de páginas por dia.

O robô "Mercator Web"(Heydon e Najork, 1999) usa uma abordagem adaptativa. Se são necessários t segundos para obter uma página, então espera $15 \times t$ para pegar a próxima página, ou seja, o tempo de espera para fazer *downloads* de um sistema é proporcional a latência do mesmo.

2.2.4 Política de paralelização

O objetivo de paralelizar é aumentar a capacidade de *downloads* do sistema. Entretanto, é necessário minimizar *overheads* derivados da paralelização e também evitar pegar a mesma página repetidas vezes.

Com a paralelização, surge um novo problema: distribuir as URLs dentre os processos que estão, em paralelo, fazendo *downloads* de páginas. Duas abordagens são descritas por Castillo (2005). A primeira, atribuição dinâmica, consiste em ter um sistema centralizado que distribui as URLs a serem baixadas. Entretanto, quando o número de URLs a serem distribuídas é muito grande, pode acontecer do sistema centralizado ficar sobrecarregado.

A segunda abordagem, atribuição estática, consiste em definir as URLs que cada *crawler* baixará através de um método que não requer a coordenação de um sistema central. Por exemplo, utilizar uma função de *hash*[7]. Caso N *crawlers* estejam baixando páginas, então dada URL deve ser baixada pelo *crawler* $hash(URL) mod N$.

A medida que novas URLs são descobertas há a necessidade dos processos se comunicarem para trocar URLs descobertas. Isso deve acontecer em *batch*. Várias URLs devem ser enviadas de uma só vez para não sobrecarregar os processos.

2.2.5 Pré-processamento de Textos

O processo de mineração de textos (*Text Mining* – TM) pode ser construído adjacente ao *crawler*. Tal processo consiste em extrair informações úteis de um conjunto de textos. O processo de extração compreende a modelagem do problema, o pré-processamento do conjunto de textos, a extração dos padrões e a utilização do conhecimento (Rezende, 2005).

[7]Funções *hash* são funções que mapeiam um conjunto de valores (chaves) em um outro conjunto bem definido de valores. Duas chaves diferentes devem gerar valores *hash* diferentes

No pré-processamento a coleção de textos passa a ser representada por uma matriz atributo/valor. Cada linha dessa matriz representa um documento. Cada coluna representa um atributo. Uma célula possui o valor que indica a relação entre um atributo e um documento. A montagem da matriz pode ser dividida em quatro etapas, conforme descrito por Rezende (2005): Seleção do *corpus*, Geração de Atributos, Montagem da Matriz e Seleção de Atributos.

A obtenção do *corpus* pode ser feita através de *crawlers*. Os documentos devem ser convertidos para texto plano. O *crawler* pode ter recuperado um conjunto heterogêneo de documentos em diferentes formatos, como formato de documento portável (*Portable Document Format* – PDF) e linguagem de marcação de super texto (*HyperText Markup Language* – HTML), por exemplo. A seleção do *corpus* é feita de acordo com os requisitos da aplicação, por exemplo, se o objetivo é obter artigos científicos, então é necessário filtrar a saída do *crawler* para que apenas arquivos dessa natureza façam parte do *corpus*.

Para a geração de atributos é necessário reconhecer, no texto, palavras que não têm relevância e também reconhecer palavras que são morfologicamente iguais (Rezende, 2005). Para reduzir a dimensionalidade da matriz, é necessário selecionar os atributos que melhor representem o *corpus*. A seleção de atributos pode ser feita através da técnica de agrupamento de palavras, *term clustering* (Slonim e Tishby, 2000).

A matriz atributo/valor resultante da seleção de atributos é utilizada na extração de padrões. Este passo consiste na definição, configuração e execução de algoritmos para a descoberta de padrões no texto (Álvares, 2007). Estes algoritmos transformam os dados textuais em padrões que podem ser interpretados por humanos (Kantardzic, 2002). Nesta etapa podem ser empregados algoritmos de diversas áreas do conhecimento, como Aprendizado de Máquina, Estatística, Redes Neurais e Base de Dados.

Segundo Silberschatz e Tuzhilin (1995), o pós-processamento é a etapa responsável por fornecer ao usuário apenas os padrões mais interessantes. O resultado desta etapa pode ser tanto textual quanto gráficos, expressos em forma de árvore de diretórios ou árvore hiperbólica (Lamping et al., 1995) e outras técnicas de visualização já citadas, ainda no pós-processamento. Uma das últimas etapas está relacionada a avaliação da qualidade dos padrões extraídos, tarefa que varia de acordo com os algoritmos e domínios em questão (Monard e Baranauskas, 2003; Silberschatz e Tuzhilin, 1995).

2.3 Recuperação de Informação

IR é o processo pelo qual é possível recuperar uma porção específica de informação no sistema que possui toda a informação a ser coletada. Registros de armazenamento de

informação de maneira organizada, facilitando o processo de recuperação, remontam a 3.000 A.C., quando os sumérios utilizavam tábuas para organizar informações.

Através dos séculos, a quantidade de informações disponíveis aumentou significativamente. Com o surgimento do computador, a tarefa de organizar grandes volumes de dados em informações passou a ser mais plausível. Segundo Gantz e Reinsel (2012), a quantidade de dados armazenados de forma digital dobra a cada dois anos. A pesquisa prevê que o volume de dados atingirá $40.000 exabytes$ em 2020.

Dentro deste contexto, vários modelos de IR foram propostos (Singhal, 2001). Os principais modelos são:

- modelo Booleano;
- modelo de Espaço Vetorial;
- modelo Probabilístico;
- PageRank.

As próximas subseções detalham esses modelos.

2.3.1 Modelo Booleano

O modelo booleano foi um dos primeiros modelos de sistemas de IR (Lancaster e Fayen, 1973). Este modelo se caracteriza pela utilização da combinação das operações booleanas *AND*, *OR* e *NOT*. Estas expressões são utilizadas para permitir que o usuário expresse quais termos os documentos devem ter, quais termos são opcionais que o documento tenha e quais termos não devem estar inclusos no documento. O maior problema com esta estratégia é que os documentos não são ordenados por relevância. Ficava a cargo do usuário determinar o que era relevante dentre os objetos retornados pela consulta.

Uma desvantagem deste modelo é que, especialmente em aplicações Web, é difícil evitar que documentos irrelevantes apareçam nos resultados das pesquisas. Pessoas tentam fazer com que seus *sites* tenham uma boa visibilidade na Web com o objetivo de atrair mais visitas. Sistemas que empregam o modelo booleano são mais suscetíveis a este tipo de engano. Uma página que possua um grande conjunto de palavras chave, por exemplo, mas que não tenha efetivamente nenhum conteúdo interessante pode aparecer próxima a um *site* com informações relevantes.

Este modelo deve ser utilizado apenas quando a origem dos dados a serem buscados é de confiança e em quantidade controlada.

2.3.2 Modelo de Espaço Vetorial

No modelo Espaço Vetorial (*Vector Space*) (Salton et al., 1975), cada documento é tratado como um vetor. Cada dimensão deste vetor é um termo dentre o conjunto de todos os possíveis termos. O valor zero em uma dessas dimensões indica que o documento não possui o termo relacionado a determinada dimensão. Considerando que o total de termos abordados na união de todos os documentos é da ordem de milhões, é comum que o vetor de cada documento seja esparso.

É possível medir similaridade entre dois documentos através de medidas como cosseno e o produto escalar. Considerando que os valores das dimensões do vetor são não negativos, o cosseno será sempre um valor entre 0 e 1.

2.3.3 Modelo Probabilístico

O modelo probabilístico foi proposto por Robertson e Jones (1976) e tem como ponto central a probabilidade de dado documento ser relevante a uma *query*. O método parte da hipótese que para toda *query* existe um conjunto que engloba todos os documentos relevantes a esta *query* e apenas os relevantes. Este conjunto ideal seria a resposta perfeita que o sistema de IR poderia fornecer.

Entretanto, não é definida a forma de calcular tais probabilidades. Além disso, não há como calcular estas probabilidades com exatidão. Diferentes implementações/técnicas utilizam meios diferentes para estimar as probabilidades das relevâncias dos documentos.

Neste modelo, inicialmente os documentos possuem probabilidades iniciais relacionadas às *queries*. Após um usuário fazer uma requisição e obter um conjunto de resultados, é requerido que ele aponte documentos relevantes e irrelevantes. O resultado desta interação com o usuário é utilizado para ajustar as probabilidades associadas à requisição feita pelo usuário. Este reajuste terá efeito na próxima vez em que algum outro usuário fizer a mesma requisição.

Este modelo faz algumas suposições. Uma delas é chamada de princípio probabilístico e diz que dada uma *query* q de um usuário e um documento d_j, pertencente a coleção de documentos, o modelo tenta estimar se o documento d_j é ou não relevante. Este modelo assume que a probabilidade depende apenas da *query* e do documento em questão. Mais do que isso, o modelo assume que há um subconjunto de documentos que é considerado relevante pelo usuário.

A principal vantagem desta modelagem é que os objetos são ordenados em ordem decrescente da probabilidade de serem relevantes, sendo uma solução ao problema de se ter muitos documentos como possível resposta a uma requisição. Dentre as desvantagens, é possível citar:

- necessidade de ter os valores iniciais sobre quais documentos são e quais não são relevantes;

- o método não considera a frequência de um termo dentro do documento. É considerado apenas se um termo está ou não presente no documento;

- a adoção da suposição de que a ocorrência dos termos dentro do documento são eventos independentes.

2.3.4 PageRank

O PageRank foi proposto por Brin e Page (1998a). Na época, havia um grande problema em identificar quais páginas eram relevantes. Com o crescimento da Internet, as buscas dos usuários retornavam centenas de milhares de páginas, das quais poucas tinham valor para o usuário. Identificar as páginas relevantes ao usuário e colocá-las no topo dos resultados era uma tarefa que outros mecanismos de busca (Ask, AltaVista e Yahoo, por exemplo) já tentavam resolver, sem muito sucesso.

O algoritmo PageRank parte da ideia de que os *hyperlinks* que páginas usam para apontar para outras páginas podem ser utilizados como uma medida de relevância. Páginas relevantes tendem a ser muito referenciadas através de links.

O PageRank pode ser expressado pela Equação 2.3, proposta por Brin e Page (1998a). $PR(u)$ é o valor do PageRank da página u. B_u é o conjunto de páginas que aponta para a página u. $L(v)$ é o número de páginas para a qual a página v aponta. Por fim, d é um valor entre 0 e 1 que representa o fator de amortecimento.

$$PR(u) = (1 - d) \times d(\sum_{v \in B_u} \frac{PR(v)}{L(v)}) \tag{2.3}$$

Entretanto, implementação do algoritmo possui dois passos. No primeiro, é atribuído um valor fixo a todas as páginas. O segundo ocorre durante o rastreamento das páginas. O valor do PageRank é calculado de acordo com o valor da Equação 2.3.

2.4 Realimentação de Relevância

A RF parte da hipótese de que é difícil, para o usuário, formular uma boa requisição, pois nem sempre é trivial escolher um conjunto de palavras-chave para representar um tema. Problemas de ambiguidade são frequentes. Entretanto, é fácil avaliar se os documentos retornados estão de acordo com o interesse de quem está buscando. Ao apresentar objetos ao usuário, é fácil para ele identificar o que é e o que não é interessante à busca.

Esta técnica tem como objetivo aumentar o desempenho do processo de recuperação de informação. RF provou ser muito eficiente em aumentar a precisão do processo de IR (Lavrenko e Croft, 2001; Robertson e Jones, 1976; Rocchio, 1971; Salton e Buckley, 1990; Zhai e Lafferty, 2001). A abordagem tradicional foi proposta por Rocchio (1971) e é chamada de *query expansion* (expansão da requisição), na qual a consulta inicial do usuário é agregada aos termos comuns dos documentos considerados relevantes e são retirados os termos pertencentes aos documentos explicitamente considerados irrelevantes.

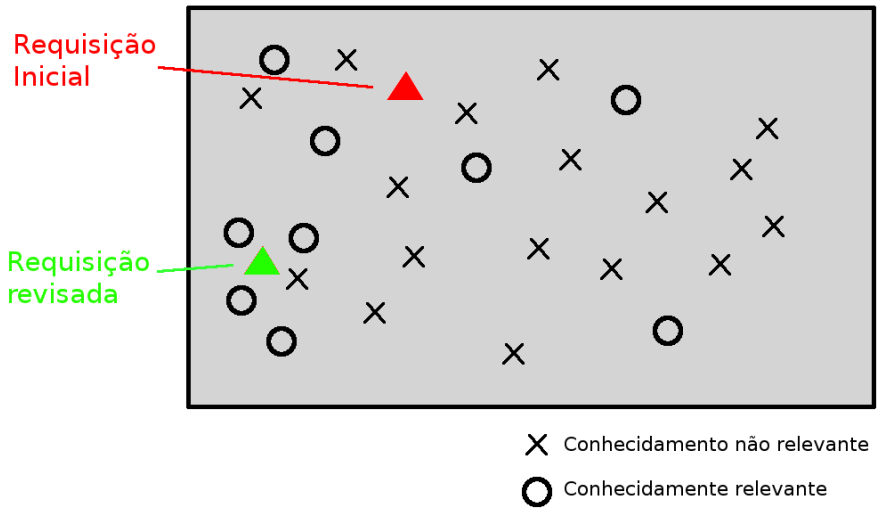

Figura 2.5: Uma aplicação do algoritmo de Rocchio. Alguns documentos foram rotulados relevantes e irrelevantes. O vetor de requisição inicial foi movido após o *feedback* do usuário.

A requisição inicial do usuário não é ótima, pois leva tanto a documentos relevantes quanto a documentos irrelevantes, conforme ilustrado na Figura 2.5. Quando termos são adicionados e removidos, de acordo com o *feedback* do usuário, a requisição é movida para um local mais próximo de documentos relevantes e mais distante de documentos irrelevantes.

$$\vec{q_m} = \alpha\vec{q_0} + \beta\frac{1}{|D_r|}\sum_{\vec{d_j}\in D_r}\vec{d_j} - \gamma\frac{1}{|D_{nr}|}\sum_{\vec{d_j}\in D_{nr}}\vec{d_j} \tag{2.4}$$

A nova requisição é calculada pela Equação 2.4 (Rocchio, 1971). O vetor $\vec{q_0}$ contém os termos da requisição inicial do usuário. As constantes α, β e γ são pesos que ponderam a influência da requisição inicial, os termos presentes nos documentos relevantes, e os termos presentes nos documentos irrelevantes, respectivamente. $\vec{q_m}$ é o vetor com os termos que serão utilizados na requisição seguinte, no algoritmo de Rocchio.

Considerando novamente a Figura 2.5, a nova requisição $\vec{q_m}$ é representada pela indicação "Requisição revisada". A requisição revisada está mais próxima de documentos

relevantes e mais distante de documentos irrelevantes. Utilizando este procedimento, o número de resultados relevantes para a nova busca tende a ser maior. Documentos relevantes que foram ocultados por documentos irrelevantes tem maior probabilidade de serem mostrados em uma posição melhor da lista de resultados.

A técnica de RF possui basicamente três formas:

- abordagem tradicional;
- realimentação de Relevância Implícita;
- pseudo Realimentação de Relevância.

As três abordagens se distinguem basicamente pelo nível de interação com o usuário e são abordadas individualmente nas subseções seguintes.

2.4.1 Abordagem Tradicional

A realimentação de relevância tradicional refere-se ao processo interativo de melhorar a recuperação de informação no cenário em que o usuário envia uma requisição. De acordo com Manning et al. (2008), o algoritmo que descreve a abordagem tradicional de RF é o seguinte:

1. usuário provê uma requisição curta e simples;
2. sistema retorna um conjunto de resultados;
3. usuário informa quais documentos são relevantes ou irrelevantes;
4. repete os passos 2 e 3 até que o usuário esteja satisfeito.

Um ponto importante a salientar é que, à medida que o usuário tem contato com outros documentos, ele se torna mais apto a refinar o que deseja buscar. Se o usuário está buscando por algo, é de se esperar que não tenha total domínio sobre o assunto. O usuário está muito propenso a aprender não apenas após encontrar o documento desejado, mas também durante o processo de busca.

Ainda na abordagem tradicional existem variações em relação ao modo como o usuário provê a alimentação. Dentre as formas possíveis, pode-se citar:

- informar apenas quais documentos são relevantes;
- informar quais são e quais não são relevantes;

- avaliar a relevância do documento de acordo com uma nota (entre 0 e 5, com 5 sendo muito relevante, por exemplo);
- utilizar graus de relevância, por exemplo: "Muito relevante", "relevante", "pouco relevante", "irrelevante".

A característica que distingue esta forma das demais é que o usuário está efetivamente envolvido no processo de RF. Opcionalmente, o sistema pode guardar o julgamento do usuário para interações futuras com outros usuários, evitando que documentos não importantes sejam supervalorizados e que documentos importantes sejam tratados com a mesma relevância que outros. Isto cumulativamente agrega informação ao sistema acerca da importância dos documentos.

Existem várias abordagens para tratar a indicação de relevância do usuário e uma delas é de expansão da requisição(Miller et al., 1999). Esta técnica consiste em utilizar a realimentação do usuário para concatenar elementos à última *query* utilizada, fazendo com que a busca se torne cada vez mais específica.

RF tem sido extensivamente estudada e a maioria dos métodos tratam o problema como aprendizado de máquina supervisionado com tratamento especial às requisições (Robertson e Jones, 1976; Rocchio, 1971; Salton e Buckley, 1990; Zhai e Lafferty, 2001). Tratar a relação entre as requisições e os documentos de resposta tem sido um problema difícil e importante. É necessário ter cuidado em relação à confiança depositada no julgamento do usuário. Se a confiança for além do ponto ótimo, então as requisições serão tendenciosas. Se for para o outro extremo, dar pouca relevância ao julgamento do usuário, o sistema de RF não traria muita vantagem. Y. Lv e Zhai (2009) tratam o problema de balancear a importância da relevância provida pelo usuário com a requisição inicial.

2.4.2 Realimentação de Relevância Implícita

A RF implícita é um método menos invasivo de coletar respostas do usuário. Corresponde a analisar quais documentos o usuário visitou após a consulta e quanto tempo permaneceu na página resultado, por exemplo. Basicamente este método tenta aferir o quanto de importância o usuário deu a cada resultado sem ter que explicitamente requisitar este *feedback*.

Por ser implícito, o usuário pode não saber que está utilizando um sistema de RF. Outra característica importante deste método é que o usuário não se beneficia diretamente de seus *feedbacks*. As realimentações são utilizadas em uso posterior, não necessariamente pelo mesmo usuário.

Em muitos casos, o usuário é relutante em prover *feedback*, então é mais fácil coletar realimentação pelo método implícito. Por não exigir nenhum esforço adicional por parte

do usuário, é um tipo de realimentação que pode gerar maior quantidade de dados. Fato é que o *feedback* pode não ser tão preciso quanto na abordagem tradicional. Entretanto, a maior quantidade de amostras é um benefício e favorece uma estimativa mais precisa.

Segundo Manning et al. (2008), o *site* DirectHit[8] introduziu a ideia de atribuir a relevância de cada página a uma requisição baseado no número de cliques que o resultado consegue para a requisição. Quanto mais cliques uma página consegue, maior sua relação com o conjunto de termos que gerou os resultados.

2.4.3 Realimentação de Relevância Cega

Também existe a realimentação de relevância cega. É chamada desta forma pois não envolve interação com o usuário no processo de refinamento dos termos. A RF cega faz uma primeira busca utilizando os termos fornecidos pelo usuário. São extraídos termos das primeiras páginas retornadas nesta busca. Os termos são então agregados e é feita uma segunda busca. Apenas os resultados desta segunda busca serão apresentados ao usuário.

No Trabalho de Yu et al. (2003), é proposto o VIPS (VIsual-based Page Segmentation), que utiliza realimentação de relevância cega para aumentar a qualidade de recuperação de informação em páginas Web. Neste contexto, partes do documento não agregam informações semânticas, como: estrutura de navegação, decoração, interação (código que tem por objetivo tornar a página interativa), palavras especiais (*copyright* e informações de contato). Após remover toda a informação irrelevante ainda é necessário ter atenção ao fato de que frequentemente a mesma página trata de tópicos diferentes.

A técnica VIPS utiliza uma combinação entre a estrutura do documento e indícios visuais para o processo de segmentação. A página Web é segmentada em blocos e a busca é endereçada a estes blocos e não necessariamente a todo o documento, de tal forma que quando os termos para a segunda etapa de busca forem escolhidos, o risco de irrelevantes a busca é minimizado.

2.5 Teste de Usabilidade

Testes de usabilidade tem por objetivo observar pessoas utilizando um determinado produto e avaliar este produto através das interações observadas. Os resultados desta avaliação podem ser utilizados para aperfeiçoar o produto ou para compará-lo com outros. De acordo com Nielsen (1993), testes de usabilidade visam medir quatro aspectos:

[8] `http://directhit.com`

- performance – quantidade de passos ou tempo necessário para concluir tarefas utilizando o produto;
- acurácia – razão entre a quantidade de movimentos corretos feitos e o total de movimentos feitos pelo usuário;
- *Recall* – o quanto o usuário ainda lembrará após períodos sem utilizar o produto;
- resposta emocional – o sentimento relacionado a concluir as tarefas. Isto pode incluir confiança, estresse, dentre outros.

Há vários parâmetros a serem definidos no planejamento do teste de usabilidade. O teste pode ser feito em laboratório ou aplicado remotamente. Sendo um teste remoto, pode ser síncrono ou assíncrono. Outro parâmetro importante é o número de usuários que irão participar dos testes e o conhecimento que estes usuários possuem acerca de computação. Outra questão importante é se o usuário já teve contato com o sistema, antes do teste. A aptidão do usuário com sistemas de computação também é algo a ser considerado.

Andreasen et al. (2007) explica que realizar o teste em laboratório possui a vantagem de permitir controlar várias condições de ambiente. Desta forma, é possível fazer estas condições serem constantes a todos os usuários que participam do experimento e agregar maior confiabilidade aos resultados. Testes remotos tendem a gerar resultados com mais ruídos, pois o controle sobre as condições em que o teste ocorre é menor e estas condições serão diferentes para cada usuário que participar do teste.

Dentre os parâmetros que podem variar estão: sistema operacional, navegador, resolução da tela, quantidade de bits para representação de cores, luminosidade do ambiente, poder computacional da máquina usada, interrupções causadas por pessoas fisicamente próximas e qualidade da conexão com a Internet.

Apesar de se ter menor controle sobre o ambiente, testes remotos podem ser mais realísticos pois não distorcem o ambiente que o usuário de fato usa para acessar a Internet. Os testes remotos podem ser divididos em duas categorias: síncronos e assíncronos. Nos testes síncronos o avaliador e o usuário estão participando da execução do teste, apesar de estarem em localidades diferentes. Testes síncronos, em geral, tentam reproduzir as condições do laboratório através da comunicação com áudio e vídeo e compartilhando a tela do usuário.

Testes síncronos, que imitam as condições do laboratório, tendem a ter menor custo em termos de equipamento. Entretanto, o processo de configurar todas as ferramentas necessárias no sistema do usuário pode ser demorado e frustrante.

Testes remotos assíncronos por outro lado, não requerem a presença do avaliador durante a execução do teste. Apesar da quantidade de erros descobertos por um usuário,

em média, ser menor, este teste pode ser aplicado a um número maior de usuário a um menor custo.

Capítulo

3

Recuperação de Informação com Realimentação de Relevância apoiada em Visualização: o Sistema Amuzi

O foco principal deste trabalho de mestrado é a exploração de recuperação de informação com realimentação de relevância apoiada por visualização. Assim, o foco é não apenas o estudo e desenvolvimento de cada uma destas técnicas em si, mas também a análise de como estas técnicas se comportam na composição de um sistema, ou seja, a investigação das sinergias entre visualização, RF e IR.

Além disso, este trabalho não está estritamente vinculado a um tema específico. Espera-se que os resultados obtidos possam ser aproveitados para diversas finalidades, como por exemplo, na busca de músicas, artigos científicos, imagens, produtos, entre outros.

Entretanto, considerando que estas técnicas são efetivamente implementadas em um sistema real, é necessário escolher um tema para o sistema. O sistema desenvolvido foi feito com o objetivo de facilitar a recuperação de informações em base de dados de música. O usuário é capaz de buscar por músicas e álbuns musicais.

Esta escolha foi feita baseada em vários fatores. Informações sobre músicas podem ser encontradas em abundância na Web. Existem sistemas, como o Last.fm [1] e MusicBrainz

[1] `http://last.fm`

(Swartz, 2002), que possuem uma base de dados com informações sobre músicas e fornecem estes dados através de uma API que pode ser acessada gratuitamente pela Web.

Música é um interesse comum à maioria das pessoas. Isto torna fácil conseguir usuários para testar o sistema. Testes de usabilidade remoto assíncronos não identificam um grande número de erros de usabilidade com um pequeno número de usuários, conforme explicado na Subseção 2.5. É necessário uma quantidade de usuários maior para alcançar eficácia similar ao teste realizado em laboratório.

Além do tema, é necessário determinar as restrições de recursos, pois diversas decisões de projeto têm por base as limitações de hardware consideradas. Foi tomado por base que o sistema executará inteiramente em uma máquina virtual do tipo *small* instância provida pela Amazon. Esta é uma categoria de máquinas com configuração de *hardware* modesta, provida pela Amazon. A escolha desta máquina se deve aos seguintes fatores:

- o serviço de *Virtual Private System* (VPS) fornecido pela Amazon é bastante difundido na industria. Isto significa que os resultados aferidos neste trabalho poderão facilmente ser reproduzidos por outros pesquisadores;
- se o sistema tiver bom funcionamento na configuração modesta, é garantido que também funcionará com igual ou maior qualidade em outras máquinas;
- a conexão à Internet fornecida a esta máquina é ideal para servidores e possibilitará tráfego de grande quantidade de dados, que é necessário na manipulação de arquivos multimídia;
- o ambiente oferecido é bastante confiável e tem um SLA (*Service Level Agreement*) de 99.95%, o que proporciona maior segurança do que hospedar o sistema na própria universidade.

A instância *small* da Amazon possui 1.6GB de memória, uma unidade de processamento EC2, sendo que cada unidade de processamento EC2 possui a capacidade de um processador Xeon 2007 de 1.0-1.2GHz, plataforma 32 bits e baixa performance de entrada/saída. O preço cobrado pela Amazon, pela instância descrita, na zona leste dos Estados Unidos é de $0.06US\$$, mais $0.10US\$$ por GB-mês de armazenamento e $0.12US\$$ por GB transerido para fora da Amazon (ama, 2013).

3.1 Arquitetura do Sistema Base

Foi necessário criar um sistema base composto por basicamente *crawler* e base de dado, conforme ilustrado no diagrama da Figura 3.1. No topo deste sistema base foram desenvolvidas as técnicas de exploração visual, RF e IR tratadas neste trabalho de mestrado.

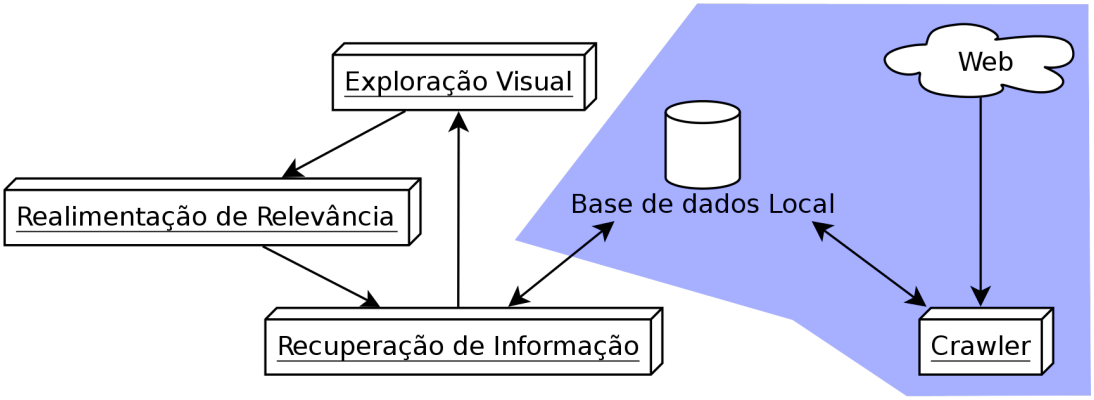

Figura 3.1: Interação entre os diferentes módulo, os módulos que não se alteram na variação do método de busca estão na área sombreada.

Dentre as vantagens de ter apenas um sistema de base de dados e *crawler*, está o fato de que os esforços serão minimizados, visto que estes módulos não terão que ser desenvolvidos mais de uma vez.

É importante salientar também que durante a fase de avaliação das técnicas abordadas, é possível agregar maior confiabilidade aos resultados, pois as únicas variáveis são as técnicas em avaliação.

No entanto, é necessário um cuidado maior na construção deste sistema, pois ele deve ser suficientemente modular, e esta característica é uma consequência de decisões relacionadas à arquitetura do sistema.

O sistema é uma aplicação Web, desenvolvido na linguagem de programação PHP. Esta linguagem é largamente usada e é relativamente fácil conseguir ajuda sobre esta linguagem em fóruns de discussão.

Toda a pilha de desenvolvimento utilizada neste sistema é composta por projetos de código aberto. Esta escolha não apenas minimiza custo com licenças de *software* como também possibilita que o sistema seja auditável em todo espectro de granularidade, do sistema operacional às chamadas aos métodos do *framework*.

Com o intuito de aumentar a agilidade no desenvolvimento do sistema com esta linguagem também é utilizado o *framework* Zend. Dentre os diversos *frameworks* existentes para PHP, este é o mais popular e bastante rico em recursos. Todo o código fonte também é livre e o principal time de desenvolvedores é composto por funcionários da fundação Zend.

O sistema base é acessado pela implementação das técnicas de exploração visual e RF através de uma API. Esta API provê funções através das quais é possível buscar por músicas. Existem dois tipos de busca. O primeiro retorna informações sobre as músicas, como título, álbum e grupo musical. O segundo tipo retorna informações específicas sobre

determinada música ou álbum, como a imagem que representa o objeto e a URL para o arquivo de áudio.

Além de proporcionar esta API aos módulos de visualização e ao RF, o sistema deve tratar a interação com o usuário que não está diretamente ligada à busca de música. Isto inclui:

- gerenciar o cadastramento e registro dos usuário do sistema;
- prover meios para o usuário tocar os arquivos de áudio e vídeo através do sistema;
- prover meios para o usuário baixar estes arquivos de áudio e vídeo para seu computador;
- organizar os arquivos dentro de listas;
- permitir que o usuário compartilhe músicas, tanto através de *link* direto como através de rede social.

Estas funcionalidades são independentes das técnicas utilizadas na interface de busca. Com estas funcionalidades o usuário é capaz de gerenciar os resultados de suas buscas.

3.2 Método de Busca Tradicional

O primeiro passo deste trabalho foi implementar um sistema de recuperação de informação bastante difundido, com o objetivo de que este possa ser utilizado como base de comparação para o conjunto de técnicas proposto nesta dissertação de mestrado.

Os seguintes pontos caracterizam um sistema clássico, ou tradicional, de IR, no que tange a interação com o usuário:

- a entrada de dados do usuário ocorre através de texto, comumente interpretadas como *keywords*;
- para cada busca do usuário o sistema retorna um conjunto de resultados;
- o conjunto de resultados retornado é exibido como uma lista com itens dispostos em ordem decrescente pela sua relevância à requisição.

No Amuzi, a implementação do sistema clássico de IR ocorre da seguinte forma. Primeiro o usuário descreve o alvo da busca, especificando o artista e o título do objeto (música ou álbum). O sistema retorna uma lista de possíveis resultados na forma de lista.

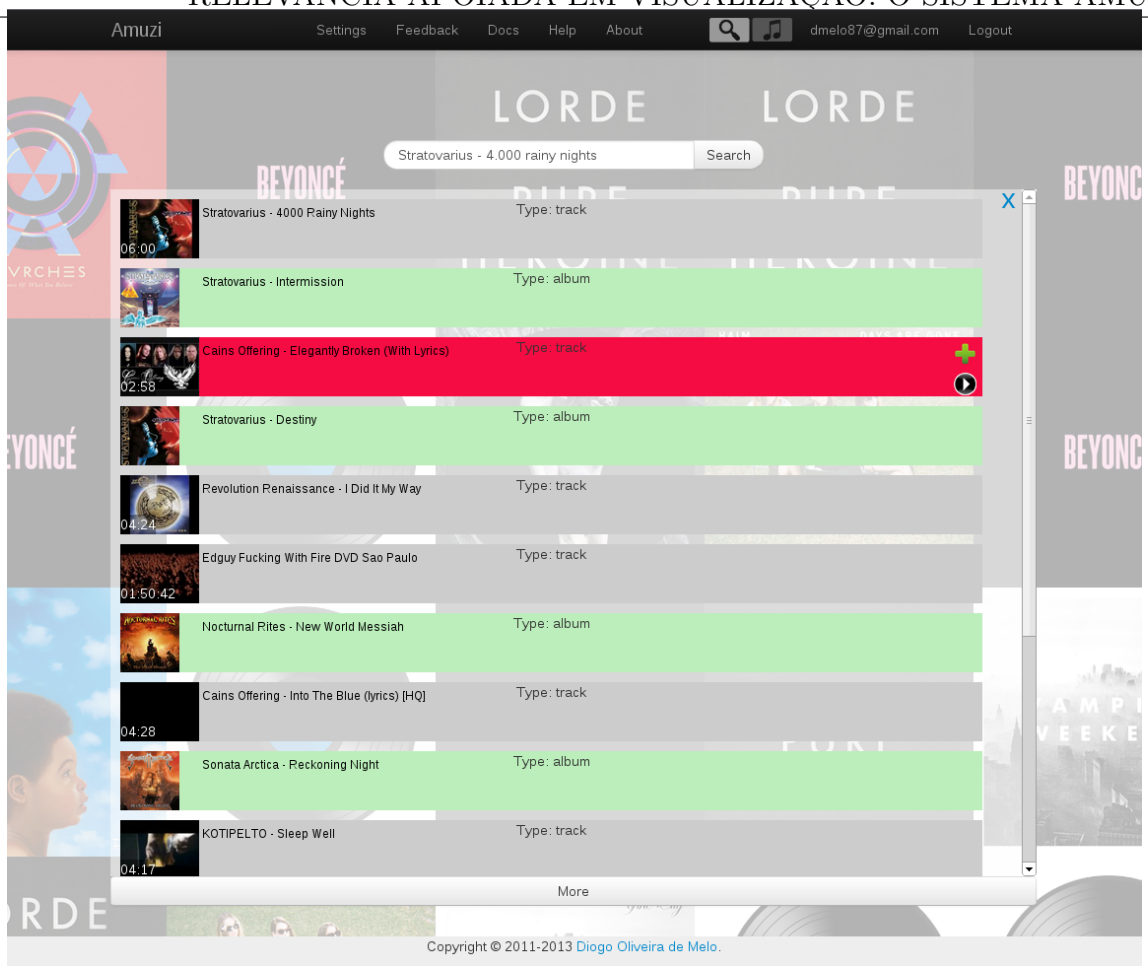

Figura 3.2: Captura de tela do sistema Amuzi com o sistema clássico de busca.

Cada item desta lista contém uma imagem que representa o resultado, uma breve descrição textual deste resultado e um conjunto de ícones que permite que o usuário execute ações com este resultado, conforme ilustrado na Figura 3.2.

É importante salientar que esta implementação não possui realimentação de relevância. Este modelo de busca é largamente conhecido. Sistemas como o providos pelo Google, Yahoo e Microsoft implementam variações. Isto é, dado um conjunto de palavras chave, o sistema retorna um conjunto de possíveis resultados apresentados na forma de lista.

3.3 Recuperação de Informação utilizando Realimentação de Relevância apoiado em Visualização

A técnica de visualização escolhida foi o IncBoard, por possuir as seguintes características:

- agrupar os elementos por similaridade;

- admitir a inclusão de elementos em um mapa já montado, minimizando as mudanças no mapa;
- representar cada elemento por um ícone;
- não sobrepor elementos;
- permitir ser estendido para aumentar interatividade com usuário.

Este conjunto de características é tal que permite fácil integração com realimentação de relevância, pois o usuário pode apontar elementos interessantes ou não interessantes à busca que está fazendo.

Além disso, na recuperação de informação, novos elementos serão retornados e estes poderão ser adicionados ao mapa corrente sem grandes alterações no mapa atual.

Pinho et al. (2009) não descreve como deve ser a interação com o usuário entretanto, o IncBoard é bastante flexível à adição de novas funcionalidades. Uma das alterações feitas no IncBoard implementado no Amuzi foi a inserção de um caixa de texto, que é exibida apenas quando o usuário posiciona o cursor para uma das células do IncBoard. Esta caixa de texto apresenta informações sobre o elemento contido na célula. A caixa desaparece quando o cursor sai da região da célula.

Outra adaptação feita no IncBoard foi controlar a cor da borda de cada célula de acordo com o artista referente à música. A cor da borda passa a alternar entre azul e vermelho na célula que está sob o cursor e em todas as células do mesmo artista. A caixa de informação, descrita no parágrafo anterior, e o controle da cor das bordas das células estão ilustrados na Figura 3.3.

3.4 Funcionalidade de Autocompletar

Em muitos casos o usuário pode não se lembrar do nome exato do artista, álbum ou música que está buscando. É desejável mostrar sugestões de busca a medida que o usuário digita a busca. Utilizar a funcionalidade de autocompletar não deve ser mandatório para realizar a busca, entretanto expor nomes completos de artista, música/álbum e imagem tendem a ajudar o usuário a identificar o objeto que está procurando.

A lista de resultados sugeridos deve ser atualizada a cada letra digitada pelo usuário. Desta funcionalidade é possível extrair duas informações importantes. Primeiro, a busca por resultados na base de dados considera que o termo a ser buscado provavelmente está incompleto, dado que o usuário ainda não terminou de digitar. Segundo, o tempo de resposta da busca deve ser muito curto, uma vez que o tempo entre digitar uma letra e outra também é muito baixo.

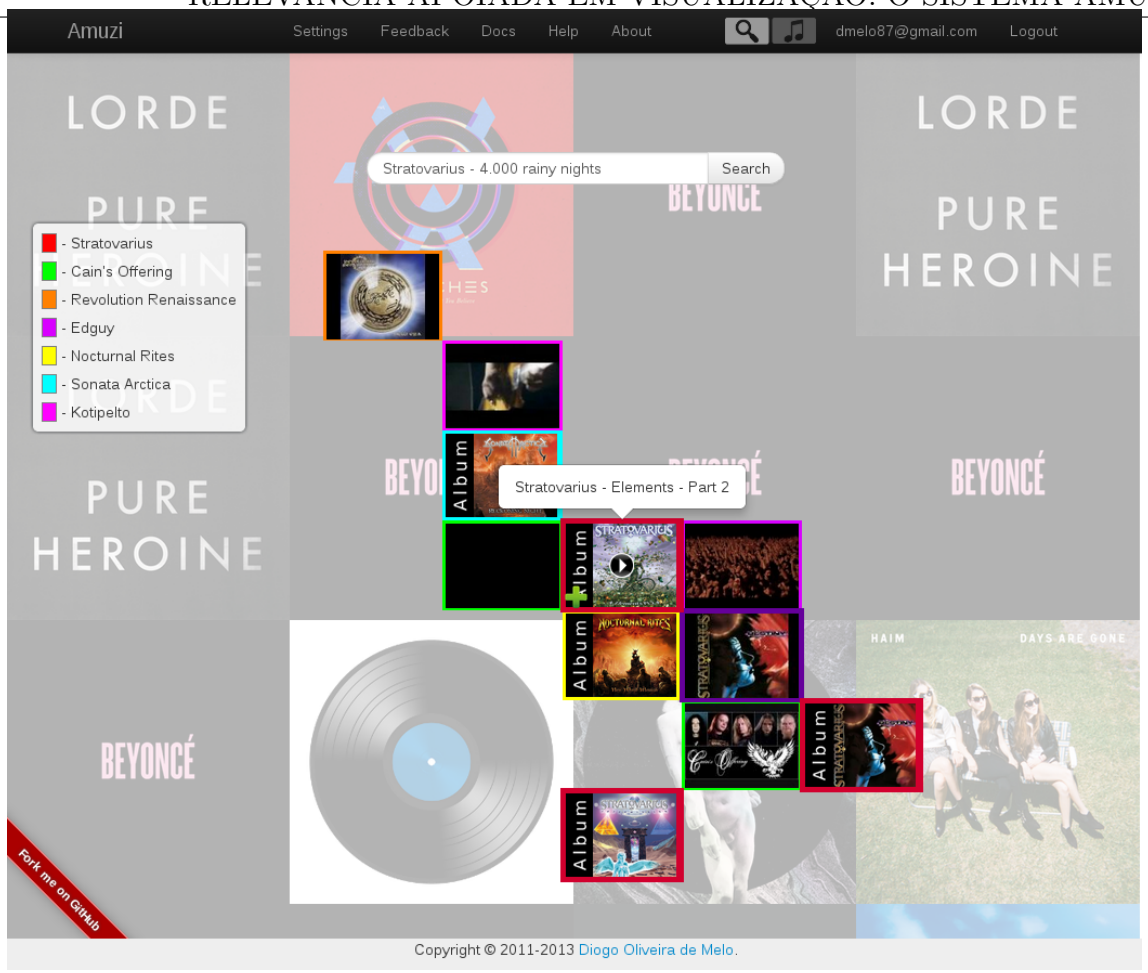

Figura 3.3: Captura de tela do sistema Amuzi com o IncBoard exibindo a caixa de texto com informações sobre uma das células e as diferentes cores que as bordas das células assumem de acordo com o artista do elemento de cada célula.

Nielsen (1993) apresenta limites de tempo para mostrar resultados ao usuário. Se as sugestões forem apresentadas em até $100ms$, o usuário terá a percepção de que a resposta é instantânea. Se houver até um segundo de atraso, o usuário interpreta que o servidor está processando a requisição. Acima de um segundo, a mente do usuário começa a pensar em outros assuntos. Se a resposta demorar mais de 10 segundos, ele desistirá de esperar.

Considerando estas barreiras, é importante minimizar o tempo necessário para retornar as sugestões. O ideal é que o tempo seja inferior à $100ms$. Entretanto, ainda que o servidor consiga calcular o resultado em tempo abaixo de $100ms$, há de se considerar a latência de rede.

O servidor e o cliente podem estar em quaisquer lugares do globo. Na pior das hipóteses, do ponto de vista de latência de rede, estarão em pontos opostos. Considerando que o raio da terra é de aproximadamente $6.4 \times 10^6 m$ (National Imagery and Mapping Agency, 2000), a velocidade da luz é de aproximadamente $2.99 \times 10^8 m/s$ (Young et al., 2011) e a velocidade da transmissão de sinais está limitada pela velocidade da luz, temos que a

latência para transmitir dados entre pontos extremos do globo tem um limite inferior de $67ms$.

Considerando que as redes reais tem performance muito inferior à rede hipotética exemplificada no parágrafo anterior, usuários de continente diferente do servidor que hospeda o Amuzi serão afetados negativamente pela latência da rede, na funcionalidade de autocompletar.

Uma possível solução para o problema da latência é utilizar uma rede de distribuição de conteúdo dinâmico (*Content Distribution Network* – CDN) (Peng, 2004), que consiste em diversos servidores espalhados em diferentes locais do globo. De acordo com a localização do usuário, o servidor de nomes de domínios (*Domain Names Server* – DNS) redirecionaria a requisição para o servidor mais próximo minimizando o tempo de resposta.

Supondo que a aplicação está sendo executada em uma estrutura baseada em CDN, e que a latência de rede efetiva é abaixo de $80ms$, o servidor deve calcular os resultados em um tempo inferior a $20ms$ para garantir que a maioria das requisições possam ser respondidas para o usuário em até $100ms$.

Três abordagens foram estudadas para resolver o problema de busca de recomendações para o mecanismo de autocompletar:

- utilizar mecanismos de busca já contidos na base de dados;
- utilizar *cache* e API do Last.fm;
- o algoritmo *prefix tree* e similares.

As três subseções seguintes descrevem os resultados obtidos em cada um dos métodos abordados.

3.4.1 Busca Direta na Base de Dados

Para buscar diretamente na base de dados, é necessário inserir todas as informações de músicas, álbuns e artistas, bem como suas conexões na base de dados. Estas informações foram extraídas do MusicBrainz (Swartz, 2002).

O MusicBrainz é uma base de dados de músicas com licença *open source*. Dentre outras informações, contém uma lista extensa de artistas, músicas e álbuns já lançados. Esta base de dados é alimentada por voluntários e está disponível gratuitamente para *download*.

Após carregar a base de dados localmente, foi necessário transformar os dados do formato que está no MusicBrainz para o esquema da base de dados do Amuzi. No total, foram importados, 5.873.814 músicas, 932.930 álbuns e 620.900 artistas.

Antes de tentar implementar a funcionalidade de autocompletar utilizando a base de dados diretamente, foi realizado um teste para aferir a plausibilidade desta estratégia. A dúvida que se desejou sanar com este teste é o tempo necessário para responder à uma requisição de autocompletar, isto é, dado uma *string*, quanto tempo é necessário para retornar o conjunto de músicas e álbuns que possuem alguma correspondência com o texto provido pelo usuário.

Foram escolhidos os seguintes termos: "Col", "Coldplay", "The Rolling Stones", "Ddd", "Dikahek", "Kie jkalkje njdjeye". Estes textos foram escolhidos de forma tal a testar os cenários em que existirá resposta e cenários em que a entrada fornecida pelo usuário é desconhecida pela base de dados. Também verifica entradas curtas, médias e longas, de 3, 7 e 18 caracteres, respectivamente.

Para simplificar, foi testado apenas se as *strings* são reconhecidas como artistas, álbuns ou músicas. Considerando que as buscas utilizadas para a funcionalidade de autocompletar teriam que utilizar a operação *JOIN*, que tende a demandar mais tempo, se esta estratégia falhar neste teste, também falharia em um teste que considera o relacionamentos entre artista, álbum e música.

Tabela 3.1: Tempo para consultar elementos utilizando SQL.

Tabela	Artista	Álbum	Música	Total
Col	0.08 s	0.03 s	0.02 s	0.13 s
Coldplay	0.50 s	1.64 s	2.31 s	4.44 s
The Rolling Stones	0.26 s	0.62 s	0.73 s	1.61 s
Gkb	0.73 s	1.52 s	8.56 s	10.81 s
Dikahek	0.87 s	1.98 s	9.07 s	11.92 s
Kie jkalkje njdjeye	0.88 s	65.87 s	61.87 s	128.62 s

Os resultados apresentados na Tabela 3.1 foram obtidos através da execução do comando SQL no seguinte formato: *SELECT * from TABLE where name like '%STRING%' limit 5;*. Substituindo *TABLE* por cada uma das três tabelas consultadas e *STRING* por cada um das 8 *strings* utilizadas. Os resultados foram limitados em 5, pois na apresentação dos possíveis resultados, para o usuário, no máximo 10 resultados são apresentados. Sendo até 5 músicas e até 5 álbuns.

Uma segunda execução destes comandos SQL foi feita. Entretanto, desta vez, ao invés de limitar o número de tuplas, foi feita uma contagem de resultados obtidos. O formato do SQL utilizado foi: *select count(*) from TABLE where name like '%STRING%';*. Os tempos obtidos nesta segunda execução estão expostos na Tabela 3.2.

Com base nos resultados obtidos nestes testes é possível extrair as seguintes conclusões:

Tabela 3.2: Tempo para consultar quantidade total de elementos que casam com padrão.

Tabela	Artista	Álbum	Música	Total
Col	1.22 s	3.83 s	27.70 s	32.75 s
Coldplay	0.79 s	1.85 s	9.81 s	12.45 s
The Rolling Stones	0.95 s	1.99 s	11.07 s	14.01 s
Gkb	0.82 s	1.71 s	8.38 s	10.91 s
Dikahek	0.78 s	1.80 s	69.36 s	71.94 s
Kie jkalkje njdjeye	27.08 s	37.15 s	24.33 s	88.56 s

- quanto maior a quantidade de vezes que uma *substring* ocorre dentro da base de dados, menor será o tempo necessário para encontrar 5 tuplas que satisfaçam o padrão;

- quanto maior o tamanho do padrão procurado, maior o tempo necessário para buscá-lo.

Durante os experimentos foi observado que, quando uma consulta é realizada, todo processamento disponível na máquina ou toda a sua capacidade de entrada/saída é consumida. Isto implica que o tempo de resposta a requisição aumenta caso outra requisição esteja em andamento, pois as requisições vão concorrer pelos recursos. Além disso, outros processos podem ter dificuldade em operar durante a resolução de uma requisição de autocompletar.

Uma possibilidade é manter toda a base de dados em memória primária utilizando *Random Access Memory File System* (RAMFS), por exemplo (Snyder, 1990). Porém, como a base de dados possui 1.2GB, esta possibilidade se torna inviável, pois o restante do sistema necessita de mais de 400MB de memória RAM para operar.

É necessário ressaltar que estes testes foram feitos utilizando a versão 5.5.34 do MySQL. O operador *LIKE* não consegue ter ganho de performance com a criação de índices, a não ser que o início da *string* a ser buscada seja constante. Situação similar acontece com os SGBDs PostgreSQL 9.2.5 e MongoDB 2.4.8 .

Há uma outra estrutura de dados que possui performance muito melhor, a qual utiliza o operador SQL *MATCH..AGAINST*. No MySQL, é necessário utilizar o índice *full-text*. Entretanto, este operador foi projetado para tratar a base o texto como um conjunto de palavras chave, ignorando a ordem de ocorrência dos termos. Esta característica se deve ao fato do operador utilizar o algoritmo de *Ranking* com Espaço de Vetores (mys, 2013)

Para o caso específico de música, a ordem das palavras tem grande influência sobre os resultados. Sendo assim, o uso deste operador foi descartado.

Com base no que foi aferido é possível afirmar que uma implementação utilizando busca direta na base de dados é pouco eficiente, em termos de responder a requisição em tempo hábil, e consome muitos recursos de processamento de entrada/saída.

3.4.2 Utilização de cache e API do Last.fm

Conforme visto na Subseção 3.4.1, manter todos os valores na base de dados é inviável. Uma alternativa é utilizar *cache*. Ao chegar uma requisição, o sistema verifica se a base de dados possui informações o suficiente para atender a requisição. Se possuir, retorna os resultados encontrados localmente. Caso contrário, consulta o sistema Last.fm.

Esta abordagem parte da hipótese de que a maioria das requisições irão referenciar apenas uma minoria dos possíveis resultados. A base de dados inicia com nenhuma informação sobre artistas, álbuns e músicas e o sistema cresce de acordo com a demanda dos usuários.

Com sorte, manter apenas um subconjunto dos dados é o suficiente para manter uma alta taxa de acerto. A busca sobre uma pequena parte dos dados será rápida o suficiente para manter o tempo de resposta das requisições abaixo de $100ms$.

Entretanto, esta abordagem possui vários pontos negativos. Fazer uma requisição ao Last.fm consome entre $10s$ e $20s$. Os primeiros usuários a utilizarem o sistema seriam muito prejudicados. Além disso, caso o usuário requisite algo que não exista, o sistema irá pesquisar no Last.fm em vão e o usuário esperará um longo tempo para receber a resposta de que não há resultados compatíveis com a busca.

Esta abordagem possui muitos riscos, principalmente por depender do empenho dos usuários em utilizarem um sistema que será lento, até que a *cache* seja populada e aumente os acertos.

3.4.3 Utilização do Algoritmo Suffix Tree e Similares

O problema imposto pela funcionalidade de autocompletar pode ser modelado como o de encontrar *substring*, ou o problema de encontrar a *substring* mais próxima, se for considerado corrigir erros de digitação do usuário.

O algoritmo Árvore de Sufixos (*Suffix Tree*) (Weiner, 1973) possui complexidade $O(m)$ com m sendo o tamanho da *substring*. O algoritmo consiste em pré processar o texto de forma a organizá-lo em uma estrutura de árvore. Nesta árvore, cada caminho da raiz à folha representa uma possível *substring*. Desta forma, para buscar uma *substring*, basta fazer uma caminhada na árvore. Caso a caminhada termine em um nó que não seja folha, significa que há um conjunto de possíveis resultados. Se a caminhada acabar em

uma folha, o algoritmo encontrou um casamento. Se não for possível caminhar usando a *substring*, não há casamento.

Entretanto, ainda em implementações cuidadosas do Suffix tree, o espaço ocupado em memória principal vai de 10 a 20 vezes o tamanho do texto. Considerando que a lista de músicas contém 284MB, a árvore precisaria de algo entre 2840MB e 5680MB para ser construída. O custo de memória inviabiliza sua utilização.

Outro algoritmo averiguado foi o *Array* de Sufixos (*Suffix Array*), proposto por Manber e Myers (1990). Este algoritmo é bastante semelhante à Árvore de Sufixos, porém a estrutura utilizada é a de um *array*. Neste *array* são guardadas as posições das *substrings*, que estão ordenadas em ordem alfabética. Para buscar uma *substring*, é necessário fazer uma busca binária no *array* de sufixos. Logo, a ordem de complexidade deste algoritmo é $O(log(n))$, sendo n o tamanho do texto. Outra vantagem deste algoritmo é a possibilidade de fazer a busca pelo padrão mais próximo e não apenas pelo casamento perfeito da *substring*.

Como cada posição é armazenada apenas uma vez, é necessário 4 bytes por caractere para armazenar o *array* de sufixos. Se o sistema utilizado fosse de 64 bits, então seriam necessários 8 bytes por caractere. Assumindo que cada caractere ocupa 1 byte, a memória necessária para armazenar o *array* é 4 vezes o tamanho do texto, em sistemas 32 bits, e 8 vezes o tamanho do texto, em sistemas 64 bits. Isto implica na necessidade de utilizar 1136MB em memória principal para armazenar o *array* de sufixos, o que também é proibitivo.

O algoritmo Boyer-Moore-Horspool (BMH) (Horspool, 1980) possui complexidade $O(n)$ e encontra *substrings* em um texto. Apesar da complexidade do algoritmo ser linear em relação ao tamanho do texto, quanto maior o tamanho do padrão menor tende a ser o tempo de execução. Dado que não houve casamento em determinada posição do texto, o algoritmo avança até M caracteres, sendo M o tamanho do padrão. Ao contrário do algoritmo de força bruta, que sempre avança apenas um caractere.

Tabela 3.3: Tempo necessário para fazer busca utilizando algoritmo de Boyer-Moore-Horspool.

Tabela	Artista - Álbum	Artista - Música	Total
Col	$51ms$	$53ms$	$104ms$
Coldplay	$86ms$	$487ms$	$573ms$
The Rolling Stones	$29ms$	$459ms$	$488ms$
Gkb	$823ms$	$1031ms$	$1854ms$
Dikahek	$88ms$	$436ms$	$524ms$
Kie jkalkje njdjeye	$17ms$	$244ms$	$261ms$

Em relação à memória, o algoritmo precisa alocar apenas o suficiente para carregar o texto a ser buscado. Ou seja, o algoritmo alocará 284MB em memória principal para buscar por títulos de música, o que é uma quantidade aceitável. A Tabela 3.3 apresenta os tempos de execução para busca de alguns padrões.

Apesar dos tempos obtidos utilizando BMH serem bem menores do que os resultados anteriores, ainda é insuficiente. Na aplicação de autocompletar, uma nova busca é necessária a medida que o usuário continua digitando as letras da busca. Considerando que uma busca pode ocupar até dois segundos do tempo do processador, as buscas de um mesmo usuário podem ter sua execução sobreposta, levando a maiores tempos de resposta. Mais de um usuário utilizando o sistema de busca pioraria ainda mais a situação.

A abordagem vista com o Array de Sufixos tem um custo de memória proibitivo, entretanto há abordagens que utilizam armazenamento em disco para implementação desta técnica.

A implementação do Suffix array é dividida em duas partes. A primeira consiste em pré-processar o texto de forma a montar a estrutura de dados necessária para os acessos de busca. A segunda parte é a busca em si.

A estrutura necessária para o *array* de sufixos consiste na lista dos índices ordenadas e divididas em arquivos de tamanhos iguais, *chunks*, com o primeiro item de cada arquivo armazenado em memória principal.

Os *chunks* podem ser obtidos utilizando um algoritmo de ordenação externa. foi implementado um algoritmo de *merge* de X vias. Primeiro, os inteiros de 0 à $n-1$ são disposto em X arquivos, tal que n é o número de caracteres do texto. Inicialmente, os primeiros 4MB de cada arquivo são carreados em memória. A cada iteração, é executado o *merge* de X vias para determinar o índice da menor *substring*, este índice é retirado das estruturas e gravado no *chunk*.

Quando um *chunk* fica completo, o arquivo é gravado em disco e o próximo começa a ser gravado. Estas iterações continuam até que não haja mais índice a ser carregado de nenhum dos X arquivos nem haja nenhum índice em memória. Ao final das iterações, os *chunks* estarão formados e prontos para serem utilizados.

Pré-processar a lista de títulos de músicas com 284MB, consome aproximadamente 3 horas de processamento, na instância *small* da Amazon. O algoritmo de ordenação externa utilizado é descrito em Knuth (1998). A ordem de complexidade é $O(log(n) \times n)$.

Uma vez que os *chunks* estão montados, é possível realizar as buscas. Utilizando *chunks* de 8192 registros, precisaremos manter apenas 36,352 registros em memória (o primeiro de cada *chunk*). Em uma máquina de 32 bits, *array* de sufixos em memória irá ocupar 142KB.

Busca binária é utilizada para encontrar o índice mais próximo do texto procurado. Uma vez que este índice foi encontrado, será necessário alocar mais 24KB de memória para carregar o *chunk* e realizar a mesma busca binária dentro do *chunk*.

Foram testadas as mesmas entradas de texto utilizadas nos testes anteriores. Em todos os casos, o sistema respondeu em aproximadamente $50ms$ para a primeira e o tempo cai para $0.5ms$, da segunda requisição em diante, considerando a mesma entrada. Este comportamento se explica pela funcionalidade de paginação do *kernel*. Uma vez que o arquivo está carregado em memória, o tempo de processamento é muito baixo.

3.5 Cálculo de Similaridade

Considere que serão inseridos N elementos em um tabuleiro vazio, utilizando o IncBoard clássico. Para inserir o i-ésimo elemento, o IncBoard precisa da similaridades entre o i-ésimo elemento e cada um dos elementos já inseridos no tabuleiro. Logo, para inserir N elementos em um tabuleiro vazio, é necessário calcular a similaridade entre todos os pares de elementos.

Entretanto, o sistema Last.fm provê similaridade entre músicas com uma semântica probabilística, não necessariamente relacionada ao conteúdo das músicas.

Dizer que duas músicas são similares se relaciona à propriedades intrínsecas ao dois objetos: gênero musical, ritmo, timbre, idioma, etc. Por outro lado, o conceito de similaridade utilizado no Last.fm está relacionado à probabilidade de dois usuários gostarem da mesma música.

No Last.fm a similaridade da música A para a música B é proporcional a probabilidade de um usuário gostar da música B dado que gostou da música A. O IncBoard espera que a similaridade $S(A, B) = S(B, A)$, dado que a similaridade estará relacionada à distância entre os elementos em um espaço N dimensional, e distância é uma propriedade simétrica. Entretanto, a medida de similaridade provida pelo Last.fm pode ser interpretada como uma probabilidade condicional $Similaridade(A, B) = P(A|B)$. Logo, similaridade de A para B não necessariamente é igual a similaridade de B para A.

Com o objetivo de utilizar a similaridade provida pelo Last.fm no IncBoard, foi feita a simplificação denotada pela Equação 3.1. Esta simplificação implica em duas vantagens: torna o conceito de similaridade utilizado simétrico e estima $P(A|B)$ dado $P(B|A)$ diminuindo a quantidade de consultas necessárias ao Last.fm.

$$S(A,B) = S(B,A) = \begin{cases} \frac{P(A|B)+P(B|A)}{2} & \text{Se P(A \mid B) e P(B \mid A) são conhecidos} \\ P(A|B) & \text{Se apenas P(A \mid B) é conhecido} \\ P(B|A) & \text{Se apenas P(B \mid A) é conhecido} \\ 0 & \text{Se nem P(A \mid B) nem P(B \mid A) forem conhecidos} \end{cases} \tag{3.1}$$

Definido o modo como a similaridade provida pela Last.fm será tratada, é necessário definir qual algoritmo será utilizado para requisitar as medidas de similaridade ao Last.fm. A relevância deste tópico se deve ao fato de que, como mencionado anteriormente, requisições ao Last.fm são custosas em termos de tempo.

Considerando que a quantidade de músicas catalogadas pelo projeto MusicBrainz é da ordem de milhões, o conjunto de todas as similaridades entre cada par de música é da ordem de 10^{12}. Não é tecnicamente viável armazenar esta quantidade de informações, ainda menos plausível calcular todas à priori. Por outro lado, requisitar estas similaridades sob demanda, à medida que chega uma requisição do usuário, degrada a experiência do usuário.

O alvo da busca, que é especificada pelo usuário, pode ser exibido imediatamente, sem a necessidade de informações de similaridade. Isto significa que o alvo da busca pode ser exibido em menos de um segundo e os demais elementos podem ser exibidos em seguida. Com o objetivo de diminuir o tempo médio, do intervalo entre exibir o alvo da busca e os elementos similares, é possível utilizar *cache*.

Sempre que uma informação de similaridade é requisitada ao Last.fm, esta é mantida na base de dados. É esperado que apenas uma pequena porcentagem do conjunto de similaridades precise ser mantido localmente e ainda assim ter uma alta taxa de acerto de *cache*. Apesar disso, uma política de renovação de *cache* deve ser empregada para 1) manter a informação atualizada e 2) impedir que a base de dados local tenha informação em excesso, o que pode provocar lentidão no sistema.

Apesar da utilização de *cache*, é esperado que falte uma quantidade significativa de estimativas de similaridades, para responder a cada requisição. Sendo assim, a extrapolação pode ser utilizada. Dado que são conhecidos os valores de similaridade entre A e B e entre B e C, é possível estimar o valor da similaridade entre A e C.

Modelando as similaridades providas pelo Last.fm como probabilidades condicionais, é possível utilizar o sistema matemático Cadeias de Markov (Kemeny e Snell, 1969). Uma matriz de similaridade é tal que, o elemento M_{ij} representa a probabilidade do usuário gostar da música j, dado que ele gosta da música i.

A técnica IncBoard requer pares de similaridade entre os elementos exibidos. Quanto mais precisas as estimativas de similaridades entre objetos, maior a qualidade da visuali-

zação gerada pelo IncBoard. Por outro lado, estimativas pobres ou falta de informação na matriz de similaridade, alta esparsidade, pode causar deficiências na visualização gerada.

Em Cadeias de Markov, a matriz de probabilidade deve ser tal que, um elemento M_{ij} representa a probabilidade do sistema assumir o estado j dado que se encontra no estado i. Logo, é possível utilizar Cadeias de Markov na matriz de similaridade desde que esta passe por uma normalização, fazendo com que a soma dos elementos de uma linha seja sempre igual a 1.

Após normalizar a matriz de similaridade, obtendo M', é possível multiplicar a matriz por ela mesma. Neste caso, M'^2_{ij} representa a probabilidade do sistema ir do estado i ao estado j em dois passos. No conceito de similaridade de músicas, isto representa uma probabilidade aproximada do usuário gostar da música j, dado que gosta da música i.

$$M''_{ij} = \begin{cases} M'_{ij} & \forall M'_{ij} \neq 0 \\ M'^2_{ij} & \forall M'_{ij} = 0 \end{cases} \tag{3.2}$$

Através de M' e M'^2, uma terceira matriz é produzida. Esta matriz é calculada de acordo com a Equação 3.2. Desta forma, a matriz final representa a matriz normalizada agregada dos valores extrapolados.

Utilizar este artifício diminui a esparsidade da matriz de similaridade e, possivelmente, aumenta a qualidade da representação produzida pelo IncBoard. Em relação a este procedimento, duas questões podem ser levantadas:

- Qual a melhoria na diminuição da esparsidade da matriz resultante deste processo?
- Qual a diferença percebida pelo usuário?

Para verificar o ganho de informação ao adotar o procedimento descrito nesta seção, o seguinte experimento foi realizado. Diversas matrizes de ordem 100 foram geradas aleatoriamente. A extrapolação foi aplicada nesta matriz e foram comparadas as esparsidades da matriz original e da matriz resultante.

A esparsidade de uma matriz é dada pela razão entre a quantidade de elementos nulos e a quantidade total de elementos, considerando que a variação de esparsidade depende da esparsidade da matriz original e da posição dos elementos não nulos. Para cada valor de esparsidade variando de 0.99 a 0.01, em decrementos de 0.01, foram geradas 100 matrizes aleatórias.

Para gerar uma matriz M' com esparsidade $0.01 \leq S \leq 0.99$, o seguinte procedimento foi adotado. Para cada célula da matriz, foi escolhido um número inteiro pseudo aleatório $0 \leq r < 10^6$. Se $r \leq S \times 10^6$, então o valor da célula é 0, caso contrário, o valor da célula é 1.

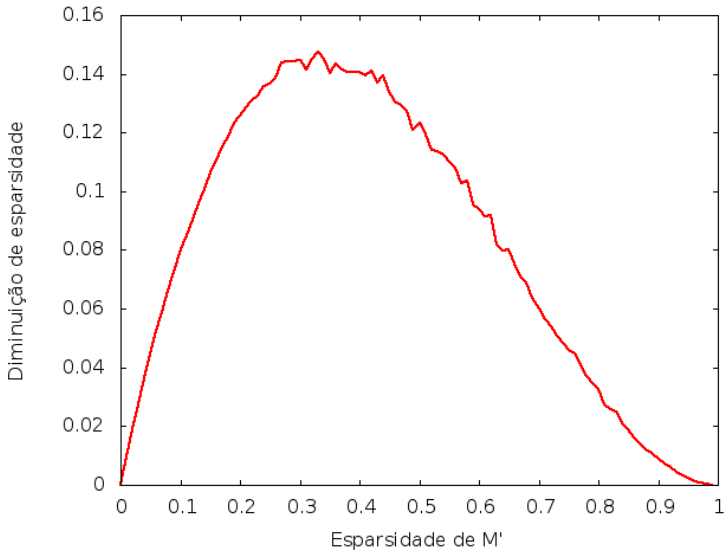

Figura 3.4: Diminuição de esparsidade por espacidade de M' para matrizes de ordem 100 geradas aleatoriamente.

A partir de M', M'' é calculada, conforme determinado pela Equação 3.2. É calculado então a diferença de esparsidade entre as duas matrizes M' e M''. A Figura 3.4 contém um gráfico que mostra a média de diminuição de esparsidade, dado a esparsidade de M'.

Para casos em que a esparsidade é maior que 0.9, temos que o ganho de informação com a extrapolação é ínfimo, menos de 1% dos elementos deixam de ser zero para assumir um valor estimado.

Este método produz um grande ganho de informação quando a esparsidade da matriz está entre 0.2 e 0.5. Para estes casos, a esparsidade diminui entre 0.12 e 0.15.

3.6 Adaptação do IncBoard para o Amuzi

O IncBoard, proposto por Pinho et al. (2009), tem a Equação 2.1 como peça central para determinar a posição de novos elementos no tabuleiro. À princípio, para resolver o estado de instabilidade, é necessário considerar 16 possibilidades: mover o novo elemento para uma das oito células vizinhas ou mover o elemento que já ocupava a célula para uma das oito células vizinhas.

A Equação 2.1 é calculada 16 vezes, sempre considerando o conjunto de todos os elementos presentes no tabuleiro L. A opção com o menor erro W_{err} é a escolhida. É necessário calcular W_{err} para cada uma das possíveis configurações pois, dependendo da posição dos elementos, o *ranking* R_{ni} será diferente.

$$W_{err}(E_i, L) = \sum_{E_j \in L_9} |R_{ci}(E_j) - R_{ni}(E_j)| \times (|L| - R_{ni}(E_j)) \tag{3.3}$$

Na Equação 2.1, pertencente ao IncBoard clássico, o novo a posição do novo elemento considera todos os demais elementos já inseridos no tabuleiro. A Equação 3.3, proposta neste trabalho de mestrado, considera apenas o elemento mais similar ao novo elemento em conjunto com seus vizinhos (L_9 representa o conjunto de elementos contidos na célula instável mais suas oito vizinhas), proporcionando uma grande diminuição no custo de computação da possibilidade com menor erro. Considerando o ponto de vista de implementação do algoritmo IncBoard, utilizar a Equação 3.3 ao invés da Equação original 2.1 implica em iterar uma quantidade constante de vezes no somatório, ao invés de iterar N vezes, para N igual ao número de elementos no tabuleiro.

A Equação 2.1 atribui um peso, $|L| - R_{ni}(E_j)$, a cada diferença entre os *rankings* que é encontrada. Este termo faz com que elementos com maior posição no *ranking* R_{ni} tenham maior impacto no valor de W_{err}. Observe que quanto maior L, menor o impacto do termo $|L| - R_{ni}(E_j)$ para os elementos em L_9.

$$W_{err}(E_i, L) = \sum_{E_j \in L_9} |R_{ci}(E_j) - R_{ni}(E_j)| \tag{3.4}$$

Baseado nestas obervações, uma nova fórmula é proposta para calcular o valor de W_{err}, dada pela Equação 3.4. Nesta equação, o peso da diferença de *rankings* para determinado elemento E_j é dado apenas pela diferença de posição que este elemento ocupa nos *rankings* e não é ponderado pela sua distância ao topo do *ranking* R_{ni}.

Dois experimentos são descritos nesta seção. Ambos requerem uma medida de qualidade para avaliar a visualização gerada por variações do IncBoard. A Subseção 3.6.1 contém a descrição desta medida de qualidade. As Subseções 3.6.2 e 3.6.3 descrevem a realização dos experimentos que foram realizados com o objetivo de testar a eficácia de utilizar a Equação 3.4 e a abordagem descrita por Pinho et al. (2009) para melhorar o tempo de execução do IncBoard.

3.6.1 Medida de Qualidade

$$Q = \frac{\sum_{E_i \in L} \sum_{j \in 1..8} |E_i - V(E_i, j)|}{|L|} \tag{3.5}$$

Dado que o objetivo do IncBoard é fazer com que elementos similares sejam posicionados próximos um ao outro, podemos analisar a medida dada pela Equação 3.5 para medir o quão próximo um tabuleiro está de um ponto ideal.

Na Equação 3.5, $V(E_i, j)$ representa o j-ésimo vizinho de E_i, ou o próprio E_i, caso o vizinho não exista. Q consiste na média da somatória das diferenças entre um elemento e seus vizinhos. Quanto menor Q, melhor a visualização gerada.

Outra medida de qualidade que se enquadra a estes testes é a *Neighboring Hit* (Paulovich et al., 2008). Nesta medida, para cada uma dos elementos da visualização, é medido a proporção de vizinhos que pertence à mesma classe que o elemento corrente. Quanto maior o valor desta medida, melhor.

Para os testes das subseções seguintes, o IncBoard é preenchido com elementos aleatoriamente gerados. Cada elemento será associado a um valor escalar inteiro entre 0 e 255. A dissimilaridade entre dois elementos E_i e E_j pode ser obtida pela expressão $|E_i - E_j|$.

Por melhor que seja o algoritmo utilizado para inserir um conjunto de elementos no tabuleiro, Q nunca chegará a zero. O valor ótimo de Q é oito. Este valor indica que, para cada elemento, todos os seus vizinhos tem diferença de apenas uma unidade. Esta, é claro, é uma situação utópica.

Outro ponto a considerar é que esta medida de qualidade Q, não contempla a fragmentação. Caso a disposição dos elementos nas células seja fragmentada, o valor de Q poderá ser baixo e ainda assim poderíamos considerar a visualização ruim.

Analisando o outro extremo, valor ruim de Q será obtido quando substituirmos a equação que determina W_{err} por uma equação que retorna valores aleatórios. Desta forma, cada uma das 16 possibilidades teria igual chance de ser escolhida e a visualização gerada pode ser considerada aleatória.

Considerado apenas elementos cujos valores são inteiros e variam entre 0 e 255, 300 elementos aleatoriamente gerados foram inseridos em um tabuleiro vazio. Após a inserção dos 300 elementos, a medida Q foi aferida. Este procedimento foi repetido 1.000 vezes e a Equação que determina W_{err} utilizada foi uma função que retorna valores aleatórios entre $0 e 1.000.000$, ou seja, qualquer uma das 16 possibilidades tem igual chance de ser escolhida.

O soma de variáveis aleatórias de mesma distribuição probabilística tende a se aproximar de uma distribuição gaussiana (Grinstead e Snell, 1997). Neste experimento, os resultados aferidos tiveram uma média de 504 e a variância foi de 2484. Podemos concluir que, se substituirmos a Equação 2.1 por uma que gere tabuleiros com qualidade Q média de 504, então a nova equação é inútil.

3.6.2 Teste de Performance da Abordagem Proposta

Para comparar as duas Equações 2.1 e 3.4 foi realizado um experimento semelhante ao descrito na subseção anterior. 300 elementos foram gerados aleatoriamente e inseridos no IncBoard. Este procedimento foi realizado 1.000 vezes para cada uma das equações.

Os elementos considerados são imagens em tons de cinza. O espaço de cores entre branco e preto foi dividido em 256 tons. A dissimilaridade entre dois elementos dada pela

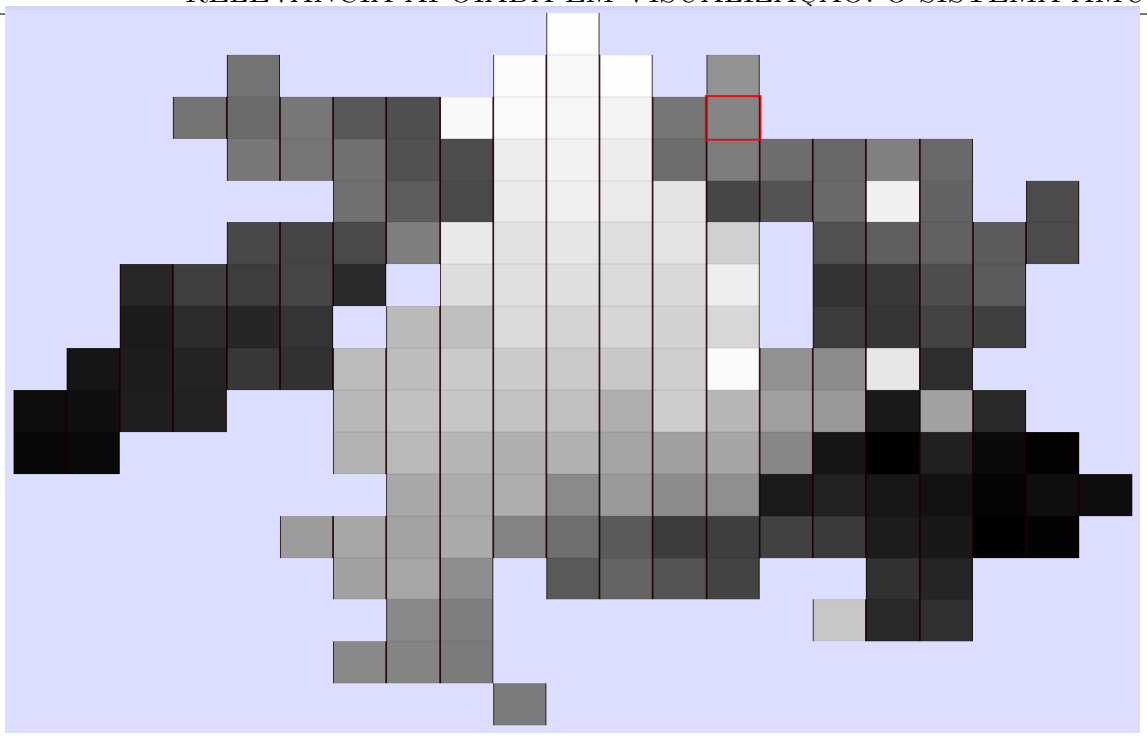

Figura 3.5: Visualização gerada com a inserção de 300 elementos gerados aleatoriamente, sendo que cada elemento pode assumir 256 valores diferentes.

diferença de tons entre os elementos, 0 sendo preto e 255 branco. As visualizações geradas são similares à da Figura 3.5.

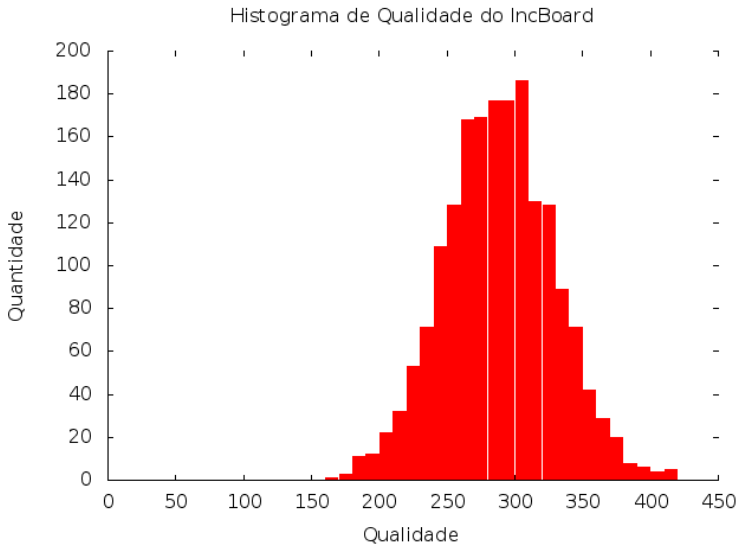

Figura 3.6: Histograma da medida de qualidade aferida em 1.000 representações aleatoriamente geradas e calculadas com a Equação 2.1

No experimento que calcula a variável W_{err} através da Equação 2.1, a média e a variância da distribuição normal foram de 288 e 1681, respectivamente. Utilizando a

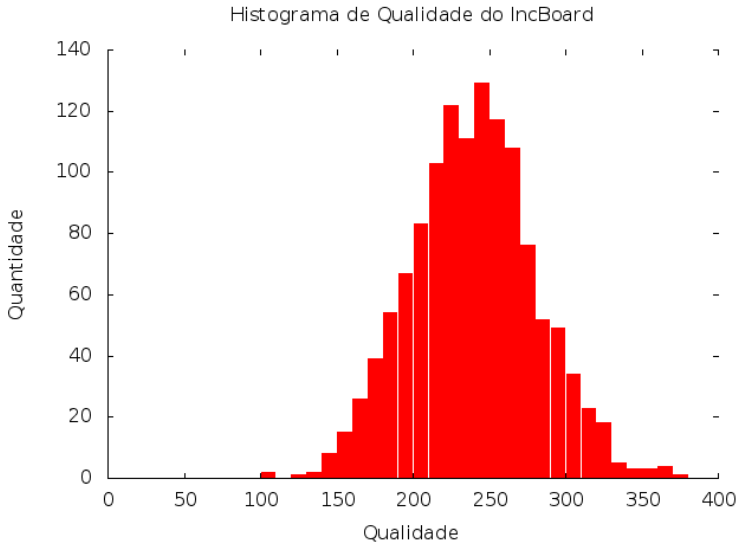

Figura 3.7: Histograma da medida de qualidade aferida em 1.000 representações aleatoriamente geradas e calculadas com a Equação 3.4

Equação 3.4, média e variância foram 239 e 1651. As Figuras 3.6 e 3.7 são os histogramas das qualidades Q obtidas utilizando cada uma das equações.

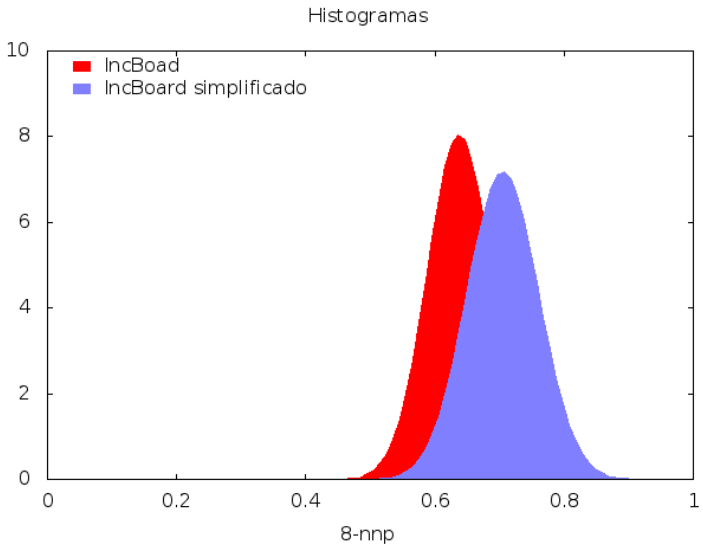

Figura 3.8: Distribuições obtidas a partir do experimento envolvendo 1.000 reprsentações aleatórias geradas por cada uma das equações 2.1 e 3.4, utilizando a medida *Neighboring Hit*, 8-nnp, (Paulovich et al., 2008)

Para medir o *Neighboring Hit* das visualizações geradas foi necessário fazer a seguinte consideração. As 256 possibilidades de cores que um elemento pode assumir estão distribuídas em quatro classes, cada uma com igual número de possibilidades. A primeira classe contém os elementos de 0 a 63, a segunda possui elementos de 64 a 127, a terceira

vai de 128 a 191 e a última de 192 a 255. A Figura 3.8 possui as distribuições encontradas para cada uma das versões do IncBoard.

Apesar da variância nos dois casos ser bastante similar, a nova Equação 3.4 gerou visualizações de melhor qualidade, de acordo com as duas medidas utilizadas. Para entender o motivo da diferença de resultados, um novo experimento foi realizado. Entretanto, desta vez foi considerado apenas a equação original do IncBoard, este experimento é descrito na subseção seguinte.

3.6.3 Aferição da Qualidade do IncBoard Otimizado para Subconjunto Aleatório

Em Pinho et al. (2009), é aconselhado utilizar um subconjunto aleatório de elementos mais os elementos mais próximos, ao invés de todos os elementos presentes no tabuleiro, na Equação 2.1, para diminuir o tempo necessário para calcular as novas posições dos elementos. Uma questão a ser verificada é a variação da qualidade das visualizações para diferentes quantidades de elementos considerados em L, na Equação 2.1.

$$x = \begin{cases} 0 & \text{se } |L_9| > |L| \times \alpha \\ (|L| \times \alpha) - |L_9| & \text{se } |L_9| <= |L| \times \alpha \end{cases} \tag{3.6}$$

Foi estabelecido um limiar α, que pode variar de 0 a 1. O conjunto L, na Equação 2.1, é composto dos elementos em L_9 agregados a x elementos aleatoriamente escolhidos. O valor de x é dado pela Equação 3.6.

Neste experimento, foi utilizado $\alpha = 0$ e depois incrementos de 0.05, até que α atingisse 0.95. Para cada valor de α, 100 visualizações foram aleatoriamente geradas. A Figura 3.9 contém a evolução da média da qualidade destas visualizações para os diferentes valores de α.

A qualidade Q variou de 417 a 286. Considerando que caso Q atingisse 504, o método já teria qualidade similar a decisão aleatório, é possível dizer que escolher poucos elementos para compor L, na Equação 2.1, causa grande degradação na qualidade da visualização.

3.6.4 Conclusões

Na Figura 3.9, quando L_9 é utilizado na Equação 2.1, o resultado é distante do encontrado quando utilizado L_9 na Equação 3.4. A diferença entre as duas equações citadas está na ponderação dos termos. Os resultados apontam que ponderar os pesos degrada a qualidade da visualização.

No experimento da subseção 3.6.3 confirmamos que utilizar todos os elementos na Equação 2.1, ao invés de usar apenas uma amostra, de fato gera visualizações de maior

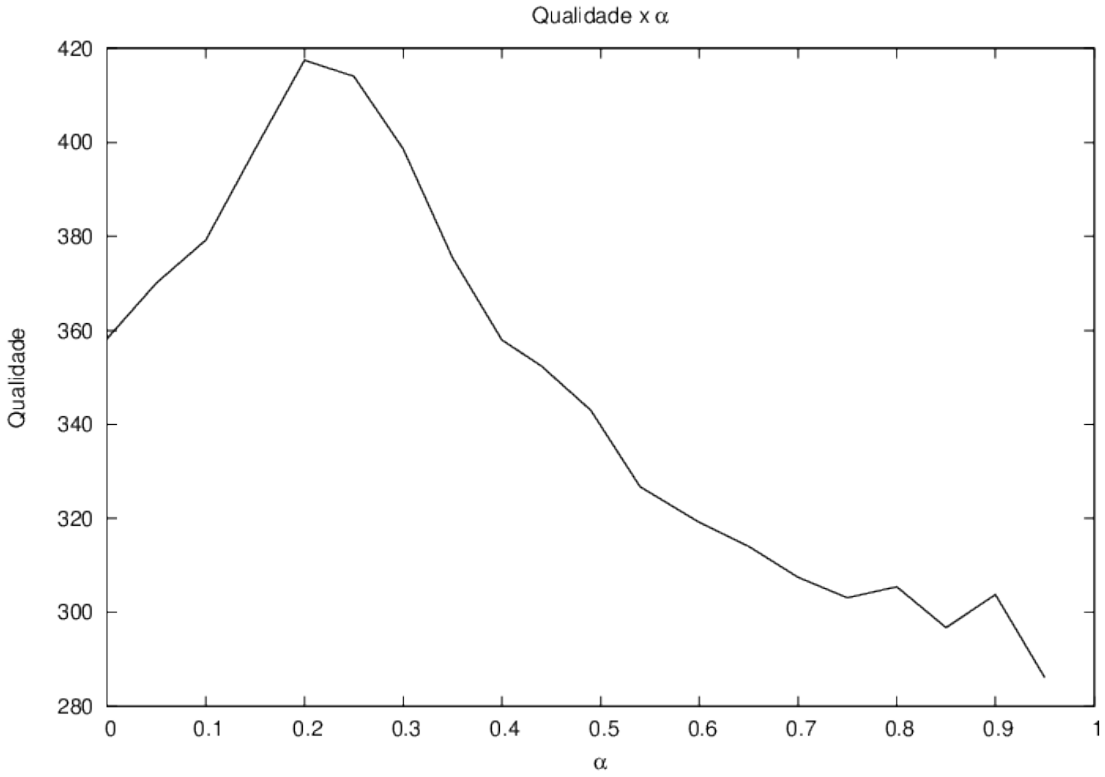

Figura 3.9: Variação da qualidade para diferentes valores de α.

qualidade. É necessário considerar que as medidas utilizadas não consideram a posição relativa entre elementos distantes, apenas a relação de um elemento com seus vizinhos. Da forma como foi avaliada, a Equação 3.4 apresenta a melhor relação entre custo e benefício.

Em relação à alteração da complexidade do algoritmo de inserção, considerando que o algoritmo original, o IncBoard clássico verifica o *ranking* de todos os elementos e não necessariamente resolve a instabilidade na primeira iteração do algoritmo. Dado que no espaço bidimensional a área tem crescimento quadrático em relação ao perímetro, pode-se esperar $O(\sqrt{n})$ iterações. Logo, a inserção de um novo elemento tem complexidade $O(n^{3/2})$, na média, ou $O(n^2)$, no pior caso.

A nova abordagem considera uma quantidade fixa de elementos, ao invés de todos os elementos presentes no tabuleiro. Logo, a ordem de complexidade para a acomodação de um novo elemento é $O(\sqrt{n})$, na média, ou $O(n)$, no pior caso. Considerando os resultados obtidos com este experimento, o Amuzi utiliza a variação do IncBoard que utiliza a Equação 3.4.

3.7 Refinamento da Interface do Amuzi

Uma vez desenvolvida a versão inicial do aplicativo, foi necessário eliminar problema de usabilidade que pudesse interferir com os resultados do teste. Nesta etapa, o Amuzi foi

apresentado a um grupo limitado de usuários que utilizaram o sistema sob a supervisão de um moderador.

O usuário recebeu um conjunto específico de tarefas que deveria realizar com a ferramenta, utilizando o protocolo *thinking aloud* (Nielsen, 1993). O moderador do teste tentou identificar os problemas que os usuário tiveram ao utilizar a interface.

O objetivo deste teste foi utilizar os problemas diagnosticados para elaborar alterações na interface que facilitassem a utilização da ferramenta. A elaboração deste teste de usabilidade está no Apêndice A.

É importante ressaltar que deficiências de usabilidade podem favorecer um modo de busca, invalidando os resultados aferidos acerca da performance de cada modo de busca. Problemas de usabilidade que possuem este potencial tiveram maior prioridade para serem corrigidos.

Nesta fase foram identificados nove problemas de usabilidade que, foram julgados ter a real necessidade de serem corrigidos e eram passíveis de serem corrigidos. Dos nove erros apontados pelos usuários, sete deles foram apontados por mais de um usuário. Os nove erros de usabilidade foram corrigidos. O Apêndice C contém a lista de erros que foram encontrados e corrigidos, durante esta fase de testes com usuários.

3.8 Avaliação da Abordagem Proposta

Para avaliar a abordagem proposta foi necessário construir um sistema de busca que pudesse servir de comparação para a abordagem que utiliza o IncBoard. Sendo assim, foi implementado a abordagem clássica de IR, na qual o usuário insere um conjunto de palavras chave que identificam o alvo de sua busca e o sistema apresenta uma lista de possíveis resultados.

Com a busca utilizando o IncBoard e a abordagem clássica em mãos, é necessário determinar sob quais moldes os testes serão executados e quais critérios serão adotados para comparar as duas abordagens, isto é, quais medidas serão utilizadas.

Com base nos objetivos da interface, é possível compará-las utilizando os seguintes medidores:

- quantidade de usuários que deliberadamente preferiram um modo de busca;
- quantidade e objetos adicionados a coleção por usuário, por modo de busca.

Após instrumentar o Amuzi para coletar estas medidas, é necessário obter uma quantidade considerável de usuários utilizando o sistema. É esperado que as diferentes características que cada modo de busca possui seja refletido nas medidas elaboradas.

Outro ponto a ser avaliado é o comportamento do IncBoard mediante matrizes de similaridade esparsas. Tendo em vista melhor entendimento do IncBoard, o Amuzi foi instrumentado para guardar informações acerca da quantidade da esparsidade original de cada matriz e da esparsidade da matriz apresentada ao IncBoard, após a extrapolação descrita na seção 3.5.

Capítulo

4

Resultados

As hipóteses levantadas neste trabalho podem ser divididas em duas partes. A primeira é a plausibilidade técnica da implementação das ideias propostas, que foram tratadas no Capítulo 3. Neste capítulo, é tratado o segundo conjunto de hipóteses, que abrange o efeito que a abordagem proposta tem sobre os usuários, comparado à abordagem clássica.

Após o desenvolvimento da primeira versão estável do Amuzi, o sistema foi disponibilizado e usuários foram convidados a utilizar o sistema. Parte das ações dos usuários no sistema foram gravadas. Os resultados apresentados neste capítulo baseiam-se na utilização feita por usuários entre 15/12/2013 e 17/01/2014.

Durante este período, 384 usuários se registraram no Amuzi e geraram 1840 ações que foram consideradas nestes resultados. Apenas a interação realizada por usuários que se registraram a partir do dia 15/12/2013 foi considerada. Esta medida foi tomada para garantir que apenas usuários que não tiveram contato prévio com o Amuzi fossem considerados.

É importante mencionar que a maior parte dos usuários que utilizaram o sistema são usuários que responderam a convites enviados. Em especial, foram convidados usuários do sistema GitHub[1] que, deliberadamente tornaram seus endereços de email publicamente disponíveis.

O GitHub reune usuários que desenvolvem projetos de código aberto. Foi considerado que há uma grande chance de que usuários do GitHub se interessem em contribuir para este

[1]GitHub é um serviço de hospedagem de projetos compartilhado, específico para projetos que usam Git (`https://github.com`)

projeto. O Amuzi é um projeto inteiramente aberto, assim como a maioria dos projetos hospedados pelo GitHub. Além disso, a finalidade do convite tem o caráter científico de testar as hipóteses levantadas neste trabalho.

A cada usuário selecionado, foi enviado um email, informando os propósitos do projeto e explicando que a simples utilização do sistema contribuiria para a resolução desta pesquisa. O conteúdo deste email está no Apêndice B.

No total, 4.300 usuários foram convidados. 236 dos destinatários foram encontrados nos dias seguintes nos registros do Amuzi. É importante ressaltar que há a possibilidade de usuários terem utilizado um endereço de email para se registrar no GitHub e outro para se registrar no Amuzi. Logo, a quantidade de usuários provenientes dos convites enviados tende a ser maior do que o que foi possível aferir.

Um registro é gravado toda vez que um usuário adiciona uma música ou um álbum à sua coleção. Neste tipo de *log*, as seguintes informações são gravadas:

- identificador do usuário;
- identificador da janela;
- modo de visualização utilizado;
- identificador do álbum ou da música;
- identificador da busca;
- profundidade do elemento adicionado.

Cada usuário que se identifica no sistema recebe um ID único. Caso o usuário mantenha o sistema aberto em mais de uma janela, ou em mais de um computador, é necessário identificar as diferentes janelas para que seja possível distinguir os registros gerados por janelas diferentes. O Amuzi disponibilizou ao usuário o modo de busca padrão e o método de buca que utiliza o IncBoard e o módulo de RF. Quando o usuário se registra no sistema, ele é aleatoriamente atribuído a um dos modos de busca. Cada álbum e música também possuem números de identificação que são únicos, no sistema.

A profundidade da busca é um parâmetro utilizado apenas com o IncBoard. Ao elemento central da busca é atribuído profundidade 0. Quando elementos similares são inseridos no sistema, a profundidade é alterada para 1. Quando o usuário insere um elemento em sua coleção, o sistema utiliza o módulo de RF para incluir elementos similares ao elemento inserido. A cada conjunto de elementos inseridos, a profundidade é incrementada em uma unidade.

Guardar a profundidade de um elemento inserido permite verificar o quanto o módulo de realimentação de relevância foi de fato utilizado. Baixa utilização deste módulo refletiria em extrema predominância de elementos de profundidade 0 e 1, nos registros.

As seções seguintes tratam dos diferentes aspectos avaliados. A Seção 4.1 trata do engajamento dos usuários de cada um dos modos de busca. Na Seção 4.2 são discutidos os dados coletados referente a utilização da técnica de RF, pelo usuário. A Seção 4.3 trata da relação entre a esparsidade das matrizes de similaridade produzidas e a interação do usuário com o IncBoard. Por fim, a Seção 4.4 discute o significado dos resultados obtidos e os possíveis entendimentos que podem ser extraídos.

4.1 Engajamento dos Usuários com o IncBoard

Tabela 4.1: Métricas da quantidade de objetos (músicas e álbuns) adicionados por usuário, por método de busca.

Métrica	Clássico	IncBoard	IncBoard / Clássico (%)
Média	2,5	3,1	+ 23,9
Variância	5,5	14,5	+ 164,6
Mediana	2,0	2,0	+ 0,0
Média $Q_{2,3}$	1,7	1,9	+ 8,6
Variância $Q_{2,3}$	0,5	0,4	- 0,9

No total, foram gravados 692 registros de inserção de música/álbum à coleção do usuário, utilizando o método tradicional de busca ou o baseado no IncBoard. Com o objetivo de compreender o conjunto de dados coletados, diferentes métricas foram aferidas. A Tabela 4.1 contém média, variância e mediana da quantidade de músicas e álbuns adicionados por usuário, em cada modo de busca. Também apresenta média e variância dos quartís 2 e 3.

A Figura 4.1 apresenta a evolução da razão entre objetos inseridos utilizando o IncBoard e objetos inseridos usando o método clássico, por usuário. O eixo da abscissa representa novos objetos sendo inseridos. Estão ordenados de forma cronológica, de acordo com ordem em que foram inseridos pelos usuários. É possível verificar a convergência para a medida com o IncBoard $23,9\%$ superior à medida com o método clássico, conforme exposto na Tabela 4.1.

Isto indica que, no modo de busca com o IncBoard, usuários tendem a adicionar $23,9\%$ mais músicas e álbuns que usuários utilizando a abordagem tradicional.

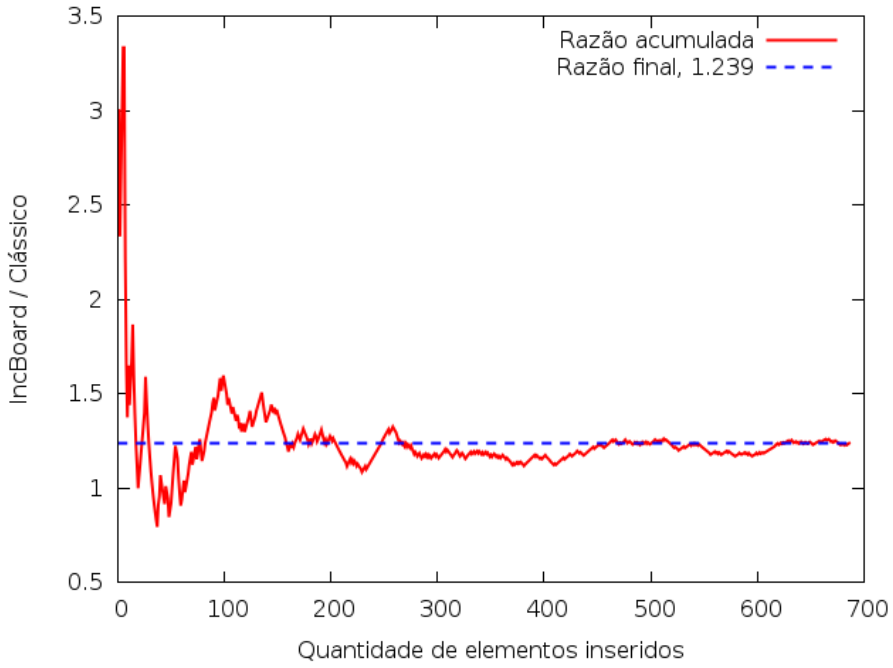

Figura 4.1: Evolução da razão da taxa de objetos inseridos por usuário, por método do IncBoard pelo método clássico.

4.2 Utilização do Módulo de RF

A cada vez que um usuário adiciona uma música ou álbum, utilizando o IncBoard, fica registrado a profundidade da realimentação de relevância atual do tabuleiro. Profundidade do RF no IncBoard é calculada da seguinte forma: após o usuário realizar a busca inicial, a profundidade é 0, pois apenas o elemento central está presente no tabuleiro. Quando os elementos similares ao elemento central da busca são inseridos no tabuleiro, a profundidade é alterada para 1.

Cada vez que o usuário insere um novo elemento do IncBoard, o sistema adiciona um novo conjunto de elementos ao tabuleiro e incrementa o valor da profundidade em 1. O valor da profundidade só volta a 0 quando o usuário realiza uma nova busca.

Apesar dos demais registros começarem a ter sido gravados a partir do dia 15/12/2013, a informação da profundidade do RF no IncBoard começou a ser gerada apenas a partir de dia 05/01/2014. Por isso, a soma dos itens coletados por usuários nestes registros é menor que a quantidade abordada nas demais seções deste capítulo.

A Figura 4.2 possui um gráfico que mostra a quantidade de elementos adicionados, por usuário, em cada nível de profundidade. É possível verificar a importância do papel que a técnica de RF desempenha na busca do usuário. Na maioria dos casos, o usuário está de fato interessado no alvo principal de sua busca. Entretanto, uma pequena parcela das músicas e álbuns foram inseridas utilizando as resposta vindas pelo módulo de RF.

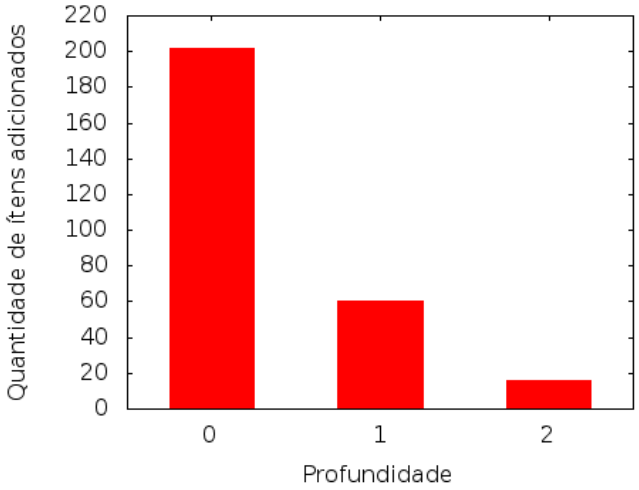

Figura 4.2: Quantidade de elementos inseridos em cada estado de profundidade do módulo de RF, através do IncBoard.

Tabela 4.2: Quantidade de elementos inseridos em cada nível de profundidade do RF e porcentagem sobre o total de elementos inseridos utilizando o IncBoard.

Profundidade	Quantidade	%
0	202	72,6
1	60	21,6
2	16	5,8
Total	278	100.00

A Tabela 4.2 contém os valores obtidos em cada um dos níveis de RF. Apenas $5,8\%$ das buscas utilizaram resultados retornados pelo RF. Alguns fatores podem ter contribuído negativamente para a utilização do módulo de RF.

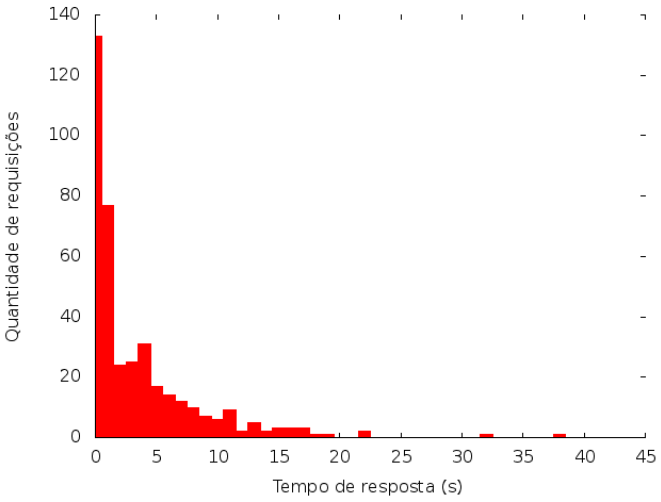

Figura 4.3: Histograma do tempo de resposta para cálculo da matriz de similaridades.

Após o usuário adicionar um objeto a sua coleção, o Amuzi espera $4s$ para começar a buscar elementos similares, o que envolve calcular a matriz de similaridade. Apesar do sistema de *cache*, o tempo para processar a requisição pode ser alto. A Figura 4.3 contém o histograma que relaciona a distribuição do tempo necessário para calcular as respostas para o cálculo da matriz de similaridade.

O tempo prolongado em realimentar o sistema pode ter diminuído as chances do usuário utilizar resultados provenientes do sistema de RF.

Outra possibilidade é que o resultado obtido seja o número certo a se esperar de utilização do módulo do RF, considerando que há grande chances do usuário estar buscando uma música em específico e não estar interessado em explorar músicas similares.

4.3 Relação entre Esparsidade da Matriz de Similaridade e Registros de Utilização do IncBoard

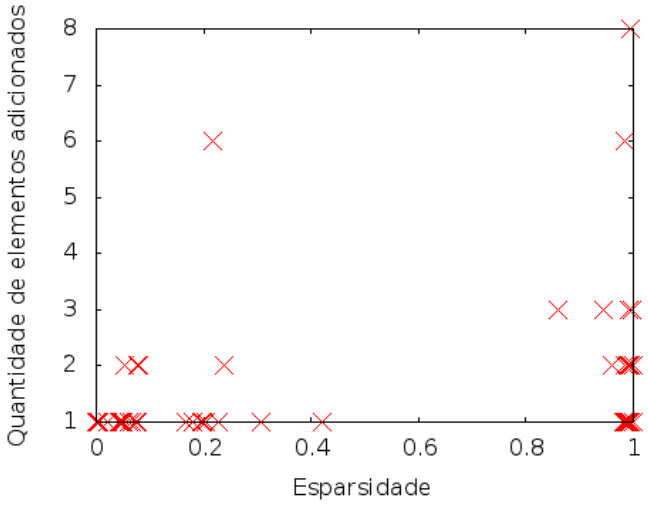

Figura 4.4: Amostra que relaciona quantidade e esparsidade de uma matriz de similaridade com a quantidade de elementos adicionados pelo usuário.

Não foi possível encontrar nenhuma tendência na relação entre a esparsidade da matriz de similaridade utilizada e o engajamento do usuário em adicionar objetos. A Figura 4.4 contém dados de buscas feitas pelos usuários utilizando o método de busca com o IncBoard. Em cada busca, foi observado a esparsidade da matriz de similaridades e a quantidade de elementos que o usuário adicionou a sua coleção, provenientes desta busca.

Nenhuma tendência pode ser identificada nestes dados. Duas possibilidades podem ser levantadas. A primeira hipótese é a de que a quantidade de dados é insuficiente para identificar a tendência. Foram utilizados dados provenientes de 52 buscas. A segunda hipótese é a de que o IncBoard é robusto a matrizes de alta esparsidade. Ainda com

muitos elementos faltando na matriz, a técnica é capaz de organizar os elementos com boa qualidade.

Na Seção 3.5, matrizes foram aleatoriamente geradas com o objetivo de estimar a diminuição de esparsidade da matriz de similaridade. Com os dados recolhidos das buscas realizadas pelos usuário, é possível ter uma estimativa mais precisa sobre a melhora gerada.

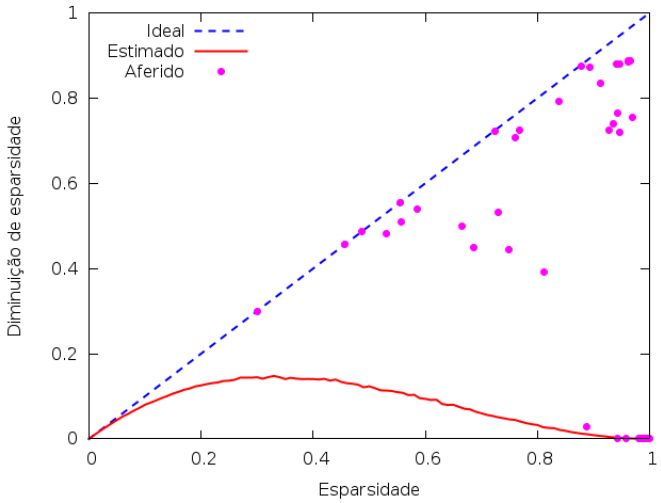

Figura 4.5: Diminuição de esparsidade da matriz de similaridades. O gráfico contém a diminuição estimada, a aferida e a função ótima.

A Figura 4.5 contém os dados estimados da Figura 3.4, os dados aferidos nos registros de utilização do Amuzi e a melhoria ótima que poderia ser obtida. A melhoria ótima ocorre quando toda a esparsidade consegue ser reduzida. Isto é, se a esparsidade é $0,3$, então o método aplicado consegue reduzir em $0,3$, a matriz não possui nenhum elemento nulo.

A redução da esparsidade foi muito maior do que o esperado em quase todos os casos. Foi analisado o modo como o Amuzi consulta o Last.fm e verificado mais profundamente os registros. Ao conseguir a informação de similaridade dos N elementos mais similares do objeto alvo, o método descrito na Seção 3.5 é capaz de extrapolar todos valores da matriz de similaridade.

Suponha que a matriz M, quadrada e sem valores negativos, possua todos os elementos da coluna i e da linha i diferentes de zero. O resultado da multiplicação $M \times M$, será uma matriz sem valores nulos.

Em alguns casos, a esparsidade da matriz é muito alta e não ouve diminuição significativa. Nestes casos, foi apurado que o Last.fm não possuía muitas informações sobre elementos similares ao elemento buscado.

4.4 Considerações Finais

É importante considerar as condições sob as quais os usuários utilizaram o sistema. Apesar do teste ter sido conduzido de forma remota e assíncrona, sem a utilização de um ambiente especial, tornando mais realista as condições do teste, os usuários podem ter utilizado o sistema com a intenção de fazer testes, não necessariamente com o objetivo de buscar por músicas. É necessário considerar que este fator pode ter influência sobre os resultados aferidos.

A diferença de engajamento dos usuários nos dois modos de busca foi positivo para o método proposto neste trabalho. Usuários tendem a encontrar mais resultados relevantes utilizando exploração visual com RF do que utilizando a abordagem tradicional.

Apesar de não ter sido possível relacionar a qualidade da matriz de similaridade com o engajamento do usuário, foi possível verificar que a extrapolação proposta na Seção 3.5 foi muito mais eficiente do que o previsto, em diminuir a esparsidade da matriz.

Capítulo

5

Conclusão

Mecanismos de busca estão em constante evolução. Não apenas os de propósito geral mas também os de domínio específico. Como exemplo desta evolução na indústria é possível citar:

- Google Scholar – que tem foco em produção científica;
- Google Reader – cujo domínio de interesse são notícias;
- Imagery [1] – Busca de imagens.

Mecanismos de busca de domínio específico tem vantagem sobre os buscadores genéricos pois são projetados com mais conhecimento sobre as informações que se dispõem a oferecer aos usuários. Isso pode levar a interfaces gráficas especialmente desenhadas para determinado conjunto de dados, que exploram as especificidades dos dados tratados para expor maior significado aos usuários.

A Web também está em constante evolução. Novos modos de usar a Internet surgem com grande frequência. Recuperação de informação na Web tem que acompanhar este desenvolvimento para prover buscadores condizentes com os novos recursos.

Uma questão sensível do projeto relaciona-se com sua avaliação. Os sistemas envolvidos neste projeto de mestrado demandam avaliação com usuário. Tais avaliações consomem muito tempo e são metodologicamente complexas. Assim, testes de usabilidade

[1] `http://elzr.com/imagery`

foram utilizados não com o objetivo de comparar as diferentes abordagens implementadas do ponto de vista de eficiência computacional e ordem de complexidade, mas com o objetivo de entender o impacto que estas abordagens têm sobre os usuários.

Este projeto de mestrado trata diretamente com técnicas de RF, visualização computacional e IR. Foram implementadas técnicas das três áreas. Entretanto o principal foco deste projeto é em utilizar as técnicas de RF e visualização para melhorar o processo de recuperação de informação.

Em relação aos objetivos do projeto, as seguintes afimações podem ser feitas. Este trabalho de mestrado foi bem sucedido em mostrar que a abordagem proposta é tecnicamente viável. Ainda nos casos em que é proibitivo armazenar as similaridades entre elementos em memória (seja principal ou secundária) a utilização de *cache* é relativamente eficiente.

Os registros de utilização dos usuários, discutidos no Capítulo 4, indicam que o método de busca que utiliza o IncBoard com o módulo de RF induz o usuário a adicionar mais objetos em sua coleção. Com isto, temos evidências que apontam para que a combinação de exploração visual com RF ajuda os usuários a explorarem mais resultados que consideram relevantes.

Entretanto, é importante observar que os resultados não são conclusivos à respeito da qualidade do método de busca que utiliza o IncBoard. Outros fatores podem ter mascarado os resultados: elementos da interface não necessários aos métodos de busca avaliados, e o fato do usuário ter sido convidado a utilizar o sistema, ao invés de ter utilizado por interesse espontâneo.

Em relação à utilização da técnica de realimentação de relevância, apenas 5.76% das músicas e álbuns foram inseridos com os resultados provenientes do módulo de RF. Este número poderia ser maior se a implementação do módulo de RF respondesse em um tempo menor. De qualquer forma, 5.76% não é um número muito baixo, considerando que já é de se esperar que usuários estejam buscando por músicas específicas, ao invés de estarem abertos a explorar resultados similares.

Ao passo que este trabalho responde a algumas questões relacionadas ao emprego destas técnicas, também levanta outras:

- Se fossem implementados sistemas semelhantes em outros contextos (como busca de artigos científicos ou filmes) quais resultados seriam obtidos?
- Há alguma forma de prever o efeito que o determinado sistema terá, baseado apenas nas características do contexto (como número de elementos na base de dados e eficiência em representar um elemento através de um ícone)?
- Qual seria o perfil de utilização do módulo de RF se sua implementação conseguir responder à requisições em um tempo abaixo de um segundo?

- O IncBoard tenta minimizar o erro W_{err}, referente à Equação 3.4. Qual o efeito deste erro sobre os usuários? O quão sensível os usuários são ao W_{err}?

A hipótese mais provável para a questão de implementar este conjunto de técnicas em outros contextos é a de que vai depender da capacidade de expressar elementos deste contexto através de ícones. Em um sistema de busca de imagens, por exemplo, é possível construir um ícone que seja uma miniatura da imagem original. Este ícone é uma excelente representação da imagem e é de se esperar que exploração visual com RF gere bons resultados no contexto de busca de imagens.

Por outro lado, em um buscador de artigos científicos, a tradução do elemento a ser buscado para ícones, pode não ser tão direta. Usuários terão mais dificuldades em associar as representações aos objetos o que pode diminuir a eficiência do sistema.

Apesar da relação entre a eficiência deste conjunto de técnicas e a capacidade de representar elementos por ícones ser a mais intuitiva, não se sabe quais outros fatores podem ter influência.

O projeto também teve subprodutos tecnológicos. Durante o desenvolvimento deste trabalho, 16 pessoas passaram a acompanhar o progresso do projeto. 27 desenvolvedores bifurcaram o projeto para suas áreas de trabalho[2], no GitHub. Destes, dois já enviaram alterações a serem agregadas ao projeto. Diversos usuários também reportaram erros ou sugestões de melhorias que podem ser aplicadas ao sistema.

Fora do GitHub, outros dois desenvolvedores propuseram e estão trabalhando em *layouts* e logotipos novos. Parte destes desenvolvedores também estão ajudando a traduzir o sistema para outras línguas. Em outro caso, usuários do GitHub propuseram formas de monetizar o sistema.

O projeto também traz benefícios para o ICMC, pois representa uma plataforma de teste para o desenvolvimento de novas pesquisas. Testes nas áreas de visualização de dados, recuperação de informações e interface humano-máquina poderão ser executados sobre o Amuzi.

Como trabalhos futuros, desenvolver um sistema de busca utilizando as mesmas técnicas, com a finalidade de prover um sistema de busca de imagens. Estudar a interação entre usuários e este sistema pode esclarecer várias dúvidas. É esperado que no domínio de imagens, os resultados sejam mais favoráveis à abordagem investigada.

Outro ponto importante é alterar a técnica de busca utilizada na funcionalidade de autocompletar para que esta contemple erros de digitação, acentuação e ordene os elementos por relevância, não por ordem alfabética, como é hoje.

[2]A bifurcação é um recurso que os desenvolvedores utilizam para desenvolver o projeto de forma paralela.

Também é interessante continuar monitorando a atividade dos usuários, no Amuzi, com o objetivo de entender melhor o modo como a abordagem proposta está sendo utilizada e aprimorar a técnica.

Com mais dados da utilização de usuários, também será possível entender melhor a variação de qualidade das visualizações geradas pelo IncBoard, de acordo com as extrapolações feitas e a esparsidade da matriz de similaridades utlilizada.

O aluno de mestrado publicou a investigação desta abordagem no *6th Workshop for Ph.D. Students at CIKM 2013* (PIKM 2013), *Workshop* que acontece dentro do *ACM International Conference on Information and Knowledge Management* (CIKM 2013)(Melo e Lopes, 2013). A investigação feita incluindo todos os resultados e análises será submetida ao Workshop de Teses e Dissertações (WTD).

Dado a importância de um sistema de IR eficiente e os resultados obtidos neste trabalho, é esperado que este trabalho de mestrado seja seguido de vários outros trabalhos explorando os frutos que RF com exploração visual podem gerar, aplicados à IR. Ainda há muitas possibilidades promissoras a serem investigadas, como por exemplo, a aplicação deste conjunto de técnicas em outros domínios.

Em relação à similaridade entre os elementos, seria interessante investigar a possibilidade de utilizar técnicas que personalizam os resultados, por usuário. Por exemplo, as músicas A e B podem ter similaridade global $0,2$ (que considera a similaridade média dos usuários de todo o Last.fm) mas ter similaridade muito maior para determinado usuário. Para este usuário o valor da similaridade deve ser o valor considerado por ele, não o valor global.

Em um sistema de busca que é integrado a uma rede social, podem ser considerados as relações de amizade entre os usuários, não apenas as medidas de similaridade dos objetos. Isto tem implicações tanto no módulo de RF quanto na parte de exploração visual.

Referências

Amazon ec2 instances. http://aws.amazon.com/ec2/instance-types/, 2013.

Full-text search. http://dev.mysql.com/doc/internals/en/full-text-search.html, last visited on 16/08/2013, 2013.

Álvares, A. C. *Extração de informação de artigos científicos: uma abordagem baseada em indução de regras de etiquetagem.* Dissertação de Mestrado, ICMC/USP, São Carlos/SP - Brasil, 2007.

Andreasen, M. S.; Nielsen, H. V.; Schrøder, S. O.; Stage, J. What happened to remote usability testing?: an empirical study of three methods. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, New York, NY, USA: ACM, 2007, p. 1405–1414 (*CHI '07*,).
Disponível em http://doi.acm.org/10.1145/1240624.1240838

Baeza-Yates, R. A.; Ribeiro-Neto, B. A. *Modern information retrieval - the concepts and technology behind search, second edition.* Pearson Education Ltd., Harlow, England, 2011.

Brin, S.; Page, L. The anatomy of a large-scale hypertextual web search engine. In: *Proceedings of the seventh international conference on World Wide Web 7*, WWW7, Amsterdam, The Netherlands, The Netherlands: Elsevier Science Publishers B. V., 1998a, p. 107–117 (*WWW7*,).
Disponível em http://dl.acm.org/citation.cfm?id=297805.297827

Brin, S.; Page, L. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, v. 30, p. 107–117, 1998b.
Disponível em http://dx.doi.org/10.1016/S0169-7552(98)00110-X

Castillo, C. Effective web crawling. *SIGIR Forum*, v. 39, p. 55–56, 2005. Disponível em http://doi.acm.org/10.1145/1067268.1067287

Chen, C. Citespace ii: Detecting and visualizing emerging trends and transient patterns in scientific literature. *J. Am. Soc. Inf. Sci. Technol.*, v. 57, n. 3, p. 359–377, 2006. Disponível em http://dx.doi.org/10.1002/asi.v57:3

Felizardo, K. R.; Nakagawa, E. Y.; Feitosa, D.; Minghim, R.; Maldonado, J. C. An approach based on visual text mining to support categorization and classification in the systematic mapping. In: *Proceedings of the 14th International Conference on Evaluation and Assessment in Software Engineering*, EASE'10, Swinton, UK, UK: British Computer Society, 2010, p. 34–43 (*EASE'10*,). Disponível em http://dl.acm.org/citation.cfm?id=2227057.2227062

Gantz, J. F.; Reinsel, D. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC*, 2012.

Garshol, L. M.; Moore, G. *ISO 13250-2: Topic Maps — Data Model*. Final draft, ISO/IEC, 2005. Disponível em http://www.isotopicmaps.org/sam/sam-model/

Grinstead, C. M.; Snell, J. L. *Introduction to probability*. 2th ed. American Mathematical Society, 1997. Disponível em http://www.dartmouth.edu/~chance/teaching_aids/books_articles/probability_book/book.html

Heydon, A.; Najork, M. Mercator: A scalable, extensible web crawler. *World Wide Web*, v. 2, n. 4, p. 219–229, 1999. Disponível em http://dx.doi.org/10.1023/A:1019213109274

Honorato, D. F. *Metodologia para mapeamento de informações não estruturadas descritas em laudos médicos para uma representação atributo-valor*. Dissertação de Mestrado, ICMC/USP, São Carlos/SP - Brasil, 2008.

Horspool, R. N. Practical fast searching in strings. *Software - Practice & Experience*, v. 10, p. 501–506, 1980.

Kantardzic, M. *Data mining: Concepts, models, methods, and algorithms*. Piscataway, NJ, EUA: Wiley-IEEE Press; 1 edition, 2002.

Kemeny, J.; Snell, J. *Finite markov chains*. University series in undergraduate mathematics, repr ed. New York: VanNostrand, 1969.

Knuth, D. *The art of computer programming*, v. 3. Second edition ed. 248–379 p., 1998.

Koster, M. Robots in the web: Threat or treat? *ConneXions*, v. 9, n. 4, p. 1–8, 1995.

Koster, M. A standard for robot exclusion. http://www.robotstxt.org/wc/robots.html., 1996.
Disponível em `http://www.robotstxt.org/wc/exclusion.html`

Lamping, J.; Rao, R.; Pirolli, P. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In: *CHI '95: Proceedings of the XIII SIGCHI Conference on Human Factors in Computing Systems*, New York, NY, EUA: ACM Press/Addison-Wesley Publishing Co., 1995, p. 401–408.

Lancaster, F.; Fayen, E. *Information retrieval: On-line*. A Wiley-Becker & Hayes series book. Melville Publishing Company, 1973.
Disponível em `http://books.google.com.br/books?id=MbOzAAAAIAAJ`

Lavrenko, V.; Croft, W. B. Relevance based language models. In: *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, New York, NY, USA: ACM, 2001, p. 120–127 (*SIGIR '01*,).
Disponível em `http://doi.acm.org/10.1145/383952.383972`

Lopes, A. A.; Minghim, R.; Melo, V.; Paulovich, F. V. Mapping texts through dimensionality reduction and visualization techniques for interactive exploration of document collections. San Jose, CA, USA: SPIE, 2006, p. 60600T.
Disponível em `http://link.aip.org/link/?PSI/6060/60600T/1`

Lopes, A. A.; Pinho, R.; Paulovich, F. V.; Minghim, R. Visual text mining using association rules. In: *Computers & Graphics*, 2007, p. 316–326.

Lv, Y.; Zhai, C. Adaptative relevance feedback in information retrieval. In: *CIKM '09 Proceeding of the 18th ACM conference on Information and knowledge management*, New York, NY, USA: ACM, 2009, p. 255–264.

Manber, U.; Myers, G. Suffix arrays: A new method for on-line string searches. In: *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '90, Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1990, p. 319–327 (*SODA '90*,).
Disponível em `http://dl.acm.org/citation.cfm?id=320176.320218`

Manning, C. D.; Raghavan, P.; Schütze, H. *Introduction to information retrieval*. New York, NY, USA: Cambridge University Press, 2008.

McGee, M. By the numbers: Twitter vs. facebook vs. google buzz. http://searchengineland.com/by-the-numbers-twitter-vs-facebook-vs-google-buzz-36709, 2010.

Melo, D. O. d.; Lopes, A. d. A. Data visualization and relevance feedback applied to information retrieval. In: *Proceedings of the Sixth Workshop on Ph.D. Students in Information and Knowledge Management*, PIKM '13, New York, NY, USA: ACM, 2013, p. 27–32 (*PIKM '13*,).
Disponível em http://doi.acm.org/10.1145/2513166.2513178

Miller, D. H.; Leek, T.; Schwartz, R. A hidden markov model information retrieval system. In: *22nd ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, 1999, p. 214–221.

Minghim, R.; Paulovich, F. V.; de Andrade Lopes, A. Content-based text mapping using multi-dimensional projections for exploration of document collections. SPIE, 2006, p. 60600S.
Disponível em http://link.aip.org/link/?PSI/6060/60600S/1

Monard, M. C.; Baranauskas, J. A. *Conceitos sobre aprendizado de máquina*, v. 1. 1 ed. Manole, 89–114 p., 2003.

National Imagery and Mapping Agency *Department of defense world geodetic system 1984: its definition and relationships with local geodetic systems*. Relatório Técnico TR8350.2, National Imagery and Mapping Agency, St. Louis, MO, USA, 2000.
Disponível em http://earth-info.nga.mil/GandG/publications/tr8350.2/tr8350_2.html

Nielsen, J. *Usability engineering*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

Paulovich, F. V.; Nonato, L. G.; Minguim, R.; Levkowitz, H. Least square projection: a fast high-precision multidimensional projection technique and its application to document mapping. In: *IEEE Trans Vis Comput Graph*, 2008, p. 564–575.

Paulovich, F. V.; Oliveira, M. C. F.; Minghim, R. The projection explorer: A flexible tool for projection-based multidimensional visualization. In: *Proceedings of the XX Brazilian Symposium on Computer Graphics and Image Processing - SIBGRAPI*, Belo Horizonte, Brazil: IEEE CS Press, 2007, p. 27–36.

Peng, G. Cdn: Content distribution network. *CoRR*, v. cs.NI/0411069, 2004.

Pinho, R.; Oliveira, M. C.; Andrade Lopes, A. An incremental space to visualize dynamic data sets. *Multimedia Tools Appl.*, v. 50, n. 3, p. 533–562, 2010.
Disponível em `http://dx.doi.org/10.1007/s11042-010-0483-5`

Pinho, R.; de Oliveira, M. C. F. Hexboard: Conveying pairwise similarity in an incremental visualization space. In: *IV*, 2009, p. 32–37.

Pinho, R.; de Oliveira, M. C. F.; de A. Lopes, A. Incremental board: a grid-based space for visualizing dynamic data sets. In: *SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing*, New York, NY, USA: ACM, (Best paper award in the Information System Theme), 2009, p. 1757–1764.

Rezende, S. *Sistemas inteligentes: Fundamentos e aplicações.* Barueri, SP, Brasil: Editora Manole, 2005.

Robertson, S. E.; Jones, K. S. Relevance weighting of search terms. *J. Am. Soc. Inf. Sci.*, v. 27, n. 3, p. 129–146, 1976.
Disponível em `http://dx.doi.org/10.1002/asi.4630270302`

Rocchio, J. J. Relevance feedback in information retrieval. In: Salton, G., ed. *The SMART Retrieval System - Experiments in Automatic Document Processing*, Englewood, Cliffs, New Jersey: Prentice Hall, 1971, p. 313–323.

Salton, G.; Buckley, C. Improving retrieval performance by relevance feedback. In: *Jornal of the American Society of Information Science*, 1990, p. 288–297.

Salton, G.; Wong, A.; Yang, C. S. A vector space model for automatic indexing. *Commun. ACM*, v. 18, p. 613–620, 1975.
Disponível em `http://doi.acm.org/10.1145/361219.361220`

Silberschatz, A.; Tuzhilin, A. On subjective measures of interestingness in knowledge discovery. In: *KDD '95: Proceedings of I International Conference on Knowledge Discovery and Data Mining*, 1995, p. 275–281.

Singhal, A. Modern Information Retrieval: A Brief Overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, v. 24, n. 4, p. 35–42, 2001.
Disponível em `http://singhal.info/ieee2001.pdf`

Slonim, N.; Tishby, N. Document clustering using word clusters via the information bottleneck method. In: *SIGIR '00: Proceeding of the 23rd Annual International ACM*

SIGIR Conference on Research and development in information retrieval, New York, NY, EUA, 2000, p. 208–215.

Snyder, P. tmpfs: A virtual memory file system. In: *In Proceedings of the Autumn 1990 European UNIX Users' Group Conference*, 1990, p. 241–248.

Swartz, A. Musicbrainz: A semantic web service. *IEEE Intelligent Systems*, v. 17, n. 1, p. 76–77, 2002.

Weiner, P. Linear pattern matching algorithm. *14th Annual IEEE Symposium on Switching and Automata Theory*, p. 1–11, 1973.

Young, H.; Freedman, R.; Ford, A. *University physics with modern physics.* Addison Wesley, 2011.
Disponível em `http://books.google.com.br/books?id=hKWlcQAACAAJ`

Yu, S.; Cai, D.; Wen, J.-R.; Ma, W.-Y. Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In: *Proceedings of the 12th international conference on World Wide Web*, WWW '03, New York, NY, USA: ACM, 2003, p. 11–18 (*WWW '03*,).
Disponível em `http://doi.acm.org/10.1145/775152.775155`

Zhai, C.; Lafferty, J. D. Model-based feedback in the language modeling approach to information retrieval. In: *Proceedings of CIKM '01*, 2001, p. 403–410.

Apêndice

A

APÊNDICE 1 - Plano de Teste de Usabilidade 1

O sistema Amuzi teve suas partes testadas individualmente, entretanto não há registros da interação entre usuários que desconhecem o sistema e o Amuzi. Este teste de usabilidade tem o objetivo de mostrar possíveis erros de interface que possam atrapalhar o usuário na utilização do sistema. O teste não visa validar cada módulo individualmente, mas testar o sistema como um todo.

A.1 Objetivos deste estudo

Coletar dados sobre a utilização do sistema para avaliar a eficiência da interface. Os objetivos deste estudo são:

- Medir a eficiência do sistema para o público alvo do Amuzi;
- Identificar obstáculos em cumprir as tarefas propostas;
- Garantir que, do ponto de vista de usabilidade, ambos os métodos de busca clássico e o IncBoard foram igualmente bem implementados.

A.2 Questões de pesquisa

Adicionalmente, este teste visa responder às seguintes perguntas:

- O quão fácil e precisamente usuários conseguem buscar músicas e álbuns utilizando os dois métodos de busca?
- Com que facilidade os usuários conseguem organizar as músicas em *playlists* e gerenciar os álbuns adicionados?
- Quanto de esforço é necessário para entender a funcionalidade do desktop estendido, que pode ser navegado pelos ícones localizados na posição superior central da tela?
- Usuários conseguem identificar que, nas *playlists*, é possível trocar a ordem de execução das músicas usando *drag and drop*?
- Usuários tem dificuldade em utilizar o sistema de cadastro/login do site?
- O quão fácil é para os usuários identificar que, no painel de músicas, é possível verificar a lista de músicas de uma *playlist* ou álbum e também a lista de ações de uma música?
- O tempo de resposta da interface, nas diferentes operações, é satisfatório para os usuários?

Ao final dos testes haverão dados quantitativos:

- Erros ao buscar por músicas e álbuns;
- Erros ao tentar reproduzir um álbum ou *playlist*;
- Quantidade de usuários que não observaram a existência de funcionalidades chave;
- Erros ao utilizar a funcionalidade de *desktop* estendido.

Também haverão dados qualitativos: observações feitas pelo usuário darão indicativos de em quais pontos do sistema o usuário se sentiu confuso.

A.3 Local e estrutura

Um laptop com microfone interno. Utilização do navegador Google Chrome versão 28.0.1496.0 dev e o software gtk-recordMyDesktop para gravar *desktop* e áudio. Não haverá um local específico para realizar os experimentos. As restrições para que um local seja considerado aceitável são:

- Possuir uma conexão com banda mínima de $5mbps$ de *download* e $250kbps$ de *upload*;
- Ser uma sala fechada que permita que o usuário se concentre no experimento;
- possuir móveis para acomodar o usuário utilizando o laptop e o moderador, que aplicará o experimento.

A.4 Recrutamento dos participantes

Os voluntários escolhidos são da área de TI. Um dos diferenciais do Amuzi é ser um sistema de código aberto, característica que é fortemente apreciada por parte dos profissionais de TI. Logo, o público alvo do Amuzi é composto por desenvolvedores. É esperado que os indivíduos escolhidos representes uma amostra representativa do público alvo do Amuzi.

A.5 Método

Este estudo é exploratório por permitir que o participante utilize o sistema além das tarefas pré-definidas, com o intuito de aprender o máximo possível sobre as impressões que o sistema causou. Também será utilizado a técnica *think aloud*, na qual o participante expressa verbalmente os pensamentos que lhe ocorrem no decorrer dos testes.

Registrar a quantidade de erros cometidos pelo usuário bem como a opinião do usuário sobre as diferentes partes do sistema.

A.5.1 Script de execução de teste

A duração de cada teste tem média de 30 minutos. Aproximadamente 2 minutos de introdução, 15 minutos para a realização das tarefas e mais 13 minutos para o usuário utilizar livremente o sistema.

A.5.2 Preparativos

Iniciar o software de gravação de áudio e do *desktop* do computador utilizado no teste.

É necessário também verificar a conexão com a Internet. Algumas pessoas se sentem desconfortáveis em abrir acessar a conta de email em um computador de testes. Então, é necessário criar um conta no Amuzi para o usuário. É importante que a conta utilizada pelo usuário nunca tenha sido usada antes.

A.5.3 Introdução - 2 minutos

Na fase introdutória, o moderador irá discutir os seguintes tópicos com os participantes:

- Experiência do participante com estudos de usabilidade;
- Importância deste estudo;
- O papel do moderador;
- O protocolo que será utilizando durante o restante do teste;
- Como funciona a técnica *think aloud*.

A.5.4 Tarefas a serem executadas utilizando o think aloud - 15 minutos

Uma conta usuário e senha específico para o teste a ser realizado deve ser provida ao voluntário. Cada participante terá uma conta exclusiva. Este método tem a desvantagem de não possibilitar que o usuário teste o método de cadastro do sistema. Entretanto, tira a necessidade de interação com outros sistemas e diminui o risco do usuário utilizar senhas pessoais durante os testes.

Nesta etapa o usuário deverá cumprir as seguintes tarefas:

- Fazer login no Amuzi;
- Buscar 5 elementos musicais que tem interesse. Dentre os cinco, no mínimo um deles deve ser um álbum. São considerados elementos musicais músicas e álbuns;
- Escolher um destes elementos e reproduzí-lo;
- Trocar o método de visualização, de clássico para IncBoard ou o contrário;
- Buscar mais 5 elementos musicais;
- Informar quanto tempo de duração tem a primeira *playlist* montada;
- Informar o tempo de duração de um dos álbuns adicionados.

A.5.5 Análise exploratória - 13 minutos

- Utilizar o sistema livremente e dialogar sobre a navegação no sistema;
- Descrever problemas que teve para utilizar o sistema, pontos de dúvida, ideias de melhorias e qualquer cometário relacionado a usabilidade.

A.6 Medições

Responder às questões de pesquisa. Não serão coletados dados relativos a tempo para executar as tarefas pois é conhecido que o processo *think aloud* pode atrasar o tempo de execução das tarefas. Também serão coletados dados da taxa de sucesso na execução das tarefas.

Para cada uma das tarefas estipuladas para a etapa do *think aloud* é possível classificar a ação do usuário dentre as seguintes alternativas:

- Tarefa realizada com sucesso e utilizando os elementos de interface da maneira esperada;
- Tarefa realizada com sucesso, porém não da forma considerada ideal;
- Tarefa executada apenas após auxilio do moderador;
- Usuário não conseguiu realizar a tarefa devido a problemas de usabilidade;
- Usuário não conseguiu realizar a tarefa devido a erro de sistema.

Apesar do propósito deste teste ser descobrir falhas de sistema, estes podem ocorrer e devem interferir o mínimo possível com os propósito principal deste teste, que é descobrir problemas de usabilidade.

Serão levados em conta:

- Erro por omissão;
- Erro por ação;
- Número de tarefas completadas com e sem assistência;
- O quão apropriado é a funcionalidade para a execução da tarefa;
- Facilidade em utilizar a interface;
- Utilidade dos termos e títulos utilizados nos elementos do sistema.

A.6.1 Papel do moderador

O moderador irá apresentar o sistema ao participante e explicar os temos sob os quais o teste será realizado, a importância do mesmo e preparar o ambiente para o inicio do teste, conforme a seção de introdução do teste.

Durante a etapa de tarefas o moderador irá interferir apenas quando estritamente necessário para dar continuidade ao teste.

Na análise exploratória o participante continua sendo o único utilizando o sistema entretanto, haverá um diálogo no qual o moderador tentará extrair o máximo de informações do participante com o objetivo de 1) explicar o que pode não ter ficado claro sobre as ações do participante para completar as tarefas e 2) identificar pontos na interface que podem ser melhorados.

A.6.2 Entregáveis

- Plano de teste (este documento) descrevendo como o teste será conduzido;
- Gravações e notas realizadas durante os testes;
- Documento de resultados dos testes.

Apêndice

B

APÊNDICE 2 - Email enviado convidando usuários a testarem o sistema

```
1  De: dmelo87@gmail.com
2  Para: usuario@email.com
3  Assunto: Amuzi
4
5  Hi FirstName LastName,
6
7
8  I've found your profile at GitHub.
9  As part of my research, at the University of São Paulo, I'm proposing
10 a new interactive technique for information retrieval. This technique
11 uses data visualization and relevance feedback to provide more
12 information regarding the user's query. We have implemented the Amuzi
13 system as a proof of concept. Amuzi is an open source project,
14 available at GitHub (https://github.com/dmelo/amuzi). I'm sending this
15 message to kindly ask for your help. When a user logs in and searches
16 for a music, the system collects logs which allow me to evaluate
17 the technique.
18
19 Here is the link for Amuzi: http://amuzi.me
20
```

```
Please log in, search and listen to the music you like. It will
help me a lot on this research and hopefully you will also enjoy
using Amuzi.

If you find any error or have suggestion to improve Amuzi, please
let me know. If you are interested in knowing more about the
academic interests behind Amuzi, here is a the paper that
describes the research http://dl.acm.org/citation.cfm?id=
2513166.2513178&coll=DL&dl=GUIDE&CFID=391725934&CFTOKEN=94348372.


Best regards,
Diogo Oliveira de Melo
Phd student at University of São Paulo, Brazil
http://amuzi.me | http://diogomelo.net
```

Apêndice

C

APÊNDICE 3 - Lista de erros encontrados e corrigidos durante testes, em laboratório, com usuários

A Tabela C.1 contém a lista de erros encontrados e as soluções aplicadas. Estes erros foram encontrados durante a etapa de testes com usuários que foi realizada em laboratório e síncronamente, com a presença de um intermediador para assistir o usuário durante o teste, conforme plano de testes do Apêndice A.

Tabela C.1: Tabela com relação de erros encontrados e medidas implementadas para corrigí-los.

#	**Erro**	**Correção**
1	Ao visulizar o conjunto de resultados de uma busca, o usuário clica no botão de ação e espera que a música/álbum comece a tocar imediatamente. Ao invés disso o Amuzi apenas adiciona a música na coleção do usuário.	Existem agora dois botões de ação associados a cada resultados de busca. O ícone de *play* adiciona o ítem à coleção do usuário e o envia para execução. O ícone de adicionar apenas salva o ítem na coleção do usuário.

Continua na próxima página

TabelaC.1 – continuação da página anterior

#	Erro	Correção
2	O modo como usuários elaboram buscas não é padronizado. Diferentes formatos foram vistos, que se resumem a permutações de artista e título da música/álbum, separados ou não por hífen. Entretanto, o o Amuzi aceitava apenas buscas no formato ARTISTA - TÍTULO.	Conforme descrito na Seção 3.4, a segunda versão da funcionalidade foi elaborada para aceitar diferentes possibilidades de descrição de artista e título, separadas ou não por hífen.
3	Entre duas buscas consecutivas, havia a possibilidade resultados se entrelaçarem. Isto é, na segunda busca, os resultados que ainda estão sendo carregados na primeira busca entra no cojunto.	A cada busca realizada, é atribuído um identificador único quando um resultado é obtido o identificar da busca do resultado é comparado com o identificador da busca corrente. O resultado é exibido apenas se os números forem iguais.
4	A mensagem do tutorial de busca e as mensagens de sistema estão em sobreposição.	A mensagem do tutorial foi movida para a parte de baixo da caixa de texto de busca, ao invés da parte superior.
5	Enquanto digita a busca, se o usuário passar o cursor pelos ítens sugeridos pelo autocompletar então o texto do ítem substitui o texto do usuário. Isso demonstrou causar frustração no usuário pois o usuário perde o texto que digitou.	A funcionalidade de autcompletar foi adaptada para não disparar a ação referida caso o cursor passe pelos ítens sugeridos.
6	O usuário não identificou o *link* de compartilhar. Não associou o ícone de *link* com a ação de compartilhar.	O ícone foi substituído pelo próprio endereço da URL a ser compartilhada.

Continua na próxima página

TabelaC.1 – continuação da página anterior

#	**Erro**	**Correção**
7	Ainda após carregar as sugestões referentes ao texto provido pelo usuário, a funcionalidade de autocompletar continua exibindo a animação que indica atividade em progresso.	A animação de progresso passou a nunca ser exibida. A funcionalidade de autocompletar responde em tempo abaixo de um segundo, na marioria dos casos, evitando a necessidade da animação. Ainda que o autocompletar não responda em tempo hábil, não é necessário que o usuário espere pois não é obrigatório usar as sugestões do sistema.
8	No modo de busca clássico, usuários foram incapazes de diferenciar álbuns de músicas individuais. O sistema difere os dois ítens apenas pela cor do fundo sem apresentar legenda. Além disso, usuários com daltonísmos para as cores utilizadas ficariam incapacitados de discernir entre os dois tipos.	Foi acrescentada uma legenda no corpo do ítem especificando o tipo: *Album* ou *Track*.
9	Após clicar sobre o botão de fechar mensagem de aviso do sistema, a mensagem não fechou.	O erro foi corrigido. O sistema de mensagens foi evoluído para que cada mensagem recebesse um ID. Desta forma, funções são capazes de excluir suas próprias mensagens. Considerando a natureza assíncrona de Javascript.