Relacionando Notícias Web: Uma Abordagem Causal e Temporal

Thiago de S. Rodrigues[1,3], Wallace A. Pinheiro[1,4], Jano M. de Souza[1,2], Geraldo Xexéo[1,2]
[1]COPPE/UFRJ, Universidade Federal do Rio de Janeiro
[2]DCC-IM, Departamento de Ciência da Computação, Instituto de Matemática
[3]Petrobras Distribuidora S.A.
[4]IME, Instituto Militar de Engenharia

***Resumo*- Com o desenvolvimento da Internet, diversos jornais e revistas, que antes publicavam suas notícias apenas através de mídias impressas, hoje em dia também disponibilizam o seu conteúdo através de páginas *web*. Nessas páginas, geralmente, o texto de uma notícia é acompanhado por um conjunto de *links* que correspondem às suas notícias relacionadas. O principal problema desse cenário é que os *links* para as notícias relacionadas são criados manualmente (estáticos) e de maneira *ad hoc*, fazendo com que os leitores fiquem restritos a um pequeno conjunto de notícias pré-selecionadas. Esse artigo propõe uma nova abordagem sobre esse assunto com o objetivo de fornecer aos usuários um amplo encadeamento histórico das notícias buscadas. Para isso foi desenvolvido um mecanismo de encadeamento de notícias causal e temporal. Esse mecanismo tem a capacidade de refinar o relacionamento entre as notícias e melhorar os seus resultados à medida que for sendo utilizado, com a aplicação do conceito de sabedoria das multidões.**

I. INTRODUÇÃO

Há alguns anos, o compartilhamento de informações possuía um custo muito elevado e uma abrangência relativamente pequena, porém com o avanço tecnológico e, principalmente, com o desenvolvimento dos meios de comunicação, onde a Internet possui uma posição de destaque, esse cenário mudou completamente. Diversos jornais e revistas, que antes publicavam suas notícias apenas através de mídias impressas, hoje em dia também disponibilizam o seu conteúdo através de páginas *web*. Essas páginas possuem a imensa vantagem de permitir a atualização das informações em tempo real, deixando os leitores sempre a par do que está acontecendo no exato momento em que estão lendo a notícia.

Atualmente a forma como as notícias *web* são disponibilizadas geralmente segue um padrão específico. Primeiramente é apresentado o texto da notícia e no fim da página são fornecidos alguns *links* para as notícias relacionadas.

O principal problema dessa abordagem é que os *links* para as notícias relacionadas são criados manualmente (estáticos) e de maneira *ad hoc*, ou seja, com base na memória das pessoas que os publicam. Esse tipo de relacionamento entre notícias acaba limitando a necessidade de informação dos leitores, pois eles ficam restritos a um pequeno conjunto de notícias pré-selecionadas.

Para contornar esse problema, torna-se necessário o desenvolvimento de um mecanismo que além de relacionar um conjunto de notícias automaticamente, permita que os usuários durante uma leitura realizem buscas e naveguem por esses relacionamentos.

Este trabalho tem como objetivo propor um método de busca voltado especificamente para notícias *web* de forma a permitir que os relacionamentos dessas notícias sejam explicitados de forma natural, automática e intuitiva, substituindo satisfatoriamente a forma como o relacionamento de notícias é realizado atualmente através de *links* estáticos para notícias relacionadas.

Para isso, foi desenvolvida a ferramenta *WhySearch*, que é composta por um conjunto de módulos que atuam desde o relacionamento entre as notícias, que seria equivalente aos *links* existentes entre páginas *web*, formando um grande grafo, até a busca e exibição das notícias relacionadas. Além disso, fatores como tempo e causa também são considerados.

Outra característica relevante é a capacidade que o sistema tem de refinar o relacionamento entre as notícias e melhorar os seus resultados à medida que for sendo utilizado, com a aplicação do conceito de sabedoria das multidões [1].

A seção seguinte apresenta os assuntos relacionados a esse trabalho. Na Seção 3 é mostrado o mecanismo de encadeamento de notícias *WhySearch*. A Seção 4 destaca a arquitetura dessa ferramenta. Em seguida, esse mecanismo é avaliado através de um simples experimento na Seção 5. Por fim, na Seção 6 são apresentadas as conclusões deste trabalho e algumas tendências futuras são descritas.

II. REVISÃO DA LITERATURA

A. *Recuperação da Informação*

O significado do termo recuperação da informação (RI) pode ser bastante amplo dependendo da área e do contexto em que é utilizado. Formalmente, pode-se definir a recuperação da informação como um artifício para se encontrar documentos que satisfaçam uma necessidade de informação dentro de grandes coleções armazenadas principalmente em computadores [2].

Definida dessa forma, a recuperação da informação era considerada uma atividade em que apenas algumas pessoas estavam envolvidas: bibliotecários, assistentes jurídicos, etc. Atualmente, com o avanço tecnológico, milhares de pessoas lidam com a recuperação da informação diariamente quando elas utilizam um mecanismo de busca na *web*. A RI vem rapidamente se tornando o principal meio de acesso a informação.

Para classificar os documentos resultantes de uma busca, os sistemas de RI geralmente adotam um modelo para representar tanto os documentos quanto a consulta do usuário. Muitos modelos foram propostos ao longo dos anos, porém três modelos são considerados clássicos: booleano, vetorial e probabilístico [3].

Esses modelos apresentam estratégias de busca de documentos relevantes distintas para uma determinada consulta, onde cada documento é descrito por um conjunto de palavras-chave representativas, também conhecidas como termos de indexação, que buscam descrever o assunto do documento e sumarizar seu conteúdo de forma significativa.

B. Detecção de Tópicos

A detecção de tópicos (TDT - *Topic Detection and Tracking*) é uma área de pesquisa que investiga a organização das notícias publicadas pelos meios de comunicação com base nos eventos que as geraram. Inicialmente, a pesquisa sobre o TDT foi desenvolvida a partir de um fluxo de texto obtido das redes de notícias e de sistemas que monitoravam programas de TV e de rádio convertendo tudo o que era anunciado para texto. De forma geral, o objetivo do TDT é dividir um texto contínuo em histórias individuais para monitorá-las como eventos que não foram detectados e para classificá-las em grupos que tratam de tópicos específicos [4]. Apesar desta pesquisa não estar direcionada especificamente para esse problema, muitas técnicas aplicadas ao TDT foram utilizadas.

A principal motivação para o desenvolvimento do TDT era fornecer uma tecnologia para um sistema que iria monitorar as notícias divulgadas pelos meios de comunicação e alertar um analista, por exemplo, sobre um novo evento de interesse que pudesse estar acontecendo no mundo. Analistas de notícias geralmente estão muito interessados em saber o que está acontecendo, porém não existem métodos efetivos para copiar e tratar o gigantesco volume de informações que é disponibilizado diariamente. Em particular, não existem sistemas que fazem isso através de um monitoramento de notícias, para reportar eventos ao invés de notícias sobre um assunto particular e amplo.

Esse problema evidencia um dos pontos que tornam essa metodologia um assunto interessante. Apesar do TDT ser similar a filtragem e a recuperação da informação, ele é mais direcionado do que as noções gerais de *aboutness* e, como resultado, admite a possibilidade de aplicar análises automáticas de conteúdo mais profundas.

O TDT está dividido em cinco tarefas. Cada uma delas é visualizada como um componente que possui uma solução que ajudará a resolver o problema de organização das notícias com base em eventos [4]. As tarefas são:

- **Segmentação de uma história:** é o problema de dividir a transcrição de um noticiário em histórias individuais.

De forma geral, comparando as tarefas do TDT com a proposta deste trabalho, pode-se dizer que a segmentação de uma história não foi aplicada pelo fato das notícias serem obtidas através de *feeds* RSS, o que já garante essa segmentação de forma natural, pois eles possuem *links* para as notícias que correspondem a esses segmentos.

Feeds RSS são listas de atualização de conteúdo de um determinado site, escritas com especificações baseadas em XML. Os *feeds* são usados para que um usuário da internet possa acompanhar os novos artigos e demais conteúdos de um site ou *blog* sem que precise visitar o *site* em si. Sempre que um novo conteúdo for publicado em determinado *site*, o "assinante" do *feed* poderá lê-lo imediatamente. Atualmente, existem três principais especificações para a criação de arquivos *feed*: RSS 1.0 (RDF *Site Summary* 1.0), RSS 2.0 (*Really Simple Syndication* 2.0) e Atom. O RSS 2.0 é o formato mais utilizado atualmente [5].

- **Detecção da primeira história:** consiste em reconhecer o início de um novo tópico dentro de um conjunto de notícias.

Essa tarefa é realizada por um componente específico do *WhySearch* que utiliza a biblioteca Lucene para restringir o conjunto de notícias de interesse do usuário com base na consulta que ele forneceu. Com isso, a primeira história passa a ser a notícia que foi selecionada pelo usuário dentro desse conjunto.

O Lucene é uma biblioteca Java voltada especificamente para buscas textuais. Ele pode ser definido como um mecanismo de busca textual de alta performance, adaptável para praticamente qualquer tipo de aplicação que requeira buscas *full-text* [7]. Um exemplo disso é o fato de ser possível aplicar diversos algoritmos de processamento de texto (eliminação de *stopwords* e *stemming*). Em relação ao *WhySearch*, foi utilizado o algoritmo de *stemming* de Porter [6]. Entre as principais características do Lucene temos a busca ranqueada, diferentes tipos de consultas, busca por campo, operadores booleanos, ordenação por qualquer campo, busca em múltiplos índices com resultado unificado e buscas e atualizações simultâneas.

Os conceitos fundamentais do Lucene são: índice, documento, campo e termo. Um índice contém uma seqüência de documentos. Um documento é uma sequência de campos. Um campo é uma sequencia nomeada de termos. Um termo é uma *string* de texto. Uma mesma *string* em dois diferentes campos é considerada um termo distinto. Dessa forma, termos são representados como pares de *strings*. A primeira corresponde ao nome do campo e a segunda ao texto desse campo.

O índice armazena estatísticas sobre os termos para que as consultas sejam mais eficientes. O índice construído pelo Lucene se encaixa na família dos índices invertidos. Isso porque ele pode listar, para um termo, os documentos que o contém, ou seja, é o inverso dos relacionamentos naturais, onde os documentos listam termos [7].

- **Detecção de *clusters*:** é o problema de agrupar todas as histórias na medida em que elas são obtidas, com base nos tópicos que elas abordam.

Pode-se considerar a detecção de *clusters* como o processamento realizado para relacionar todas as notícias do repositório, onde o coeficiente de similaridade de Jaccard foi utilizado para atribuir pesos a esses relacionamentos. Dessa forma, os subconjuntos das notícias relacionadas com os maiores pesos seriam os *clusters*.

O coeficiente de Jaccard [2], usado neste trabalho para medir a similaridade entre duas notícias, é definido por |A ∩ B| / |A U B|, onde A e B representam os termos existentes nas notícias A e B.

- **Rastreamento:** consiste em monitorar o fluxo de notícias com a finalidade de encontrar histórias adicionais sobre um tópico que foi identificado através de algumas histórias de exemplo.

A tarefa de rastreamento corresponde aos relacionamentos criados entre as novas notícias adicionadas ao repositório e as notícias já armazenadas. Além disso, pode-se considerar a própria utilização da ferramenta pelos usuários como parte dessa tarefa, tendo em vista que o *WhySearch* tem a capacidade de refinar os relacionamentos entre as notícias a medida que é utilizado. Isso é possível através da aplicação da sabedoria das multidões, que é uma teoria sobre a agregação da informação em grupos, resultando em decisões que são frequentemente melhores do que as decisões tomadas por um único membro do grupo [1].

- **Detecção das ligações de uma história:** é o problema de decidir se duas histórias selecionadas randomicamente tratam do mesmo tópico. Essa tarefa não é abordada neste trabalho.

III. Uma Proposta para Relacionamento de Notícias *Web*: O Mecanismo *WhySearch*

O *WhySearch* é um mecanismo de encadeamento de notícias causal e temporal, baseado em um repositório de *feeds* RSS controlado.

Após a realização de uma busca, ao selecionarmos uma notícia, o sistema permite obter a notícia correlacionada mais próxima e assim sucessivamente através do botão *Why* (Por que). Além disso, também é possível recusar uma notícia considerada inadequada através do botão *Refuse* (Recusar). Esse artifício permite explicitar de forma clara e intuitiva toda a relação existente em um conjunto de notícias.

Outra característica de extrema importância dessa ferramenta é a capacidade que ela possui de melhorar os seus resultados à medida que for sendo utilizada. Para isso, foi aplicado o conceito de sabedoria das multidões [1].

Dessa forma, pode-se dizer que o mecanismo de encadeamento proposto é causal porque utiliza o conceito de sabedoria das multidões com o objetivo de permitir aos próprios usuários confirmarem e estabelecerem relações de causa e efeito entre as notícias pesquisadas. Além disso, ele também é temporal porque seus algoritmos priorizam notícias mais recentes, ou seja, o relacionamento entre um conjunto de notícias tenta partir de uma notícia mais atual para uma notícia mais antiga.

A Fig. 1 apresenta a interface do sistema.

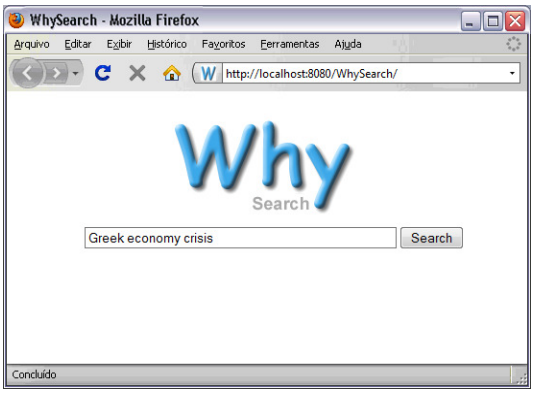

Fig. 1. Interface do sistema

A Fig. 2 ilustra a realização de uma busca que é muito semelhante à maioria dos sistemas existentes atualmente.

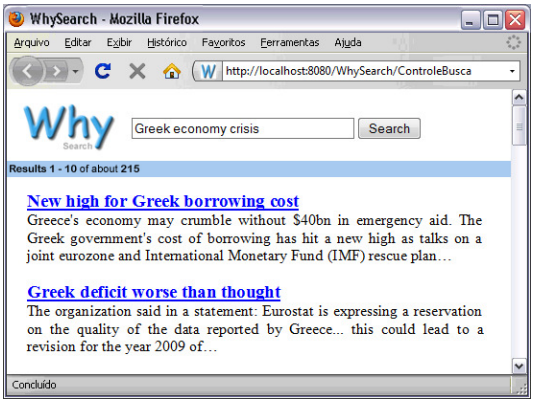

Fig. 2 Busca de notícias

A Fig. 3 mostra o detalhamento de uma notícia. Uma característica particular do *WhySearch* é exibir o conteúdo das notícias sem apontar para páginas externas, como os sistemas atuais costumam fazer, entre eles o Google news[1]. Isso só é possível porque todas as notícias exibidas são previamente processadas e armazenadas no banco de dados do sistema. Dessa forma, deixamos de lidar apenas com *links* e passamos a atuar sobre o conteúdo de cada notícia. Com essa abordagem, conseguimos evitar problemas comuns como a indicação de páginas (*links*) que deixaram de existir nos sites de origem, o que geralmente ocorre com as notícias mais antigas.

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

Fig. 3 Detalhamento de uma notícia

A Fig. 4 ilustra a contextualização de uma notícia realizada pelo mecanismo *WhySearch*, ou seja, o usuário clicou no botão *Why* e a ferramenta exibiu a notícia correlacionada mais próxima em relação a notícia atual.

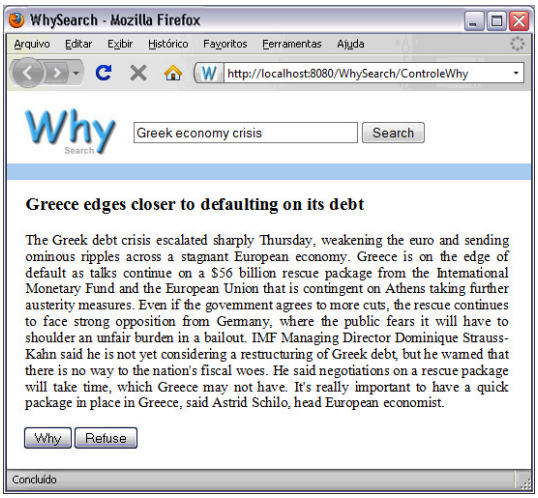

Fig. 4 Contextualização de uma notícia

Se uma determinada seqüência de notícias for esgotada, ou seja, se os botões *Why* ou *Refuse* forem clicados até que não existam mais notícias a serem exibidas, o sistema irá informar essa situação ao usuário. Esse tipo cenário, apesar de possível, é um pouco raro, tendo em vista que para repositórios grandes, a seqüência de notícias será muito longa, ou seja, a profundidade do encadeamento de notícias é diretamente proporcional ao tamanho do repositório.

IV. Arquitetura do Sistema

A Fig. 5 mostra a arquitetura proposta. Ela é formada por duas grandes camadas e uma série de componentes que, atuando em conjunto, compõem todas as funcionalidades do mecanismo de encadeamento de notícias *WhySearch*.

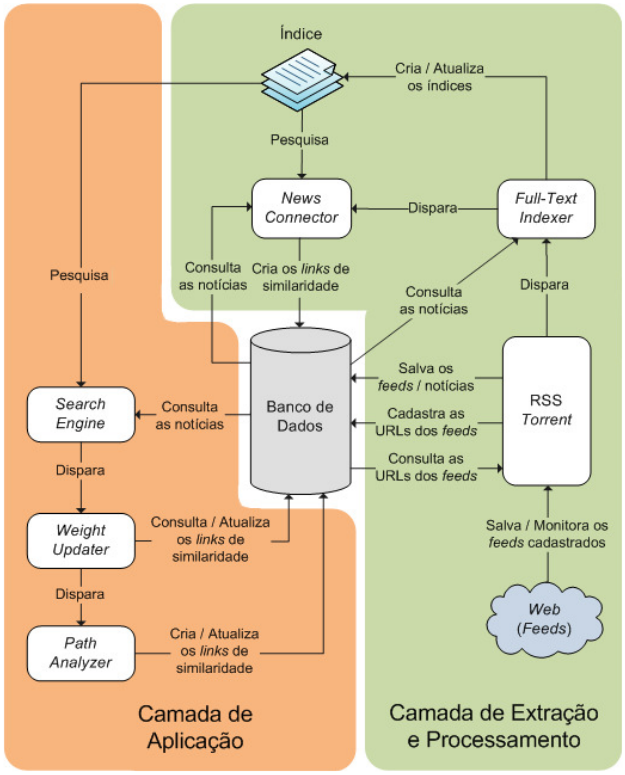

Fig. 5 Arquitetura do sistema

A. *Camada de Extração e Processamento*

A Camada de Extração e Processamento envolve todos os módulos responsáveis por obter (RSS *Torrent*), indexar (*Full-Text Indexer*) e relacionar (*News Connector*) as notícias, ou seja, ela prepara o banco de dados para a Camada de Aplicação. Outra característica de extrema importância dessa camada é o fato dela atuar de forma completamente independente (*background*) da Camada de Aplicação. Como os processos da Camada de Extração e Processamento são contínuos, com essa abordagem, evitamos que o seu processamento impacte a utilização do sistema através da Camada de Aplicação. A seguir é apresentada uma descrição detalhada de cada componente que compõe essa camada.

- **RSS *Torrent*:** responsável por processar e salvar no banco de dados as notícias extraídas dos *feeds* que tiveram seus URLs previamente cadastrados. O principal objetivo desse módulo é formar um grande repositório de notícias. Ele é composto por duas aplicações, o RSS *Manager* e o RSS *Downloader*.

O RSS *Manager* é uma aplicação *web* desenvolvida em Java e executada no servidor de aplicação JBoss[2]. Ela tem como finalidade fornecer uma interface para cadastrar os URLs dos *feeds* que serão monitorados e permitir o acompanhamento do processamento das notícias apontadas por esses *feeds*.

Atuando em conjunto com o RSS *Manager*, temos o RSS *Downloader* que, em linhas gerais, é responsável por obter, processar e armazenar as notícias apontadas pelos *feeds*

[2] http://www.jboss.org

cadastrados, ou seja, de acordo com um intervalo de tempo configurável, essa ferramenta processa a lista de *feeds* cadastrados, adicionando novas notícias ao repositório. Outra característica relevante é o fato dessa aplicação não possuir uma interface gráfica. Ela é executada diretamente pelo *prompt* de comando, e não recebe nenhum tipo de entrada fornecida pelo usuário. Nesse *prompt*, as únicas informações exibidas são a data e hora de início e término do processamento de todos os feeds cadastrados e uma mensagem informando que a atualização foi realizada com sucesso.

● ***Full-Text Indexer*:** indexa as notícias obtidas a cada rodada de execução do RSS *Torrent* através da biblioteca Lucene [7].

● ***News Connector*:** É o módulo responsável por relacionar a maior parte das notícias armazenadas no banco de dados, resultando na criação/atualização do grafo *Why*. Nesse grafo, as notícias correspondem aos nós e os relacionamentos às arestas. A força dos relacionamentos entre as notícias é determinada pelos pesos (similaridade) atribuídos as suas arestas.

O *News Connector* é acionado sempre que o módulo *Full-Text Indexer* termina de criar/atualizar o índice com as novas notícias obtidas através do RSS *Torrent*. Seu funcionamento consiste em relacionar diversos conjuntos de notícias. A Fig. 6 ilustra esses conjuntos.

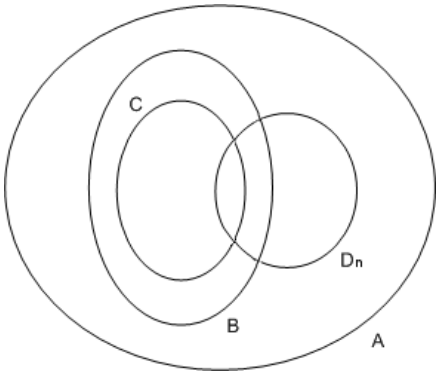

Fig. 6 Conjuntos de notícias

Os conjuntos da Fig. 6 são representados por:

A – Conjunto de todas as notícias do repositório;
B – Subconjunto das notícias recém armazenadas e indexadas;
C – Subconjunto das notícias mais recentes de B;
Dn – Subconjunto das notícias mais similares de A em relação a cada notícia de C, onde n corresponde ao número da notícia de C.

As interseções da Fig. 6 podem ser vazias. O objetivo dessa figura, ao ilustrar essa particularidade, é apenas mostrar que situações desse tipo podem ocorrer.

Para formar os subconjuntos Dn, foram utilizados os índices criados pelo módulo *Full-Text Indexer* e a biblioteca Lucene para buscar as notícias mais similares a cada notícia do subconjunto C. Após a seleção das notícias que serão efetivamente relacionadas (por exemplo, a notícia n e as notícias do conjunto Dn), basta apenas criar os relacionamentos entre elas. Para isso, o coeficiente de similaridade de Jaccard foi utilizado. Esse coeficiente foi escolhido, porque ele permite uma avaliação isolada da similaridade entre duas notícias, diferentemente do tf-idf que leva em consideração todas as notícias do repositório. Para esse problema específico, essa característica coloca o tf-idf em desvantagem tendo em vista que a adição de novas notícias ao repositório faria com que os pesos das correlações já criadas ficassem defasados.

A primeira versão do *News Connector* tentava relacionar todas as notícias do conjunto B com todas as notícias do conjunto A. Essa abordagem mostrou-se inviável devido à grande quantidade de notícias envolvidas, o que fazia com que o tempo de processamento fosse muito longo. Para contornar esse problema, o tamanho desses conjuntos passou a ser limitado através de arquivos de configuração do sistema (*.properties*). Além disso, o tamanho do subconjunto C sempre é configurado com um valor maior do que o tamanho do subconjunto Dn. Essa medida foi adotada com o intuito de priorizar a criação de relacionamentos entre as notícias mais recentes.

Essa alteração resolve o problema da escalabilidade, porém a limitação do tamanho dos conjuntos de notícias faz com que nem todas as notícias do banco de dados sejam relacionadas após uma rodada de execução do módulo *News Connector*. Isso não chega a ser um problema, se considerarmos que o isolamento de algumas notícias é temporário. Para exemplificar esse cenário, basta analisarmos a interseção entre os conjuntos (B-C) e Dn. A princípio, as notícias no conjunto (B-C) não seriam relacionadas, porém a própria regra de formação dos subconjuntos Dn garante que qualquer notícia do repositório pode ser relacionada. Além disso, foi criado o módulo *Path Analyzer* que será explicado em detalhes na seção *B*.

B. Camada de Aplicação

A Camada de Aplicação envolve todos os módulos responsáveis por consultar (*Search Engine*), atualizar (*Weight Updater*) e criar (*Path Analyzer*) relacionamentos entre notícias, ou seja, ela corresponde a parte do sistema que interage diretamente com os usuários. A seguir é apresentada uma descrição detalhada de cada componente que compõe essa camada.

● ***Search Engine*:** esse módulo apresenta a estrutura de um mecanismo de busca tradicional, onde os usuários fornecem um conjunto de palavras de interesse e o sistema retorna um conjunto de documentos ordenados de acordo com sua relevância em relação à consulta. O que diferencia esse módulo dos principais sistemas de busca utilizados atualmente é o fato dele lidar exclusivamente com notícias que são ranqueadas com base apenas em seu conteúdo.

Para implementar esse módulo, foram utilizados os métodos de busca do Lucene e o índice criado/atualizado pelo componente *Full-Text Indexer*.

● ***Weight Updater*:** esse módulo é caracterizado pela navegação e atualização de pesos no grafo *Why*. Quando um usuário seleciona uma notícia retornada pelo componente *Search Engine*, significa que ele está escolhendo um vértice do grafo. À medida que ele clica nos botões *Why* ou *Refuse* o

grafo é percorrido e os pesos das arestas são modificados de acordo com as medidas apresentadas a seguir:

$$why = averageWeight$$

$$refuse = why + \frac{why}{2} = \frac{3 \times why}{2}$$

Onde:

▪ *averageWeight* - média aritmética de todos os pesos armazenados no banco de dados. A média aritmética foi utilizada porque ela é a medida estatística que melhor resume os pesos do grafo *Why*.

▪ *why* – valor adicionado a uma aresta quando o usuário clica no botão *Why*.

▪ *refuse* – valor subtraído de uma aresta quando o usuário clica no botão *Refuse*. Isso equivale a desfazer um *why* equivocado e diminuir o peso da aresta de *why* / 2, ou seja, o *refuse* tem a metade do impacto do *why*. Essa medida foi adotada para evitar que os pesos das arestas diminuam de forma muito rápida.

A Fig. 7 mostra um diagrama de atividades que descreve o funcionamento do módulo *Weight Updater*.

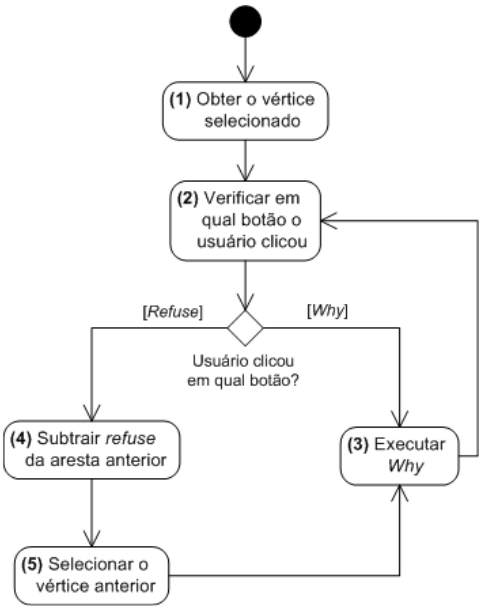

Fig. 7 Diagrama de atividades do *Weight Updater*

Abaixo cada atividade da Fig. 7 é descrita de acordo com a sua numeração correspondente:

1) Obtém o vértice correspondente à notícia selecionada pelo usuário após a busca inicial;

2) Identifica a opção que foi escolhida pelo usuário (*Why* ou *Refuse*);

3) Executa o procedimento *Why*. Essa atividade será expandida e explicada em detalhes na Fig. 8;

4) Subtrai o valor *refuse* da aresta que liga o vértice atual ao vértice anterior;

5) Seleciona (sem mostrar) o vértice anterior.

A Fig. 8 apresenta o diagrama de atividades referente à expansão da atividade "*Execute Why*".

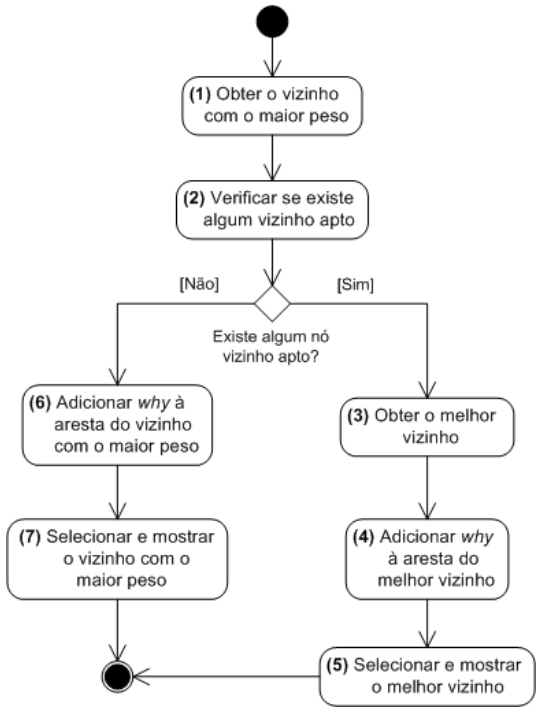

Fig. 8 Expansão da atividade "*Execute Why*"

A seguir cada uma das atividades presentes na Fig. 8 é detalhada.

1) Identifica o vizinho do vértice atual que possui o maior peso;

2) Verifica se existe algum vizinho apto. Um vizinho apto é caracterizado pela seguinte condição:

((vizinho.peso + why) >= vizinhoMaiorPeso.peso) and (vizinho.data <= verticeAtual.data) and (vizinho <> vizinhosVisitados);

De forma muito mais simples, poderíamos utilizar diretamente o vizinho de maior peso, porém como o *WhySearch* é um mecanismo temporal, as notícias são priorizadas em função de sua data de publicação, ou seja, o sistema tenta avançar das notícias mais recentes para as mais antigas.

3) Obtém o melhor vizinho. O melhor vizinho corresponde ao vizinho apto com a maior data;

4) Soma o valor *why* à aresta que liga o vértice atual ao melhor vizinho;

5) Seleciona e mostra a notícia do melhor vizinho para o usuário;

6) Soma o valor *why* à aresta que liga o vértice atual ao vizinho de maior peso;

7) Seleciona e mostra a notícia do vizinho de maior peso para o usuário.

Para determinar os vértices que serão visitados no grafo *Why*, o algoritmo do *WhySearch* considera o peso das arestas e

as datas de publicação das notícias que, neste caso, correspondem aos vértices. Dessa forma, ele tenta obter um equilíbrio entre esses dois valores. Por conta disso, pode-se dizer que o *WhySearch* é um mecanismo de encadeamento de notícias causal e temporal.

● ***Path Analyzer*:** responsável por monitorar a seqüência de notícias confirmadas pelos usuários, ou seja, as notícias onde o usuário clicou no botão *Why*. A cada três notícias confirmadas, um novo registro é criado no banco de dados para a primeira e última notícia da seqüência. Se esse registro já existir, o seu contador é incrementado. Quando esse contador atinge um valor limite (configurável através de um arquivo *.properties*), ele é zerado e um relacionamento direto envolvendo essas notícias é criado com o valor *why*. Caso essa correlação já exista, o valor *why* é adicionado ao peso dessa aresta.

Com esse módulo, o fato de não estarmos relacionando todas as notícias do repositório (*News Connector*) é compensado através da criação de ligações indiretas entre nós de caminhos freqüentes.

C. Banco de Dados

Uma característica comum a todos os módulos do *WhySearch* é atuar sobre o mesmo banco de dados. O modelo desse banco de dados é o resultado de um esforço conjunto de três diferentes iniciativas: o *WhySearch*, o Dado Autonômico [8] e a ferramenta *Feed Organizer* [9][10]. A idéia é combinar essas propostas e fornecer um ambiente integrado para lidar com notícias web.

V. Resultados Esperados

Com o objetivo de realizar uma primeira avaliação do mecanismo de encadeamento de notícias *WhySearch*, as respostas fornecidas por um grupo de pessoas foram comparadas com os resultados obtidos pela execução dessa ferramenta em um ambiente controlado.

Nessa avaliação inicial, participaram 10 estudantes de graduação e foram avaliadas 5 notícias. A Tabela I mostra as notícias que foram selecionadas.

TABELA I

Notícias Selecionadas para o Experimento

News Id	Title	Description
1	In Brazil, Paying Farmers to Let the Trees Stand	In an effort to prevent farmers from cutting down rain forest, environmental groups are offering money (August 22, 2009).
2	Brazilian Rancher Convicted in Death of Nun	A jury convicted a Brazilian rancher early Saturday of orchestrating the murder of U.S. nun and Amazon defender Dorothy Stang (May 1, 2010).
3	Brazil Orders New Trials in Nun's Death	Vitalmiro Bastos de Moura, a rancher, had previously been acquitted of orchestrating the murder of Sister Dorothy Stang, a 73-year-old rain forest activist (April 8, 2009).
4	Forests and the Planet	With the rain forests shrinking and the planet warming up, it is crucial that the right domestic and global incentives are put in place (May 29, 2009).
5	Forest Plan in Brazil Bears the Traces of an Activist's Vision	This month, the Brazilian government introduced ambitious targets for reducing deforestation and carbon dioxide emissions (December 22, 2008).

O passo seguinte deste experimento foi a criação de um repositório específico (ambiente controlado) que contivesse apenas as notícias apresentadas na Tabela I. Em seguida, os módulos *Full-Text Indexer* e *News Connector* foram executados para que essas notícias fossem devidamente indexadas e relacionadas entre si de acordo suas similaridades. Como nesse caso específico o repositório é muito pequeno, todas as suas notícias foram relacionadas. A Fig. 9 ilustra o grafo *Why* que foi gerado.

Considerando a Fig. 9, definimos a notícia de origem, ou seja, o ponto de partida para as análises que foram realizadas sobre o grafo *Why*. Para esse experimento foi escolhida a notícia 2 (mais recente do conjunto). Dessa forma, cada participante recebeu o texto completo das cinco notícias e foi orientado a definir uma ordenação entre elas partindo da notícia 2. Foi sugerido que essa ordenação fosse determinada de forma a criar uma idéia cronológica (da notícia mais recente para a mais antiga) de causa e efeito.

O problema consiste em ordenar as notícias com o objetivo de facilitar o entendimento da história como um todo. É importante ressaltar que as datas em que as notícias foram publicadas não foram informadas aos participantes do experimento para evitar que eles fossem influenciados por essa informação. A Tabela II apresenta as ordenações definidas pelos participantes.

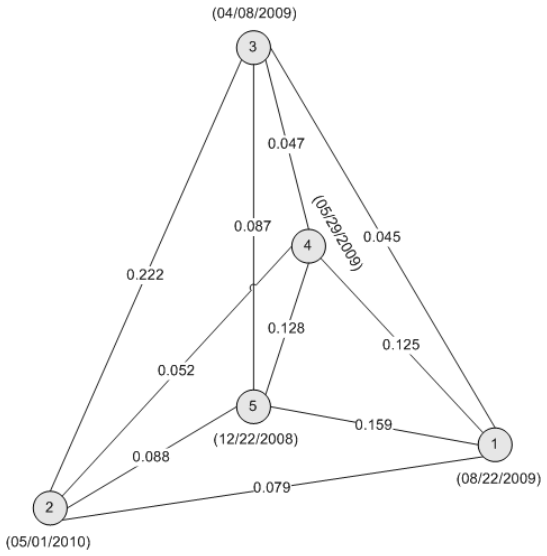

Fig. 9 Grafo *Why*

TABELA II

Ordenações dos Participantes do Experimento

Participant Id	News Sequence
1	2, 4, 5, 1, 3
2	2, 3, 5, 4, 1
3	2, 3, 4, 1, 5
4	2, 1, 4, 3, 5

5	2, 3, 5, 1, 4
6	2, 1, 3, 5, 4
7	2, 4, 1, 5, 3
8	2, 3, 5, 4, 1
9	2, 5, 1, 4, 3
10	2, 3, 5, 4, 1

A partir das ordenações definidas pelos participantes na Tabela II, é possível determinar a seqüência de notícias ótima: **2, 3, 5, 4, 1**. Essa seqüência foi criada com base nas notícias mais escolhidas para cada posição. Por exemplo, na posição 2, a notícia 3 foi a mais escolhida entre as outras (5 vezes).

Como já temos a seqüência de notícias mais escolhida pelos usuários, basta obtermos a seqüência gerada pela execução da ferramenta. Para isso, selecionamos a notícia 2 no *WhySearch* e clicamos no botão *Why* até que o sistema informe que não existem mais notícias a serem apresentadas. O *WhySearch* retornou a seguinte seqüência: **2, 3, 5, 1, 4.**

Comparando a seqüência ótima com o resultado gerado pelo *WhySearch*, verificamos que apenas as notícias 1 e 4 aparecem em posições distintas. Isso já caracteriza um bom resultado, porém o conceito de sabedoria das multidões ainda não foi considerado. Para que esse conceito fosse avaliado, executamos a ferramenta direcionando-a a retornar as seqüências definidas pelos 10 usuários, ou seja, clicamos no botão *Why* se a ferramenta retornou a notícia esperada pelo usuário, caso contrário, clicamos no botão *Refuse*. Esse procedimento foi realizado até que as seqüências coincidissem. A ferramenta foi executada de acordo com o identificador de cada participante. A primeira execução foi realizada para a seqüência de notícias do participante 1 e a última para a seqüência do participante 10. A Fig. 10 mostra como os pesos do grafo *Why* foram modificados pelo conceito de sabedoria das multidões.

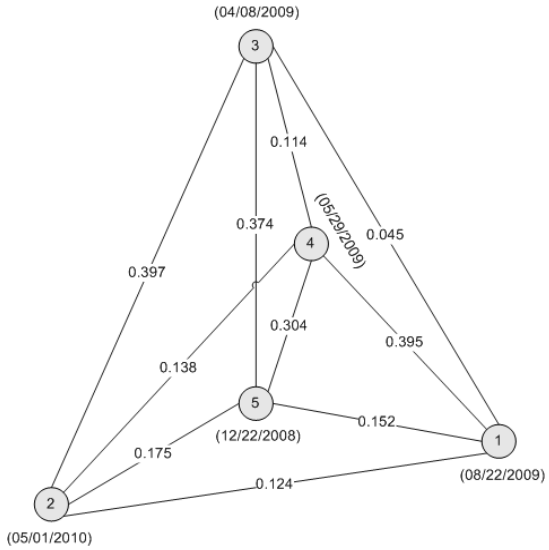

Fig. 10 Novo grafo *Why*

A partir do novo grafo *Why* mostrado na Fig. 10, executamos o mesmo procedimento descrito anteriormente para obter a seqüência de notícias gerada pela ferramenta, ou seja, selecionamos a notícia 2 no *WhySearch* e clicamos no botão *Why* até que não existissem mais notícias a serem apresentadas. A seguinte seqüência foi obtida: **2, 3, 5, 4, 1**. Como podemos notar, a seqüência obtida é idêntica a seqüência ótima definida pelos participantes do experimento, ou seja, um excelente resultado. Além disso, conseguimos mostrar a influência positiva da sabedoria das multidões nas relações de causa e efeito.

VI. Conclusão

Neste trabalho, foi apresentada uma nova abordagem sobre o encadeamento de notícias *web*, onde tentamos inferir relações temporais de causa e efeito levando em conta diversos fatores como a similaridade das notícias, suas datas de publicação e a sabedoria das multidões. Para isso, mostramos em detalhes todos os módulos que compõem o mecanismo de encadeamento de notícias *WhySearch*.

Com o objetivo de avaliar essa ferramenta, foi realizado um experimento que apresentou resultados extremamente satisfatórios, tendo em vista que ao considerarmos a sabedoria das multidões, a seqüência de notícias fornecida pela ferramenta coincidiu com a seqüência preferida pelos participantes do experimento.

Como trabalhos futuros, esperamos melhorar a interface gráfica da aplicação com a apresentação do histórico das notícias exibidas pela ferramenta através de uma árvore, ou seja, à medida que o usuário visualiza novas notícias através dos botões *Why* e *Refuse*, novos nós são adicionados a essa árvore. Além disso, o usuário ficaria livre para visualizar qualquer notícia que já tenha sido apresentada.

References

[1] J. Surowiecki, "The Wisdom of Crowds", Anchor, ISBN: 0385721706, 2005.

[2] C. D. Manning, P. Raghavan, H. Schütze, "Introduction to Information Retrieval", 1º ed., Cambridge University Press, ISBN: 0521865719, 2008.

[3] R. Baeza-Yates, B. Ribeiro-Neto, "Modern Information Retrieval", 1º ed., Addison Wesley, ISBN: 020139829X, 1999.

[4] J. Allan, "Topic Detection and Tracking: Event-based Information Organization", The Kluwer International Series on Information Retrieval, Volume 12, 1º ed., Springer, ISBN: 0792376641, 2002.

[5] D. Winer, "RSS 2.0 Specification". Disponível em: http://cyber.law.harvard.edu/rss/rss.html.

[6] M. F. Porter, "Readings in information retrieval", Readings in information retrieval, Morgan Kaufmann Publishers Inc, 1997.

[7] Apache Software Foundation, "Apache Lucene". Disponível em: http://lucene.apache.org.

[8] M. Silva, J. M. Souza and J. Oliveira, "Autonomic Data: Use of Meta Information to Evaluate, Classify and Reallocate Data in a Distributed Environment", 2008.

[9] W .A. Pinheiro, M. Silva, R. Barros, "Autonomic collaborative RSS: An implementation of autonomic data using data killing patterns", 2009.

[10] W. Pinheiro, M. Silva, T. Rodrigues, "Discarding Similar Data with Autonomic Data Killing Framework Based on High-Level Petri Net Rules: An RSS Implementation", ICAS, 2010.