Projecto RGI (Retriever for Gmail in the Internet): Uma Ferramenta para Recuperação de Emails

Tiago Garcia
Instituto Superior Técnico
tiago.garcia@ist.utl.pt

João Aires
Instituto Superior Técnico
joao.aires@ist.utl.pt

Palavras-chave
Recuperação de Informação, Email, Recuperação de Email

1. INTRODUÇÃO

Nos dias de hoje, o uso de *email* é algo que se encontra perto do irracional. Esta tecnologia é sem sombra de dúvida algo completamente indispensável na vida de grande parte das pessoas, quer por razões profissionais ou de negócios, quer nas suas vidas pessoais.

Segundo os números revelados pela *Royal Pingdom*[1], no ano de 2010 foram enviados cerca de 32 mil milhões de mensagens de *email* úteis por dia (tendo em conta que 89% do total são consideradas spam). Esses números ainda indicam que existem cerca de 1,88 mil milhões de utilizadores de *email* pelo mundo. Isto significa que, em média, cada pessoa recebe cerca de 17 *emails* úteis por dia. Isto significa que no decorrer de um ano, um utilizador de *email* dentro da média terá recebido mais de 6000 mensagens. Com o tamanho de armazenamento elevado das caixas de correio electrónico, apagar mensagens deixou de ser uma necessidade, e devido ao trabalho que dá, os utilizadores preferem claramente não ter que o fazer. Isto tudo para chegar à conclusão de que se quisermos encontrar um *email* no meio duma caixa de correio electrónico, ler milhares de emails para encontrar uma agulha num palheiro é claramente pouco prático. Daí surge a necessidade de funcionalidades e ferramentas de recuperação de *email* que permitam a um utilizador encontrar com facilidade uma mensagem que procure, sem que tenha que perder uma quantidade de tempo impensável a ler emails não interessantes.

Este documento apresenta uma solução que permite fazer a recuperação de mensagens de *email* em contas Gmail. Esta solução será explicada em mais detalhe na secção seguinte. Depois serão apresentados alguns valores correspondentes a uma avaliação da ferramenta. Por fim, apresentaremos algumas conclusões.

2. SOLUÇÃO

Esta secção do documento serve para expor e explicar a solução, desde a arquitectura da aplicação, à explicação dos diversos módulos implementados, passando pela explicação da tecnologia utilizada.

2.1 Arquitectura

Na Figura 1, é possível ver uma vista de topo da arquitectura do sitema.

A aplicação está implementada como *web application*, sendo portanto a interacção com os utilizadores feita num *browser*. O *RGIServer* funciona como servidor HTML e é onde está implementada a camada de visualização.

Existe também um RetrievalServer que corre em paralelo ao RGIServer e do qual este é cliente. Este servidor é aquele que faz grande parte do "trabalho" comunicando com os módulos implementados no projecto. Este servidor está constantemente num *socket* à escuta, à espera que que o *RGIServer* lhe envie *queries* a ser resolvidas. Este servidor comunica com os módulos Indexer, Retriever e Tokenizer. A comunicação com o *Indexer* é para lhe indicar quando este deve indexar. O processo de indexação consiste em ir buscar *emails* a uma conta Gmail usando a *EmailLib* e indexá-los. A indexação dos *emails* é feita começando pela tokenização da informação a indexar, usando a função *Tokenize* do *Tokenizer*. Isto é feito iterativamente para cada parte de um *email* porque existem três índices para as três partes dum email: Remetente, Assunto e Corpo. Mais à frente será explicado o processo mais em pormenor. O Indexer cria, para além dos índices, a cache das mensagens e um ficheiro onde é guardada alguma informação pertinente ao cálculo do ranking de retrieval, informação possível de calcular no processo de indexação. Isto é feito porque é informação não depende do tempo nem da query e permite poupar tempo ao retornar resultados ordenados segundo o método de ranking implementado. A comunicação entre o *RetrievalServer* e o *Tokenizer* é só para o uso da função *TokenizeInput* que faz o *parse* do *tokenize input* devolvendo três listas: lista de termos normais, lista de termos obrigatórios, e lista de termos proibidos (estes dois últimos dois tipos de termos serão explicados mais à frente no documento). Depois de ter esta informação, o RetrievalServer passa-a ao Retriever que faz a recuperação dos resultados usando a informação que se encontra nos índices, na cache e no ficheiro que guarda informação dos vectores de termos.

2.2 Tecnologias Utilizadas

2.2.1 Python

A linguagem usada para desenvolver grande parte da aplicação foi a linguagem Python. Isto deveu-se a ambos os elementos do grupo terem tido bastante contacto com a linguagem recentemente, e portanto estava presente a noção do que era possível fazer com a linguagem. Também foi factor determinante o facto de que já serem conhecidas certas estruturas de dados que seriam boas para implementar o mecanismo de índices invertidos e formas eficientes de as serializar, usando bibliotecas da linguagem.

2.2.2 HTML/CSS

Para facilitar o desenvolvimento da interface, devido já a alguma experiência no grupo de desenvolvimento em aplicações com interface web, foi escolhido para a GUI usar então HTML/CSS para a interface.

[1] http://royal.pingdom.com/2011/01/12/internet-2010-in-numbers/

2.2.3 JQuery

No nosso projecto usámos ainda a *jQuery*, uma biblioteca de *JavaScript* que facilita em muito o desenvolvimento da lógica dos *websites* bem como a criação de eventos, animações, interacções, etc. Para além disso, esta biblioteca é independente do *browser* utilizado e tem ainda a particularidade de ser *open-source*. O facto de ser a biblioteca de Javascript mais utilizada hoje em dia foi outro factor importante.

2.3 Componentes/Mecanismos Implementados

2.3.1 Servidores

Para este projecto foram implementados dois servidores, o RGIServer e o RetrievalServer. Embora tinham sido implementados dois servidores, correm ambos na mesma janela, por uma questão de simplicidade.

O *RGIServer* é o responsável por processar os pedidos http vindos do *browser* e gerar as páginas html correspondentes sendo que pode ainda correr scripts CGI. Uma vez que o servidor foi implementado em python, foi possível utilizar as classes python HTTPServer e CGIHTTPServer para assim só adicionar o comportamento específico da nossa aplicação.

Já o *RetrievalServer* é responsável por fazer a recuperação dos *emails*. Basicamente, este servidor está à escuta num *socket* à espera que lhe dêem uma *string* que corresponde a uma *query* introduzida pelo utilizador. Após processar a *query* e calcular os *emails* que estão de acordo com a *query* fornecida, os mesmos são devolvidos. Embora seja necessário enviar os identificadores de todos os *emails* calculados, por motivos de eficiência apenas é enviado o remetente, assunto, data e corpo dos 20 primeiros emails: uma vez que os resultados são organizados em páginas de 20 resultados cada, o resto da informação só será transmitido quando, e se, o utilizador mudar para uma outra página de resultados. Verificámos que, para queries com mais de 1000 resultados, esta medida permitiu-nos uma poupança de tempo que ultrapassa os 90%. Paralelamente, este servidor é ainda responsável pela indexação automática dos *emails*, actualizando os índices de forma regular.

2.3.2 Módulo para fetch dos emails

O módulo desenvolvido para fazer o *fetch* dos *emails* recorre ao protocolo IMAP, um protocolo de gestão de email suportado pelo Gmail. A biblioteca *imaplib* do Python já contém alguns dos mecanismos mais básicos para fazer a recuperação de *emails*, mas aspectos mais avançados tiveram que ser resolvidos com recurso a outras bibliotecas e/ou técnicas. Em especial, o *parsing* dos *emails* revelou-se um desafio bastante moroso.

Resumidamente, esta biblioteca possui um conjunto de métodos para aceder ao servidor IMAP do Gmail e assim retornar os *emails* nele contido.

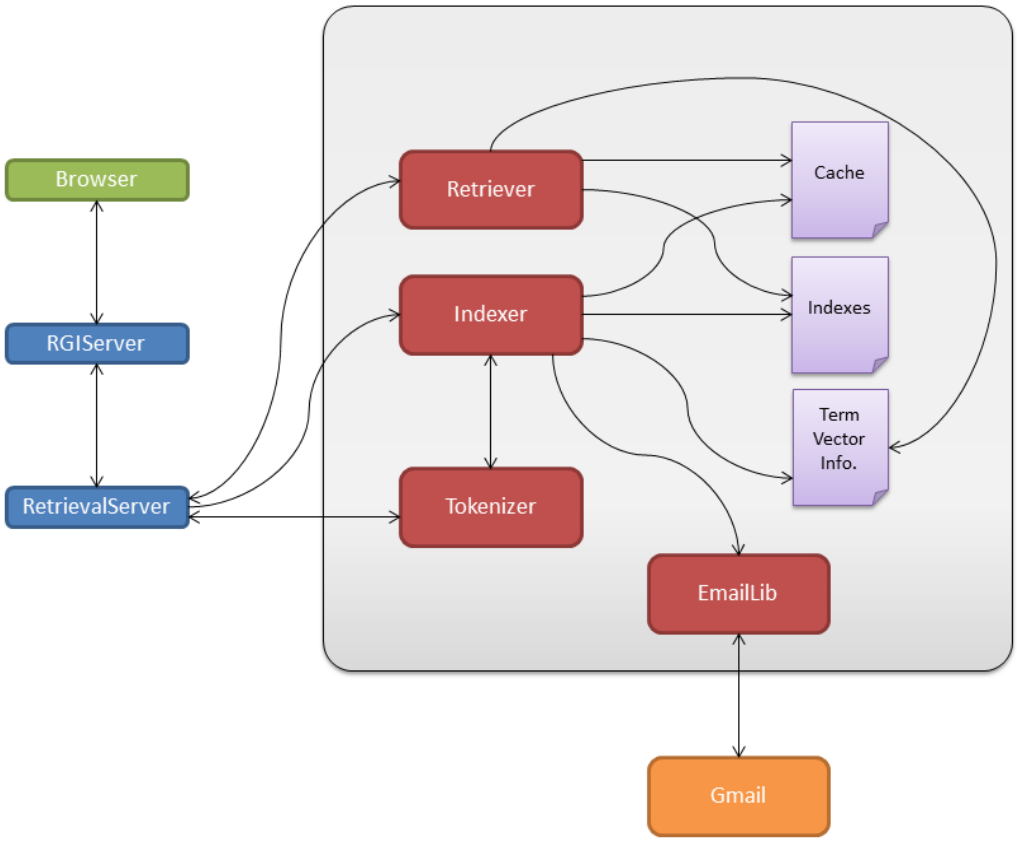


Figura 1 – Vista de topo da arquitectura do sistema

2.3.3 Módulo para indexação

O mecanismo de indexação foi implementado usando umas estruturas de dados de Python, "dicionários". Esta estrutura permite um mapeamento entre um tipo ou estrutura de dados que se comporta como chave e outro(a). A existência desta estrutura é a indicada para conseguir implementar facilmente a estrutura de indexação escolhida para a ferramenta: índices invertidos. Os índices invertidos chamam-se assim por serem invertidos em relação ao que seria talvez mais convencional: em vez de para cada "documento" ter uma lista de termos associados, para cada termo, tem uma lista de "documentos" que o contêm. Neste caso, os "documentos" são mensagens de *email*, identificadas pelo campo *Message-ID* do próprio email, identificador unívoco atribuído pelo *host* que gera a mensagem de email.

A forma como foi implementado o mecanismo de indexação foi relativamente simples. Antes de ser começado o processo de indexação propriamente dito, todas as sequências de caracteres a indexar são passadas por um *Tokenizer* que usando expressões regulares, divide o *input* em termos e elimina as chamadas *stopwords*, palavras demasiado comuns que, se consideradas, têm grande probabilidade de devolverem demasiados resultados sem interesse.

Existe um glossário e três índices invertidos. O glossário contém todos os termos indexados pela aplicação em que cada um está associado a um número do tipo inteiro, para ser mais rápida a computação das procuras nos índices. Os três índices são para as três partes distintas de um *email*: remetente, assunto e corpo da mensagem. Ao separar a informação pelos diversos índices veio simplificar várias coisas. Para começar, veio permitir a procura em campos específicos, facilitando a filtragem da procura por qualquer combinação desses três campos. Esta separação também trouxe uma mais-valia muito grande para a implementação do mecanismo de ordenação com base num ranking que se baseia em que partes dum *email* aparecem os termos para atribuir um valor de relevância a um e*mail* quando é feita uma determinada *query* (ver 2.3.4.2). Durante este processo também é guardada uma lista dos identificadores dos *emails* já vistos para que num processo de actualização dos índices não seja perdido tempo a processar de novo todos os emails e possam ser ignorados os já vistos.

Adicionalmente, o módulo de indexação é ainda responsável por outros cálculos que, por motivos de alinhamento com o resto do documento, serão descritos e debatidos na secção 2.3.4, mais concretamente no ponto "*Ordenação optimizada por relevância*

2.3.4 Retrieval

O *retrieval* é o processo de recuperação de informação, neste caso, de *emails*. Para além de simplesmente receber alguns termos e encontrá-los, a ferramenta faz mais algumas coisas que serão explicadas nas próximas subsecções. Essas funcionalidades incluem, por exemplo, permitir que se obrigue a que os resultados tenham ou não determinados termos, permitir escolher ordenar resultados por data ou usando o mecanismo de ranking implementado e ainda permitir fazer a busca em campos específicos do *email*.

2.3.4.1 Termos Obrigatórios/Proibidos

Quando um utilizador faz uma pesquisa com mais do que um termo, o que acontece na prática é que o conjunto de resultados corresponde à união dos conjuntos de e*mails* que incluem cada termo, ou seja, é como um *or* booleano. Tendo em conta que em certos casos pode haver demasiados resultados a devolver e o utilizador pode querer diminuir esses resultados foi implementado um mecanismo que o pode fazer.

Na sequência de termos apresentados, caso haja termos precedidos do carácter '-', significa que nenhum dos resultados pode conter esses termos. Por sua vez, caso haja termos precedidos do carácter '+', significa que os *emails* no conjunto de resultados tem que conter esses termos, ou seja, do conjunto de resultados proveniente dos termos neutros, só ficam aqueles que contenham esses termos.

É claro que existe potencialidade para conflitos entre resultados, portanto teve que ser estabelecida uma ordem de precedência em que aqueles que contêm termos "proibidos" são excluídos, mesmo que contenham termos obrigatórios. Portando o processo acontece da seguinte forma: Primeiro cria-se o conjunto inicial unindo os conjuntos de mensagens que contenham termos neutros. Depois disso, são eliminados aqueles que não contêm os termos obrigatórios. No fim são eliminados os que contêm termos proibidos.

2.3.4.2 Ordenação de Emails

A ordenação dos resultados num sistema de recuperação de *emails* é um aspecto absolutamente fundamental: de nada serve ter os resultados certos se o utilizador se aborrecer de procurar os resultados que pretende antes de chegar a eles. Nesse sentido, foram implementadas duas soluções para a ordenação dos *emails*.

Ordenação Temporal

Tipicamente, os serviços de recuperação de *email* clássicos optam por devolver os resultados calculados por ordem inversa à cronológica (ou seja, o resultado mais recente em primeiro lugar). Tal ordenação pode ser útil quando o utilizador tem uma noção mais ou menos clara do contexto ou da altura em que ocorreu o *email* que procura.

Ordenação por relevância

A ordenação temporal, apesar das suas vantagens, não é muitas vezes a melhor solução uma vez que implica uma procura potencialmente demorada por parte do utilizador que pode, no limite, abandonar a procura ao fim de poucos minutos.

Nesse sentido, foi desenvolvido um algoritmo para o cálculo da relevância de um *email*. Para o cálculo de tal relevância (*relevance*) são considerados não só os termos contidos no *emails* mais ainda os campos do email em que estes termos ocorrem. A equação seguinte apresenta a função proposta, no nosso sistema, para o cálculo da relevância de um *email* para uma dada *query*.

$$relevance(email, query) = \alpha \cdot relevance_{remetente}(email, query) + \beta \cdot relevance_{assunto}(email, query) + relevance_{corpo}(email, query)$$

$$com: \alpha > \beta > 1$$

Foram considerados apenas os campos do remetente, assunto e o corpo do próprio *email*, por serem estes os campos que, mais naturalmente, o próprio ser humano utilizada para inferir a importância de um determinado email. O cálculo da relevância passa então pelo somatório da relevância da query para cada um destes três campos. De forma a dar mais importância a uns campos, face a outros, foram introduzidos factores multiplicativos neste somatório (nomeadamente α e β em que $\alpha > \beta > 1$). A introdução destes factores garantem assim que o campo do remetente é o que mais importante para a relevância do *email*, seguindo-se o assunto e, por fim, o corpo do email. O racional por detrás desta opção é explicado nos três parágrafos seguintes.

O remetente tem, tipicamente, o nome ou o *email* dos nossos contactos. Assim, se a *query* introduzida contiver "josé", é de esperar que o utilizador esteja à procura de *emails* enviados por um José e não de emails em que o nome José é mencionado. Este é o racional por detrás do facto do remetente ter sido considerado o campo mais importante para a relevância de um *email*.

O assunto do *email* é o campo onde normalmente se encontram as palavras chave do email. Assim sendo, é sensato considerar este o segundo campo mais importante para a relevância de um *email* uma vez que aqui podem ser encontradas muitas das palavras chave do próprio email.

O corpo do *email* é um pedaço do texto, normalmente de maior dimensão que o remetente e o assunto, que descreve o propósito do email. Embora possa conter muitos termos chave para o *email*, é no entanto de esperar que contenha muitos outros menos relevantes para o mesmo. Por isso mesmo, foi considerado o campo menos importante para o cálculo da relevância do *email*.

A determinação dos valores de α e de β de forma a obtermos bons resultados foi, por si só, também parte importante do trabalho. Após alguns testes chegámos a uns valores que considerámos sensatos e que nos permitiram obter os resultados que pretendíamos. Esses valores foram α = 25 e β =15.

Para o cálculo da relevância de cada um dos três campos é calculada a semelhança do coseno (função *Sim*) entre o vector de termos da query (*query_vt*) e o vector de termos de cada um desses campos (*remetente_vt, assunto_vt e corpo_vt*), tal como se pode observar nas equações seguintes.

$$relevance_{remetente}(email, query) = Sim(remetente\_vt, query\_vt)$$
$$relevance_{assunto}(email, query) = Sim(assunto\_vt, query\_vt)$$
$$relevance_{corpo}(email, query) = Sim(corpo\_vt, query\_vt)$$

Por sua vez, cada posição nos vectores de termos é calculado de acordo com o TF-IDF[1] o que permite obter uma boa estimativa da importância de um termo em cada um dos campos e mesmo na própria *query*. Uma vez que existe o conceito de termo obrigatório (ver 2.3.4), decidimos considerar os obrigatórios mais importantes que os não obrigatórios. Assim sendo, e resumindo, o valor da posição de um termo no vector de termos da query (*pos(termo, query)*) é dado pela seguinte equação:

$$pos(termo, query) = \gamma \cdot TF\text{-}IDF(termo, query)$$
$$com: \quad \gamma = 2, \text{ se termo e obrigatórios}$$
$$\gamma = 1, \text{ caso contrário}$$

Note-se que a mesma equação pode ser utilizada para calcular o vector de termos do remetente, assunto e corpo do *email*.

Ordenação optimizada por relevância

Numa primeira abordagem, o cálculo de relevância (função *relevance*) de cada um dos campos de *email* era totalmente calculada em *retrieval time* (ou seja, quando o utilizador introduz a *query* para a procura). De forma a poder optimizar este processo e diminuir assim o tempo de resposta da aplicação (o tal *retrieval time*) parte deste cálculo foi transferido para o momento da indexação (*indexing time*). No fundo, o que se fez foi transferir o cálculo dos vectores de termos de cada parte do *email*, que era realizado em *retrieval time*, para o momento em que se faz indexação dos emails. Isto implica no entanto calcular vectores de termos que contenham todos os termos da colecção, o que aumenta um pouco o tempo de indexação. Tal aspecto não é no entanto importante, uma vez que aumentar o tempo de indexação não é considerado um problema de maior se, com isso, conseguirmos diminuir o tempo de resposta do sistema. O resto do cálculo da relevância, nomeadamente o cálculo da semelhança do coseno com a query introduzida, é feito em *retrieval time*, uma vez que depende totalmente da query em questão. Para efeitos do projecto foram mantidas ambas as abordagens à disposição do utilizador na interface gráfica (opções *"Relevance (on the fly)"* e *"Relevance"* no campo *"Sorted by"*, abaixo da barra de pesquisa.), até para facilitar a avaliação dos benefícios da optimização. Num produto final, a abordagem mais lenta seria obviamente eliminada, por não apresentar qualquer benefício para o utilizador final.

Uma comparação entre estas duas abordagens, e a ordenação cronológica, pode ser encontrada no ponto 3.2.3 deste documento.

2.3.4.3 Busca em campos Específicos

Para facilitar a recuperação de um *email*, um utilizador pode querer procurar por um campo específico. Por exemplo, se o utilizador souber que o *email* foi enviado por *John Smith* pode escrever essas palavras como termos e fazer a procura apenas no índice do remetente, devolvendo desta forma um conjunto de resultados mais pequeno e mais rápido, visto que apenas um índice é percorrido. Este mecanismo está ainda disponível para os campos assunto e corpo do *email*.

2.3.4.4 Filtros para os remetentes

De forma a ajudar os utilizadores a filtrarem os resultados obtidos com a query submetida desenvolvemos filtros para os remetentes.

O seu funcionamento é simples: após obter os resultados de determinada *query*, o sistema coloca, do lado esquerdo dos resultados, a lista de remetentes para os *emails* presentes nessa lista de resultados (Figura 2). O utilizador pode então seleccionar o ou os remetentes que lhe interessam ver e o sistema apresentará apenas os *emails* enviados por esses remetentes. Alternativamente, o utilizador pode ainda seleccionar o ou os remetentes que não lhe interessam ver e o sistema, desta feita, apresentará os emails de todos os remetentes que não estejam naquela lista.

Desta forma o utilizador pode filtrar os resultados por remetente.

Figura 2 - Filtros para os remetentes

2.3.4.5 Interface Gráfica

A interface gráfica de qualquer aplicação é sempre um ponto fundamental para o sucesso da mesma: uma interface mal feita e com pouca usabilidade pode deitar por terra todo o restante trabalho, por mais brilhante que este possa ser.

Foi portanto com algum cuidado que desenvolvemos a nossa interface gráfica. O aspecto geral da nossa interface pode ser observado na Figura 3.

Como se pode observar na figura acima, os resultados obtidos são apresentados em barras azuis que indicam apenas o remetente, assunto e data do *email*. Para aceder ao conteúdo do corpo do *email*, o utilizador só necessita de carregar na barra do email correspondente (Figura 4). De resto, e tal como já foi referido anteriormente, os resultados são apresentados em páginas de 20 resultados cada por considerarmos este o número de *emails* que os utilizadores estarão dispostos a percorrer até desistir da busca ou recomeçarem com outras palavras-chave.

Figura 3 - Interface gráfica do nosso projecto.

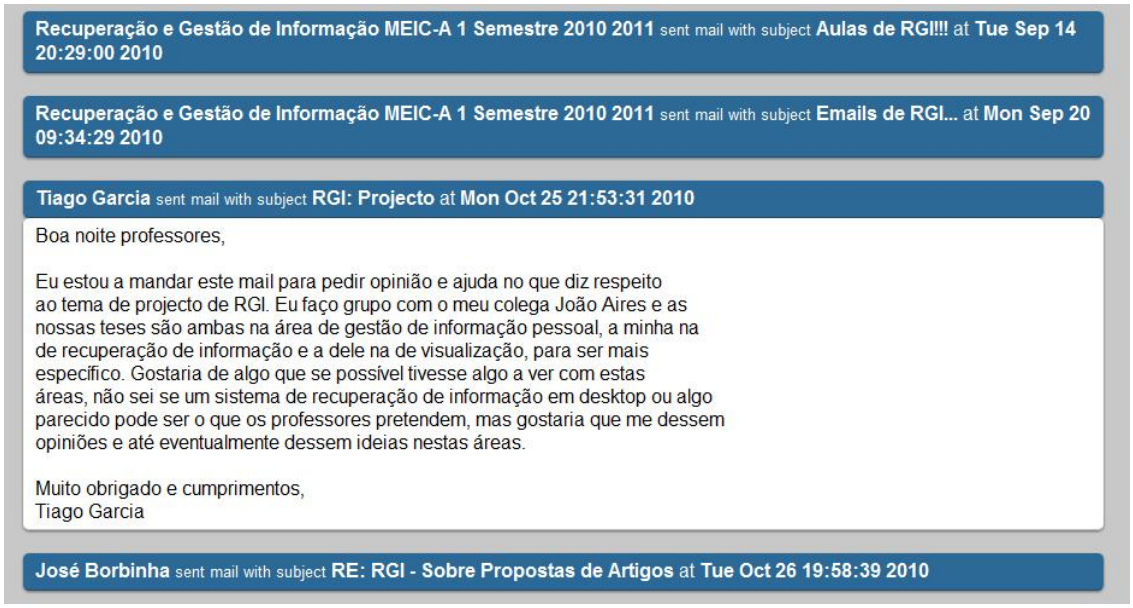


Figura 4 - O utilizador pode aceder ao corpo do *email* carregando no email desejado.

3. AVALIAÇÃO

3.1 Descrição da Avaliação

De forma a podermos a avaliar a solução proposta, decidimos fazer testes comparativos entre o nosso projecto, o sistema de recuperação de *emails* que o Gmail fornece como padrão e, aquilo a que se poderia chamar de, um motor de busca de email "tradicional". O motor de busca tradicional considerado foi o sistema do próprio Gmail mas recorrendo apenas à busca simples do mesmo.

Como medidas de comparação, decidimos usar a *precision* e o N*DCG @ 10* (*Normalized Discounted Cumulative Gain*[2] até ao 10º email). Escolhemos estas métricas por julgarmos serem aquelas em que o nosso sistema mais se poderá destacar em relação aos outros sistemas em análise. Outras medidas como o *recall* poderiam ser também testadas mas consideramos que o *recall* obtido no nosso sistema deverá ser muito semelhante (se não mesmo igual) ao *recall* obtido nas outras duas soluções uma vez que não julgamos ter implementado qualquer medida que nos permita esperar um melhor valor de *recall*.

Para estes testes, decidimos recorrer a dois cenários que espelham o uso típico deste tipo de mecanismos de recuperação. Estes cenários são descritos, mais à frente, neste documento. Por fim, para estes testes decidimos utilizar os *emails* presentes na caixa de entrada e de saída do correio electrónico de um dos membros do grupo do projecto, João Aires. Poderia ter sido utilizado um qualquer outro conjunto de *emails*, mas considerou-se mais conveniente usar um conjunto previamente conhecido, em que já temos algum conhecimento do seu conteúdo para assim melhor simular o contexto em que, mais normalmente, ocorre a recuperação de emails em sistemas como o Gmail (ou seja, quando o dono de uma conta de email procura um ou vários emails). Para além do mais, usar um outro conjunto de *emails* implicaria transferir todos esses emails para uma conta do Gmail para assim podermos usar esse sistema. Tal processo poderia ser moroso sem que esse investimento de tempo, na nossa opinião, representasse uma mais-valia. No total, esta colecção contempla mais de 5000 *emails*.

Cenário I:
Tarefa: Encontrar o *email* enviado pelo regente da cadeira de RGI em que é indicado o template a utilizar no *state of the art* que faz parte da avaliação desta cadeira.

Cenário II:
Tarefa: Encontrar o conjunto de *emails* enviados pelo colega de grupo (Tiago Garcia) sobre o projecto de RGI. *Emails* que não sejam sobre o projecto, i.e. que apenas o refiram, devem ser considerados não relevantes.

No caso específico dos nossos testes, apenas houve necessidade de considerar dois níveis de relevância: relevante (relevância = 1) e não relevante (relevância = 0).

Adicionalmente, realizámos ainda testes de comparação ao tempo de resposta das três abordagens de ordenação que implementámos. Para tal foram utilizadas as seguintes queries:

Query	Número de resultados
"tiago"	579 Resultados
"ist"	1432 Resultados
"cumprimento"	3389 Resultados
"aires figueira ist"	4877 Resultados

3.2 Resultados da Avaliação

3.2.1 Cenário I

A tabela seguinte resume os resultados obtidos para o primeiro cenário.

Sistema	Query / Filtros	Precision	NDCG
Clássico	Recuperação template paper; rgi template paper;	0.25	0.195
Gmail	Recuperação OR rgi template paper	0.25	0.195
RGI	recuperação rgi +template +paper	0.25	1.0

Tal como se pode observar, a *precision* obtida é igual para todos os sistemas uma vez que todos eles retornaram 4 resultados, onde apenas um era relevante (aquele que se estava à procura).

O valor do *NDCG* para o sistema clássico e para o Gmail é idêntico uma vez que ambos os sistemas retornam os resultados ordenados por ordem inversa à cronológica. Assim sendo, o *email* que se pretendia encontrava-se na 3º posição da lista. Por outro lado, uma vez que a nossa solução ordena os resultados por relevância, os resultados obtidos apresentaram o *email* pretendido em primeiro lugar, o que dá origem a um NDCG de 1.0.

3.2.2 Cenário II

Sistema	Query / Filtros	Precision	NDCG
Clássico	rgi projecto tiago garcia	0.25	0.278
Gmail	rgi projecto from:tg.shady@gmail.com	0.308	0.5
RGI	rgi +projecto Filter by senders:Tiago Garcia	0.571	0.704

O sistema clássico teve os piores resultados neste cenário. O facto de não ser possível especificar que se pretende apenas os *emails* enviados por determinada pessoa está na base dos seus resultados: 3 dos 16 *emails* retornados não foram enviado pelo Tiago Garcia, mas sim por outra pessoa mencionava o seu nome.

Com o Gmail já é possível especificar o remetente mas, uma vez que o Gmail retorna *threads* inteiras, foram retornados não apenas *emails* enviados pelo remetente pretendido mas sim todos os

emails contidos em *threads* onde o remetente pretendido participou (13 emails no total, em que apenas 4 são relevantes).

A diferença do *NDCG* entre o sistema clássico e o Gmail prende-se apenas com o facto de o sistema clássico ter retornado mais resultados irrelevantes uma vez que, de resto, ambos ordenam os resultados segundo o mesmo critério (tempo).

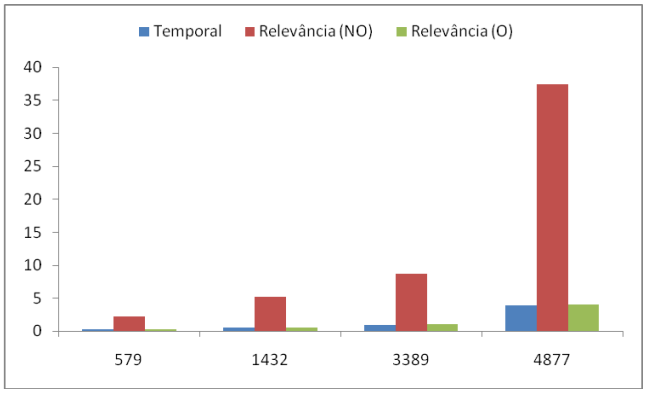
Já o nosso sistema retornou apenas 7 resultados, todos eles enviados pelo remetente pretendido e com os 3 resultados considerados irrelevantes nas últimas posições o que explica os bons resultados apresentados na tabela anterior.

3.2.3 Comparação das abordagens de ordenação

A tabela seguinte resume os tempos de resposta de cada um dos métodos de ordenação para cada uma cada *queries* testadas (ver 3.1).

Ordenação	**Tempo de resposta (s)**			
	579 resultados	**1432 resultados**	**3389 resultados**	**4877 resultados**
Temporal	0.242	0.545	1.01	3.88
Relevância (NO)	2.248	5.171	8.711	37.422
Relevância (O)	0.274	0.565	1.129	4.108

O gráfico seguinte apresenta estes mesmos valores de forma mais gráfica o que facilita a compreensão dos mesmos.



Analisando a tabela e o gráfico facilmente se tiram duas conclusões importantes:

- A relevância optimizada, Relevâcia (O), apresenta grandes melhorias face à relevância não optimizada (NO). Esta diferença tende, obviamente, a aumentar com o número de resultados retornados. No caso da query com mais resultados, o modo optimizado teve um desempenho de cerca de 13 vezes mais rápido.
- A diferença de desempenho entre a ordenação temporal e a ordenação por relevância é mínima.

Estes resultados permitem afirmar que o nosso algoritmo de ordenação com base na relevância dos *emails* é uma alternativa perfeitamente admissível, pelo menos em termos de desempenho, à típica ordenação temporal.

4. CONCLUSÃO

Encontrar um *email* ou conjuntos de emails numa caixa de correio electrónico pode ser uma tarefa muito demorada para o utilizador caso não se utilizem as ferramentas certas.

Sistemas de recuperação de *email* como o disponível no site do Gmail ordenam os resultados obtidos por ordem cronológica e optam ainda por retornar *threads* ao invés de devolver emails individuais. Estas duas opções acabam por afectar os seus resultados, mas concretamente ao nível da *precision* e do *NDCG*. Ao retornar os *emails* ordenados por ordem inversa à cronológica, não usa qualquer heurística que lhe permita inferir a importância de cada um dos emails, pelo que o *NDCG* é afectado, a menos que se queiram sempre os emails mais recentes. A *precision* é afectada no sentido em que, tipicamente, se procura um *email* em particular e não uma *thread* inteira. Ao devolver toda uma *thread*, a probabilidade de devolver um *email* não relevante é maior o que diminui a *precision*.

A nossa proposta tenta combater estes e outros aspectos menos bons apresentados pelo Gmail. Para tal implementámos um algoritmo de relevância que ordena os resultados segundo uma série de heurísticas que calculam a relevância de cada *email*. Os resultados dos nossos testes mostram ainda que a diferença de desempenho (em tempo) entre o nosso algoritmo de ordenação por revelância e um algoritmo de ordenação temporal é mínima para volumes de resultados considerados "normais". Como consequência, o *NDCG* apresentado pelo nosso sistema revela-se melhor face àquele apresentado pelo Gmail sem que seja de esperar uma diminuição significativa no desempenho do sistema. O facto de o nosso sistema devolver apenas os *emails* realmente relevantes (e não toda a *thread*) acaba por melhor também a *precision* do nosso sistema, face ao valor apresentado pelo Gmail.

Por fim, implementámos ainda uma série de outros aspectos que permitem procuras mais interessantes face ao que é costume encontrar em motores de recuperação de *emails* tradicionais.

REFERÊNCIAS

[1] Manning, C.D., Raghvan, P., & Schütze, H (2008). *Introduction to Information Retrieval*. Cambridge University Press.

[2] Järvelin, K., & Kekäläinem, J. (2002). *Cumulated Gain-based Evaluation of IR Techniques*. AM Transactions on Information Systems (TOIS), 20(4), 422-446