HappyHillary – PP-2Q/2016

Lenin Cristi
github.com/lnncrs
RA-21028214
lenin.cristi@aluno.ufabc.edu.br

Gustavo Murayama
github.com/gmurayama
RA-21028214
g.murayama@aluno.ufabc.edu.br

ABSTRACT
Functional programming driven data analysis project about Hillary Clinton's mail data leaked in 2015. This project was created for 2Q/2016 "Programming Paradigms" class on UFABC (Federal University of ABC/BR) of teacher Diego Martins and has two main goals: Be done primarily in Scheme language and implement solutions in concurrency.

Projeto de análise de dados dos e-mails vazados da candidata à presidência dos EUA, Hillary Clinton, em 2015. Este projeto foi feito no âmbito do curso 2Q/2016 "Paradigmas de programação" da UFABC (Universidade Federal do ABC) ministrado pelo Prof. Diego Martins e tem duas metas: Ser feito prioritariamente em linguagem Scheme e implementar soluções de concorrência.

CCS Concepts
•**Theory of computation → Design and analysis of algorithms → Data structures design and analysis → Pattern matching**

Keywords
Data analysis; Scheme; Hillary Clinton; Leak.

1. INTRODUÇÃO
Em 22 de agosto de 2016, um juiz federal dos EUA ordenou ao Departamento de Estado Americano que planejasse a liberação de aproximadamente 15000 descobertos pelo FBI durante sua investigação contra a Sra. Hillary Clinton sobre a utilização de um servidor de email privado para assuntos de Estado no período em que ocupava o cargo de Secretária de Estado. Em julho, o FBI recomendou que nenhuma ação deveria ser tomada contra a Sra. Clinton e afirmou que não haviam evidências de que eles teriam sido "intencionalmente excluídos", mas pontuou sua conduta com relação a correspondência eletrônica no Departamento de Estado como "extremamente descuidada".

Foram liberados um total de 30000 emails, sendo 8 conversas com informação "top secret", 36 com informação "secret", 8 "confidential" e mais 2000 foram classificados tardiamente como "confidential".Mais 14900 mensagens ainda não foram liberadas e incluem 1 conversa marcada "secret" e 2 "confidential".

Muitos grupos dentro e fora dos EUA se reuniram em torno destes dados para analisá-los das mais diversas formas e ângulos, o site WikiLeaks por exemplo gerou uma base de dados pesquisável com eles, o site de entusiastas de análise de dados Kaggle tem uma base destes dados com diversas pesquisas em Python e R.

Quando da produção do projeto, achamos natural pela disponibilidade de funções de alta ordem na linguagem do projeto fazer uma análise de dados, e como no momento de sua produção este vazamento se tornou uma peça-chave na campanha presidencial dos EUA uma vez que a Sra. Hillary conseguiu a indicação do Comitê Democrata a corrida presidencial, entendemos que seria interessante de alguma forma analisar seu conteúdo.

2. FONTES DE DADOS
Tivemos duas fontes de dados usadas como partida, mas somente uma de fato pode ser consumida: A base de dados cedida no laboratório de BigData da Semana do CMCC, evento ocorrido em 2016 na UFABC, e a base de dados da Kaggle.

Usamos a base de dados da Kaggle pois ela continha a integra das mensagens em um arquivo no formato "SQLite", que se mostrou mais versátil no momento da exportação em formato específico.

Usamos como ponte o SGBD "SQL Server 2016" para exportar os dados da base "SQLite", processo esse que pode ser reproduzido com os passos contidos na pasta "\data" numerados de "01" a "05", tivemos diversas dificuldades nessa fase:

- A codificação, que versões anteriores do SQL Server exportavam em formato desconhecido, mas apartir da versão 2014 SP2 permitiu ser em UTF-8, o que também otimizou o processamento dos arquivos, que diminuíram sensivelmente de tamanho;
- As quebras de linha, que apesar de trivial, exigiram implementações de código não tão simples, pois o ambiente de desenvolvimento teve de ser heterogêneo, visto que o SQL Server para Linux que está em testes ainda não foi lançado para o público geral.

Foi adotada a convenção de agrupar as mensagens em arquivos de texto representando um mês com nome "leak_mail_yyyymm" para arquivos com mensagens agrupadas por mês de vazamento e "sent_mail_yyyymm" para arquivos com mensagens agrupadas por mês de envio (os envios ocorreram entre 2008 e 2014, os vazamentos ocorreram em 2015).

É importante notar que os arquivos de "enviadas" têm relação com a história em curso, é possível gerar tags onde se notam conflitos na líbia, a queda do governo hondurenho, etc. Os arquivos por mês de "vazamento" podem ser relacionados com pesquisas de opinião e textos da imprensa com sua repercussão.

Os arquivos marcados "mail" contém o corpo das mensagens, os arquivos marcados "subj" contém os assuntos somente, essa divisão se tornou necessária pois em determinados arquivos e dependendo do hardware a disposição as funções utilizadas para agrupar e contar palavras estavam demorando demais e atrapalhando os testes constantes do desenvolvimento das funções.

3. DEFINIÇÃO DO PROBLEMA
A dificuldade de analisar essa quantidade de mensagens inicia na pesquisa, o que procurar exatamente? Palavras-chave como "Khadafi, Libya ou Operations" podem nos levar a mensages específicas, como é possível na ferramenta do WikiLeaks, mas uma tagcloud com esses dados possibilitaria entender mês a mês, seja no envio ou no vazamento, quais eram os assuntos mais importantes transitando no Departamento de Estado Americano.

Decidimos implementar uma solução nesse sentido com gráficos lineares de apoio com totais somente, e procuramos tornar as funções o mais portável o possível na medida da praticidade de uso, para que fosse possível reaproveitá-las no futuro com novos conjuntos de dados relacionados, ou mesmo reutilizá-las em conjuntos diversos.

4. ESTRUTURA

A estrutura de diretórios foi pensada essencialmente para separar os arquivos de dados (tanto fontes como exportados), dos locais de código e da pasta de relatórios, que depois de populada pelas rotinas, é portátil ou seja, dentro dela ela leva os arquivos de apresentação (HTML5, CSS) e de dados já processados (JS), pode ser copiada livremente para fora do diretório e executada em qualquer lugar, já que toda saída produzida é compeltamente compatível com padrões do W3C. Descrição das estruturas principais:

- /data
    - leak - arquivos de assunto e corpo de e-mail divididos por mês de vazamento
    - sent - arquivos de assunto e corpo de e-mail divididos por mês de envio
    - sources
        - cmcc - arquivos de e-mail utilizados no laboratório de dados da semana do cmcc da UFABC. Estes arquivos não puderam ser usados pelo formato.
    - kaggle - arquivos de dados recuperados do site Kaggle, o banco de dados sqlite desse conjunto foi utilizado como fonte primária do projeto
    - test - pasta temporária para despejo de arquivos e testes de codepage
- /docs - docmentação
- /lib - pasta com uma biblioteca “utils.scm”, usada como ponto central de funções utilitárias
- /modules
    - io
        - funções de entrada / saída (as em uso aceitam uma “porta” como parâmetro)
    - workers
        - funções de processamento de dados
    - webcrawlers
        - funções de extração de texto de páginas da web (em pyhton). Chegou a ser construído o módulo de extração de texto das matérias do site HuffingtonPost, mas não chegou a ser implementado, apesar de estar funcional.
    - tagcloud.scm – função de despejo de dados das tag clouds
    - linear.scm – função de despejo de dados dos gráficos lineares
- /report - pasta portátil de saída de dados
    - /data - saida de dados processados em formato JS
    - /index.html – ponto de entrada do site

5. MÓDULOS

Os módulos foram pensados para serem feito de funções puras sempre que possível e que pudessem ser utilizadas em “cascata” sem prejuízo. Também demos atenção a funções que teriam de ser repetidas numa lista de arquivos de dados para serem preparadas para recursão.

Módulo IO, que contém dois conjuntos, reader e writer.

A função reader se mostrou um desafio por determinados motivos que variaram de limitações de memória ao que chegamos cogitar ser um limite numérico de parâmetros na chamada de uma função para a linguagem, e teve de ser reescrita e ajustada várias vezes. Apesar de ser possível substituí-la num primeiro momento por funções mais simples como a “read-all” que lê um arquivo inteiro e a “string-split” que encadeadas o transformam em lista, construímos a função “reader” utilizando portas e leitura caractere a caractere, essa abordagem nos permitiu mais controle no fluxo de dados que entra na função, que lê caracter por caracter até o final do documento, separando-o em palavras. É feito um filtro dos caracteres válidos, ou seja, pontuações em geral são ignoradas.

Rotinas de filtragem de caracteres e regras específicas para determinadas palavras foram possíveis já que conseguimos controlar a montagem da palavra.

A função writer foi escrita originalmente como uma função de despejo com um caminho de arquivo, mas teve de ser alterada pois como usamos múltiplos arquivos, foi mais conveniente utilizar “wrappers” ou funções que usam a writer em cada item de uma lista (passando para ela uma porta específica) e antes e depois de iterar manipulam essas portas. Isso permite reutilizar essas funções em outros arquivos de dados, ou outro número de arquivos de dados. Ela escreve um conjunto de dados em formato JS.

Módulo Workers, que contém a função de contagem.

A função de contagem agrupa uma lista de palavras em uma lista de pares de “palavra e total”, é uma função pura mas de custo elevado e que nos testes que fizemos em máquinas de quatro núcleos (dois hyper threading) e seis núcleos (todos “reais”) foi distribuída para exatamente a metade dos núcleos disponíveis. Não conseguimos achar recursos para sustentar o porquê desse episódio, mas é reproduzível e observável nos três ambientes de produção que utilizamos, dois Windows 10 e um Ubuntu 14.

Ainda nesta função foi implementado um recurso de filtro de palavras por uma lista de exclusão. É importante realizar essa filtragem, pois há algumas palavras que não possuem valor semântico (como alguns pronomes ou artigos), e apenas se repetem com frequência para formar e conectar frases. Portanto, retiramos palavras como “fw”, “rw”, “what”, “is” e “who”, porém, mantivemos outras, como “h” e “no”. Nesse caso, o “h” se refere à própria Hillary mas de maneira abreviada. O “no” se repetido muitas vezes, pode indicar que o conteúdo das mensagens possuí um aspecto negativo, de insatisfação. Cada palavra a ser mantida ou excluída foi avaliada individualmente, a lista de exclusão de palavras e caracteres está em /lib.

Módulo WebCrawler, que foi construído mas não chegou a ser implementado.

Esse módulo extrai texto do site HuffingtonPost, para ser processado pelas mesmas funções e em mesmo formato que os arquivos de email.

Em alto nível, o caminho dos dados é o que segue:

Texto → Função Reader → Função filter padrão (com uma lista personalizada de palavras para exclusão) → Função counter → Função filter padrão (estabelece um mínimo de ocorrências) → Writer (o cloud no caso).

Os scripts que invocam essa cadeia e geram os dados são "linear.scm" e "tagcloud.scm"

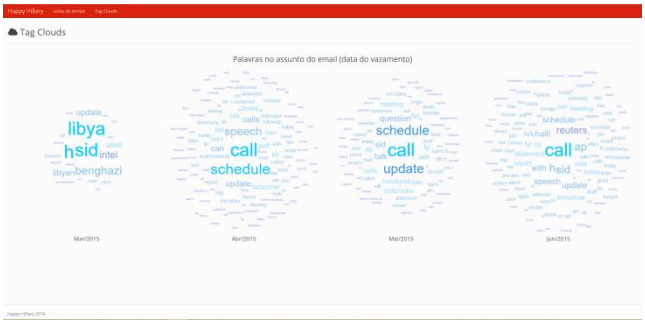
6. REPOSITÓRIO

O repositório está no endereço https://github.com/lnncrs/HappyHillary/tree/fase03, mantivemos os branches fase01, fase02 e fase03 pois representam as fases reais de produção do projeto. O Branch fase03 é a versão atual.

7. TELAS DE SAÍDA

Após o processamento, dentro da pasta "/reports" a página índex.html dá acesso aos dois relatórios atualmente no projeto, o linear:

Que mostra acima uma série com totais de mensagens pelo mês de envio e abaixo uma série com totais de mensagens pelo mês de vazamento



E o relatório de tags por mês, Que mostra uma tagcloud de palavras nos assuntos de email por mês de vazamento (é possível substituir por palavras do corpo do email, mas o trabalho é bem mais custoso)

8. ACKNOWLEDGMENTS

Nossos agradecimentos ao professor pelo curso e a ACM SIGCHI por permitir modificarmos os templates que desenvolveram.

9. REFERÊNCIAS

[1] What We Know About Hillary Clinton's Private Email Server - By ALICIA PARLAPIANO - UPDATED AUG. 23, 2016 – The New York Times http://www.nytimes.com/interactive/2016/05/27/us/politics/what-we-know-about-hillary-clintons-private-email-server.html

[2] The Huffington Post http://www.huffingtonpost.com/

[3] WikiLeaks Searchable database https://wikileaks.org/clinton-emails/

[4] Scheme I/O http://www.scheme.com/tspl3/io.html

[5] StackOverflow – File I/O Scheme http://stackoverflow.com/questions/4181355/file-i-o-operations-scheme

[6] Utilitário BCP (exportação de dados) https://msdn.microsoft.com/pt-br/library/ms162802.aspx https://www.simple-talk.com/sql/database-administration/working-with-the-bcp-command-line-utility/

[7] Suporte a UTF-8 no BCP https://support.microsoft.com/en-us/kb/3136780

[8] Componente JQCloud (exibição de dados) http://mistic100.github.io/jQCloud/

[9] ACM SIG Templates http://www.acm.org/publications/proceedings-template

[10] Font Awesome (iconografia) http://fontawesome.io/

[11] Bootstrap (HTML5 framework) http://getbootstrap.com/

[12] The R5RS Standard https://wiki.call-cc.org/man/4/The%20R5RS%20standard

[13] Unit SRFI-1 https://wiki.call-cc.org/man/4/Unit%20srfi-1

[14] Unit SRFI-13 https://wiki.call-cc.org/man/4/Unit%20srfi-13