UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO
INF01124 – CLASSIFICAÇÃO E PESQUISA DE DADOS - TURMA B

JOÃO PEDRO GUBERT - 273166
PEDRO SCHANZER OLIVEIRA - 208612
JOSUÉ FILIPE KEGLEVICH DE BUZIN - 166409

Grupo Mapeamento Politico

Orientador: Profa. Karin Becker

Porto Alegre
2016

SUMÁRIO

1 PROBLEMA

A expansão do uso do Facebook levou seus usuários a serem expostos diariamente a diversas correntes ideológicas. Páginas, Figuras Públicas e Partidos Políticos aproveitaram dessa oportunidade para disseminarem as suas informações. O problema é que o próprio Facebook tem um algoritmo de ordenamento para que o usuário receba em sua timeline apenas informação relevante. Isso é feito a partir das suas curtidas e compartilhamentos. As páginas mais acessadas tornam-se chaves para que a pessoa seja ligada a novas páginas sobre o mesmo tema. Isso fez elas ficarem presas à chamada "bolha ideológica". O nosso aplicativo tenta de maneira simples, gerar uma reflexão sobre o grau de inserção do usuário nesta bolha.

O aplicativo lê o perfil de um usuário do Facebook e orienta a sua posição politica, a partir das informações coletadas, ou seja, as suas curtidas e compartilhamentos. Para essa classificação foram utilizadas as postagens de páginas de políticos, figuras públicas, páginas ideológicas e portais de notícias que o usuário curte e compartilha em sua timeline. Estes dados são extraídos do arquivo gerado pelo Facebook chamado de timeline.htm, formatados e comparados com um banco de dados predefinido das principais páginas relacionadas à politica com a finalidade de gerar a definição política do usuário. Este banco de dados está separado em dois arquivos txt, um para as páginas de direita e outro para as páginas de esquerda. A cada comparação positiva deste arquivo txt com o arquivo extraído da timeline do usuário é criada uma estrutura com o nome do usuário, nome da página, o post relacionado, um contador de frequência (que será incrementado se ele interagir mais vezes com a página e serve de chave para ordenar o HeapSort) e o alinhamento político da página. Após toda timeline ser analisada é criado um array de estruturas que serão ordenadas através de um HeapSort pela frequencia de cada estrutura. Foi utilizado o algoritmo de ordenamento HeapSort, pela facilidade em descobrir a pagina mais curtida, uma vez que o Array é ordenado, a pagina com maior frequencia se encontra na raiz. O Array de estruturas é salvo em disco, juntamente com um arquivo de índices, que é utilizado para tornar as pesquisas mais eficientes. Esse arquivo de índices contém um índice por nome de pagina que o usuário interagiu.

Deste modo, o usuário tem acesso a uma consulta de suas curtidas e compartilhadas de forma decrescente ou crescente de frequência, podendo até pesquisar por páginas ou por dias específicos, através de um Hashing. Após ter acesso a essas informações cabe ao usuário a reflexão se o seu grau de posição política não é muito extremo ou limitado, se não

predominam a presença de páginas parciais no seu dia a dia e se ele não deveria a ter acesso a outras páginas do nosso banco de dados para expandir o seu paradigma político.

O grupo utilizou a linguagem de programação Java para implementar o trabalho, pois ela é ótima para comparação entre strings gerados na extração do texto e as páginas do banco de dados de esquerda e direita. Os algoritmos de HeapSort e Hashing podem ser facilmente implementados, necessitando apenas da realização prévia de um parser para que o arquivo extraído da timeline possa ser convertido em strings ordenáveis.

2 IMPLEMENTAÇÃO

Estrutura de dados: ArrayList

ArrayList é um array extensível em que cada elemento do array é a estrutura abaixo:

public class Estrutura implements Serializable:
private static final long serialVersionUID = 1L;- para serialização da estrutura de dados
public String nome;- nome da pessoa
public String comunidade;- pagina relativa ao elemento da estrutura de dados
public String post;- string de posts relativos à mesma pagina
public int freq;- Contador para o numero de posts relativos à mesma pagina
public String posicao;- se a pagina está mais alinhada à esquerda ou à direita

O ArrayList é usado para armazenar na memória os dados da timeline. Depois de preenchido, o ArrayList é ordenado por frequência através do Heapsort e armazenado em um arquivo binário através do empacotamento de estruturas(serialização). Uma lista somente das paginas é armazenada num arquivo binário que servirá como indexador para a pesquisa dos dados posteriormente.

O HeapSort consiste em um Array que pode ser visto na forma de uma árvore binária completa, que mantem uma ordenação da raiz até as folhas. O HeapSort usa 2 formas de ordenação para a árvore: Max-Heap e Min-Heap.

No Max-Heap o pai sempre é um valor maior do que o valor dos filhos, sendo que o array é ordenado de forma decrescente. E no Min-Heap a ordem é crescente.

O HeapSort mostrou-se muito útil em nosso trabalho, pela facilidade de exibir a página que o usuário mais interagiu. Foi justamente essa facilidade que motivou a escolha do ordenamento em um Heap pela frequência das paginas.

O HashMap é uma classe de Hashing extensível que usa um índice(lista de strings ou numeros) para acessar os dados gravados na estrutura geral do aplicativo, seu diferencial é a velocidade na busca de dados.

A biblioteca de terceiros jsoup foi utilizada para tirar o texto formatado em htm e salvar em um txt no formato de texto comum para facilitar a comparação de frases no formato de strings.

A maior dificuldade no uso da estrutura para a solução do problema está na forma de gravação dessa estrutura relativamente complexa em um arquivo binário. A necessidade da serialização dos objetos dificulta a pesquisa posterior dos dados no arquivo sem ter que deserializar os dados novamente, ocupando muito espaço desnecessário na memória. Nesse

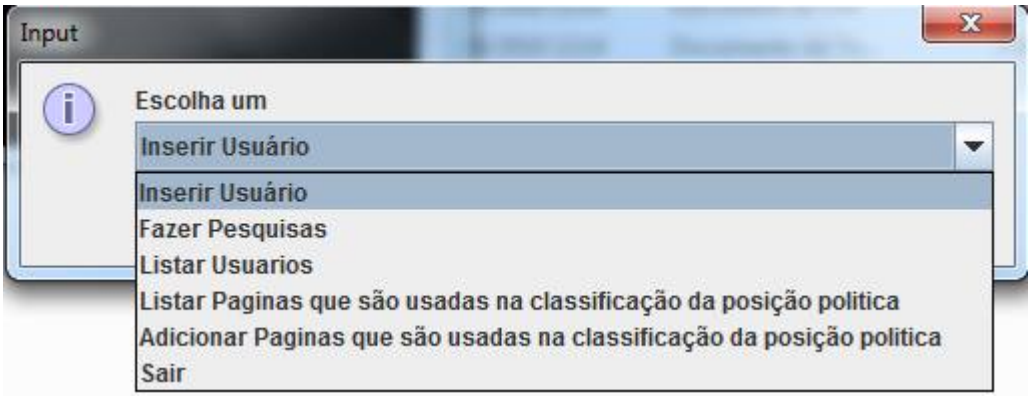
aspecto a escolha da linguagem Java trouxe problemas, pois a mesma não prevê o uso de uma função para pesquisar esses dados empacotados diretamente no arquivo de forma simples. Para isso seria necessário saber quantos bytes cada elemento da estrutura ocupa no arquivo e guardar sua posição no índice, o que só é possível saber se o arquivo for randômico, criado a partir de uma classe especifica que não suporta a serialização de estruturas com tamanho de bytes variável.

3 GUIA DE USO

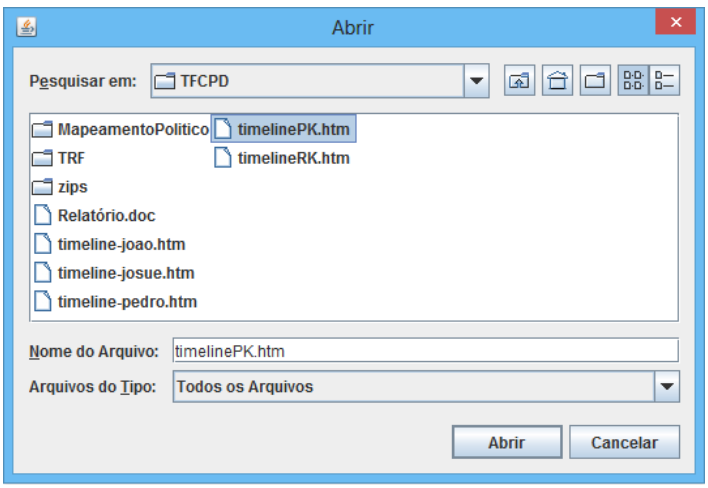
Para utilizar o aplicativo Mapeamento Político o usuário deve fornecer um arquivo que pode ser facilmente extraído do seu facebook chamado “timeline.htm”. Para adquirir o arquivo é necessário fazer o login no Facebook, entrar nas configurações do usuário e clicar em ‘Baixe uma cópia dos seus dados do Facebook.’. Após a confirmação, um link será enviado para o email cadastrado na conta do Facebook, no link que estiver no email o usuário entra no Facebook e precisa confirmar a sua senha para fazer o download de um arquivo zip. Dentro deste zip o arquivo que será usado no programa está em ‘\html\timeline.htm’.

Este arquivo será utilizado para ordenar as informações do usuário. Com este ordenamento temos o objetivo que gerar uma reflexão no usuário, tornando o mais consciente de suas atividades no Facebook ao mostrá-lo maneiras de expandir o seu paradigma político.

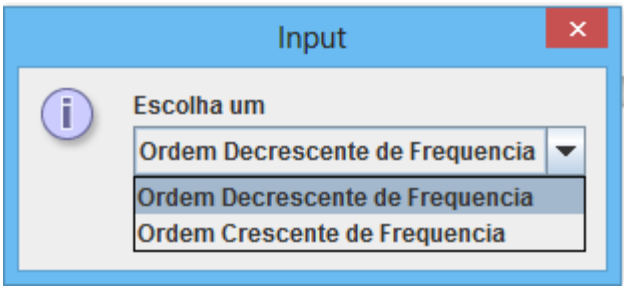
Ao iniciar o programa é possível escolher opções em um menu. São elas:



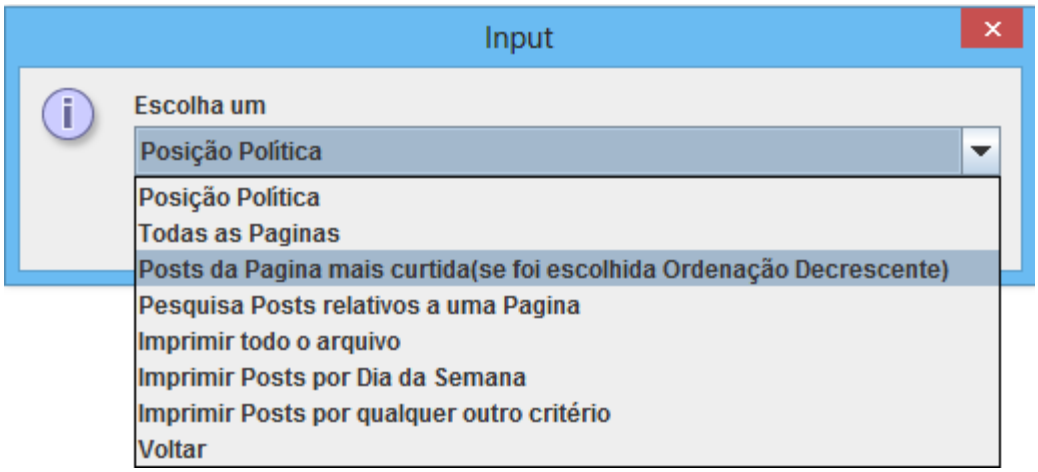
Ao pressionar “Inserir Usuário” será requisitado que ele procure em seu sistema pelo seu arquivo “timeline.htm” desejado. Abaixo temos exemplos de arquivos aceitáveis.



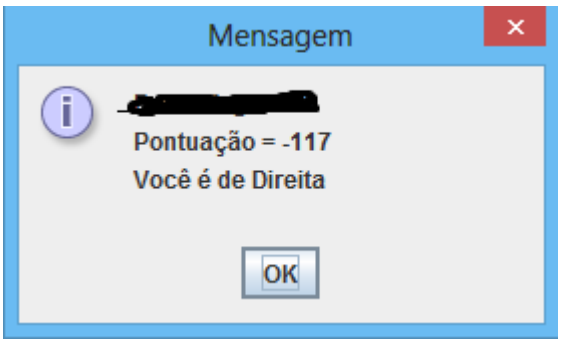
Após o usuário será induzido a escolher se ordena esses dados por ordem decrescente ou crescente de frequência.



Então será exibido um menu com diversas opções. São elas:

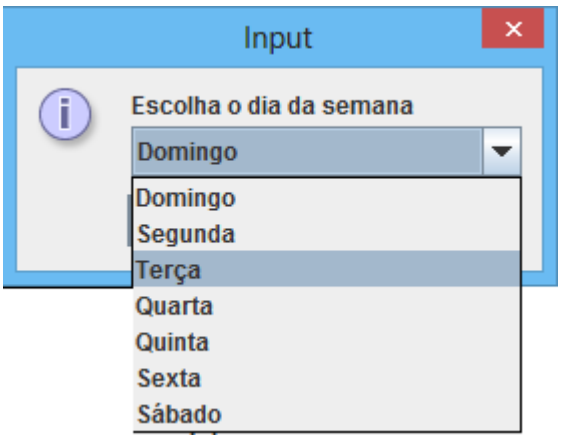


Mostraremos dois exemplos de escolha nesse menu. Primeiramente a sua função principal, Posição Política.



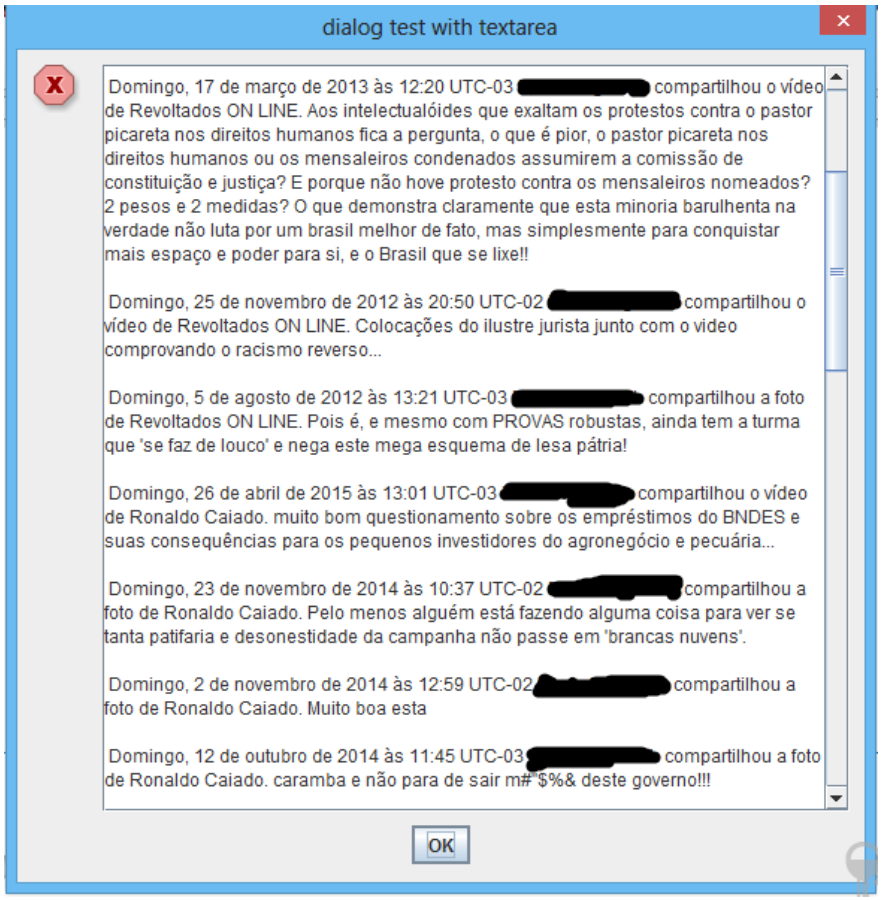
Neste exemplo vemos que a pessoa teve pontuação de -117. O que significa que ele está com uma visão de extrema direita. Esse é um bom exemplo de Bolha Ideológica.

Outra opção do menu é a de mostrar as atividades de um dia específico da semana.

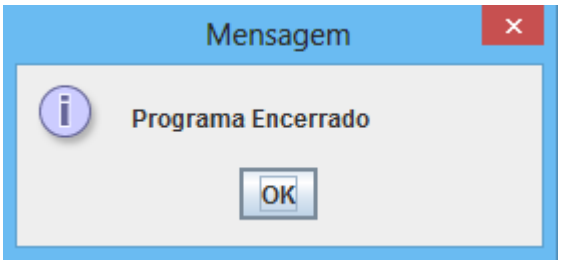


Ao observarmos, por exemplo, suas atividades no Domingo, podemos ver que a pessoa realmente está presa a uma corrente de conhecimento onde só aparecem sugestões de direita.

Esta é a caixa de dialogo que aparece com a consulta:



O usuário tem também a opção de voltar ao primeiro menu e encerrar o programa ao pressionar sair. Com isso deve aparecer a mensagem “Programa Encerrado”.



4 CONTRIBUIÇÃO

Hoje o Facebook faz parte da rotina de muitas pessoas e é alarmante a parcialidade com a qual os dados são propagados nele. Muitos boatos viram manchete e as pessoas ficam presas em correntes de informação chamadas bolhas ideológicas. O algoritmo que o Facebook usa para mostrar dicas de informações e páginas aos usuários está baseado na frequência com que a pessoa acessa determinada informação. Assim o indivíduo é progressivamente induzido a ter acesso somente a corrente ideológica que apetece os seus desejos. Pensando nesse problema o aplicativo tenta mostrar ao usuário o paradigma político no qual ele está inserido, para confrontá-lo com a sua realidade. O usuário pode ter acesso aos arquivos que determinaram esse ordenamento, podendo modificá-los e contribuir com a pesquisa.

O primeiro caso de reuso de código ocorreu no momento da extração dos dados do arquivo htm(função public void LeituraHTM(File arq)), foi utilizado um código do site StackOverflow(disponibilizado pelo usuário 'RobMen') que usa a biblioteca de terceiros jsoup para tirar o texto formatado em htm e salvar em um txt no formato de texto comum para facilitar a comparação de frases no formato de strings. O código foi utilizado sem modificações significativas.

A classe Heapsort foi reutilizada a partir de um algoritmo que está no site GitHub(autor WWaldo), o algoritmo foi modificado para ordenar a ArrayList de estruturas de dados e não mais somente uma ArrayList de números, tanto na ordem crescente quanto decrescente de frequência, conforme a opção do usuário.

O último código reutilizado foi o que está na classe SalvaD e vem do fórum GUJ(autor Pablosaraiva), que é um exemplo de como utilizar a serialização para salvar uma estrutura de dados num arquivo binário sequencial, sendo que a leitura posterior é feita usando um índice de números que é tirado de um arquivo binário. O código sofreu muitas modificações e está particionado entre várias outras funções conforme a necessidade, sendo que a parte em que ocorre a escrita do que está na memória para os arquivos binários está separada da leitura desses mesmos arquivos para permitir a inserção de dados de vários usuários diferentes. Para essa leitura, foi necessário desempacotar os dados do arquivo em uma estrutura qualquer na memória, que no código do aplicativo pode ser tanto um HashMap quanto uma estrutura comum. A estrutura comum foi usada nos casos em que não é necessário passar todos os dados do arquivo para a memória. Quanto o usuário necessita visualizar todo ou a maior parte do arquivo, se usa um HashMap que acessa os dados através de um índice de páginas, similar ao índice de números do código original, que é incorporado ao HashMap sendo as chaves

específicas para a pesquisa. O valor associado a essas chaves específicas é a estrutura de dados que estava empacotada no arquivo binário, sendo que tanto as chaves quanto seu valor associado já foram ordenados previamente por Heapsort.

5 CONSIDERAÇÕES FINAIS

O programa tinha como objetivo gerar uma reflexão no usuário sobre a extensão do seu paradigma político. Ao mostrar para ele suas fontes e uma leitura sobre as suas tendências políticas encontramos uma maneira de posicionar ele em um "vetor", no qual quanto mais comparações positivas ele tivesse em relação ao banco de dados da esquerda o seu vetor iria em direção aos números positivos, quanto mais comparações com o banco de direita, mais negativo ficava o seu vetor. Números finais extremos (muito positivos ou muito negativos) demonstram que o usuário está inserido em uma bolha ideológica. A maior limitação frente a isso é que hoje em dia o paradigma político brasileiro não esta dividido em direita e esquerda, existem esquerdas fascistas mascaradas com ideais de que todos devem ser um e direitas liberais com tendências que procuram diminuir a desigualdade social e buscar inclusão de camadas mais pobres. Para que o nosso algoritmo fosse mais efetivo deveríamos buscar a associação de palavras nos posts das páginas e usuários. Deste modo, se o usuário falasse sobre "saúde pública" ou sobre "educação privada" o algoritmo lhe diria com mais precisão sua visão política e seus temas de interesse. Esta solução será desenvolvida com mais tempo até a metade de Julho e será utilizada em um Aplicativo do coletivo Minha Porto Alegre, onde os usuários serão informados das suas tendências ideológicas e poderão perguntar sobre esses temas para políticos que concorrerão na próxima eleição. Esse outro aplicativo já foi utilizado nas eleições do Rio de Janeiro em 2012, mas não tinha essa opção de interação com o Facebook e análise de temas para dar dicas ao usuário. Com os avanços que a nossa versão Beta trouxe, junto com outros melhoramentos que serão desenvolvidos, os usuários vão ter acesso às propostas e perfis desses candidatos na palma da mão, aumentando a sua consciência política e diminuindo a alienação que o modelo de propaganda eleitoral atingiu atualmente.

O primeiro algoritmo utilizado foi o de parser, que usa a biblioteca jsoup. Com ele foi possível limpar do arquivo original os dados que não nos interessavam. Assim, ficamos só com os textos que utilizaríamos para fazer o escaneamento da visão política. Para a análise das frequências de cada página nós utilizamos o algoritmo de Heapsort que foi muito efetivo, pois ordenou os strings do array pela frequência como facilidade. O Hashing retirou os dados da estrutura no arquivo binário para serem lidos com menos uso da memória. Por causa da serialização feita pelo hashing não foi possível ler diretamente do disco dos dados, uma limitação da linguagem Java. Porém mesmo assim o Hashing foi efetivo por que só o utilizamos quando foi necessário ler todo arquivo binário.

As maiores dificuldades do desenvolvimento foram exatamente com o Hashing para a implementação do índice no arquivo binário, por que a linguagem java tem restrições sobre salvar em arquivo binário uma estrutura de dados de tamanho variável. É possível fazê-lo, mas muito difícil. A explicação que encontramos foi a de que são operações de baixo nível, usando uma linguagem de alto nível (Java).

Tendo em vista todas as dificuldades apresentadas e as oportunidades que surgiram com o desenvolvimento do aplicativo, estamos satisfeitos com os objetivos que alcançamos. Agora será possível aperfeiçoar essa aplicação para que seja útil nas próximas eleições, um objetivo que não estava previsto, mas nos gerou contato com um grupo de desenvolvedores de nível nacional.

REFERÊNCIAS

GITHUB - WWALDO. **Sorting-Algorithms**. Disponível em: <https://github.com/WWaldo/Sorting-Algorithms >. Acesso em: 18 de junho de 2016.

GUJ - PABLOSARAIVA. **Arquivos e Indices - Estrutura de Dados**. Disponível em: <http://www.guj.com.br/t/arquivos-e-indices-estrutura-de-dados/117462/9>. Acesso em: 18 de junho de 2016.

STACKOVERFLOW - ROBMEN. **Remove HTML tags from a String**. Disponível em: <http://stackoverflow.com/questions/240546/remove-html-tags-from-a-string>. Acesso em: 18 de junho de 2016.