Trabalho Prático 2
Recuperação de Informação

Luís E. O. Lizardo[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais
Caixa Postal 702 – 30.123-970 – Belo Horizonte – MG – Brasil

lizardo@dcc.ufmg.br

Resumo. *Máquinas de busca revolucionaram a Internet ao possibilitarem que usuários tenham acesso aos conteúdos mais diversos e de forma rápida. Devido ao grande tamanho da Web, essas máquinas precisam ser eficientes e ainda precisas em relação as respostas retornadas aos usuários. Neste trabalho são implementados e analisados quatro modelos de recuperação de informação: Booleano, BM25, Cosseno e um modelo resultante da combinação dos modelos BM25 e Cosseno. As consultas foram realizadas sobre o índice invertido desenvolvido no Trabalho Prático 1. Também foi desenvolvido uma interface Web para permitir consultas ao índice. Uma avaliação experimental mostrou que o BM25 apresentou resultados de precisão e revocação superiores aos demais.*

1. Introdução

Máquinas de busca são mecanismos importantes na era da Internet. Elas permitem procurar por páginas Web tendo como base palavras chaves ou expressões, e retornam de forma eficiente uma lista de páginas candidatas a resposta da consulta. A forma mais simples de se realizar ou processar uma consulta, é retornar todas as páginas que contém os termos pesquisados, porém, desta forma nem todas as páginas retornadas podem ser relevantes para o usuário, mesmo que elas possuam o termo pesquisado. Ocorre também de inúmeras páginas serem encontradas, e aquela que o usuário realmente precisa, aparecer entre as últimas exibidas, dificultando assim sua pesquisa.

Identificar páginas relevantes a uma consulta em um índice é uma tarefa difícil e incerta, pois uma página considerada relevante para um usuário, pode não ser para outro. O que as máquinas de busca fazem é utilizar modelos de ranqueamento para predizer uma ordem de relevância das páginas encontradas. O objetivo deste trabalho é implementar e avaliar modelos de ranqueamento. Foram implementados os modelos: Booleano (sem a expressão *NOT*), BM25, de espaço vetorial (Cosseno) e um modelo desenvolvido com a combinação dos modelos BM25 e Cosseno. Também foi desenvolvido uma interface Web, que permite ao usuário fazer consultas utilizando os modelos implementados. As consultas são realizadas sobre o índice construído no Trabalho Prático 1.

2. Melhorias no TP1

A seguir são listadas as principais melhorias realizadas nos programas desenvolvidos no TP1.

- Antes, os processos de *parsing* da coleção e de ordenação do arquivo temporário eram feitos em programas separados. Agora essas duas rotinas foram integradas em um único programa, o *index_writer_sorter*.

- Redução do número de bytes de *padding* inseridos no arquivo temporário para a ordenação. Antes o *padding* era inserido até completar o tamanho da *run* estabelecida. Agora são inseridos apenas o suficiente para completar o tamanho de um bloco, setado por padrão como 50KB, que é muito menor que o tamanho de uma *run*.
- Informações sobre os documentos, como título e URL, agora são salvas em um arquivo para serem utilizadas durante as consultas. Também é salvo um segundo arquivo que funciona como índice para o primeiro.
- O algoritmo de compressão foi reimplementado utilizando *arrays* do tipo *char*. Essa nova implementação é muito mais eficiente em tempo de execução que a anterior, que utilizava um *bitvector*. Na Seção 7.1 são apresentados resultados comparativos de construção do índice com e sem compressão.
- O processo de construção do índice foi modificado para calcular a norma vetorial de cada documento e também o TF-IDF de cada termo. Esses valores são armazenados em memória e são utilizados pelos modelos de consulta.

3. Modelos

Nesta seção são apresentados os modelos Cosseno, BM25 e um modelo resultante da combinação dos dois, chamado neste trabalho de Custom.

3.1. Vetorial

O Modelo Vetorial ou Modelo de Espaço Vetorial representa cada documento como um vetor de termos e, cada termo possui um valor associado que indica seu grau de importância (peso) para o documento.

A representação dos documentos dj e da consulta q são vetores t dimensionais dados por:

$$\vec{d_j} = (w_{1,j}, w_{2,j}, ..., w_{t,j})$$
$$\vec{q} = (w_{1,q}, w_{2,q}, ..., w_{t,q})$$

onde

$$w_{i,j} = (1 + \log f_{i,j}) \times \log\left(\frac{N}{n_i}\right)$$
$$w_{i,q} = (1 + \log f_{i,q}) \times \log\left(\frac{N}{n_i}\right)$$

O modelo vetorial avalia o grau de similaridade entre um documento d_j e uma consulta q como uma correlação entre os vetores d_j e q. A correlação é quantificada pelo cosseno do ângulo entre esses dois vetores:

$$sim(d_j, q) = \frac{\vec{d_j} \bullet \vec{q}}{\left|\vec{d_j}\right| \times |\vec{q}|}$$

onde $\left|\vec{d_j}\right|$ e $|\vec{q}|$ são as normas dos vetores que representam a norma do documento e da consulta. Como $|\vec{q}|$ é o mesmo para todos os documentos, ele não afeta o *ranking*, e neste trabalho foi considerado sempre igual a 1.

Nos experimentos apresentados na Seção 7.3.1, foram utilizados 3 tipos diferentes de normalização dos documentos:

- Normalização vetorial, dada por:

$$\vec{d_j} = \sqrt{\sum_{i}^{t} w_{i,j}^2}$$

- Normalização pelo número de termos no documento; e
- Normalização igual a 1.

O algoritmo para calcular as similaridades dos documentos processa as palavras da consulta em ordem e mantem um conjunto de acumuladores para cada documento encontrado. Os acumuladores armazenam o somatório do TF-IDF. Por fim, o valor de cada acumulador e divido pela norma do documento. Neste trabalho os acumuladores são estruturas *Hash*, então a complexidade de tempo e espaço deste modelo é $O(n)$, onde n é o número de documentos encontrados pelos termos pesquisados.

Mais informações sobre o Modelo Vetorial podem ser encontradas em [Baeza-Yates et al. 1999].

3.2. BM25

O modelo BM25 foi criado como resultado de uma série de experimentos sobre variações da fórmula probabilistica clássica:

$$sim(d_j, q) \sim \sum_{k_i \in q \wedge k_i \in d_j} \log\left(\frac{N - n_i + 0.5}{n_i + 0.5}\right)$$

Esses experimentos foram motivados pela observação de que, como no modelo vetorial clássico, uma boa ponderação de termos é baseada em três princípios: (1) frequência inversa de documentos, (2) frequência dos termos e (3) normalização pelo tamanho dos documentos. A formula probabilistica clássica cobre apenas o primeiro princípio.

O BM25 foi motivado pela combinação de fatores de frequência de termos das fórmulas de ranqueamento BM11 e BM15, como segue:

$$ß_{i,j} = \frac{(K_1 + 1) f_{i,j}}{K_1 \left[(1 - b) + b \frac{len(dj)}{avg\_doclen}\right] + f_{i,j}}$$

onde b é uma constante introduzida com valores no intervalo [0,1]. Se $b = 0$, a equação acima é reduzida ao fator de frequência de termos usado na BM15, Se $b = 1$, ela é reduzida ao fator de TF da BM11. Para valores de b entre 0 e 1, a equação fornece uma combinação da BM11 com BM15.

A equação de ranqueamento do modelo BM25 pode ser escrita como:

$$sim_{BM25}(d_j, q) \sim \sum_{k_1 [q, d_j]} ß_{i,j} \times log\left(\frac{N - n_i + 0.5}{n_i + 0.5}\right)$$

onde K_1 e b são constantes empíricas na expressão para $ß_{i,j}$. Neste trabalho, o BM25 foi avaliado para $k_1 = 1$ e b iguais a 0, $0,25$, $0,50$, $0,75$ e $1,00$.

O algoritmo para calcular as similaridades deste modelo é similar ao do modelo vetorial. Sendo sua complexidade $O(n)$ em tempo e espaço.

Mais informações sobre o BM25 podem ser encontradas em [Baeza-Yates et al. 1999].

3.3. Custom

Este modelo foi criado por meio de uma combinação dos modelos Vetorial e BM25. Ele é o modelo vetorial normalizado com um peso b, como no modelo BM25. Sua similaridade é dada por:

$$sim(d_j, q) = \frac{\vec{d_j} \bullet \vec{q}}{\left[(1-b) + b\frac{len(dj)}{avg\_doclen}\right] + f_{i,j}}$$

O algoritmo para calcular as similaridades deste modelo é similar ao do modelo vetorial, linear em tempo e espaço em relação ao número de documentos encontrados.

4. Processador de Consultas

O processador de consultas foi desenvolvido para que ele possa, de forma simples, suportar diferentes modelos. Sua arquitetura é formada pela classe *Index*, que constroi e gerencia o arquivo invertido. Requisições a essa classe permite pesquisar por um termo, e ela responde com a lista invertida de documentos e frequências dos termo pesquisado. A classe abstrata *Query* instancia a classe *Index*. Todos os modelos de consultas desenvolvidos estendem essa classe *Query*, conforme esquematizado na Figura 1.

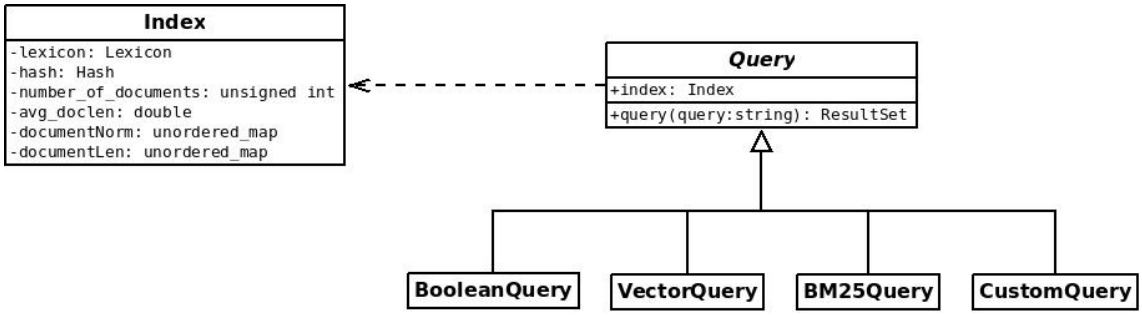


Figura 1. Esquema de classes simplificado do processador de consultas.

5. Interface Web

A interface Web foi desenvolvida para possibilitar o uso fácil do projeto. Essa interface comunica com o processador de consultas por meio de *sockets*. A interface faz uma requisição enviando o modelo de consulta escolhido pelo usuário e a consulta pesquisada e recebe como resposta os dados dos documentos no formato JSON. A Figura 2 exibe o esquema simplificado da comunicação entre a interface e o processador de consultas.

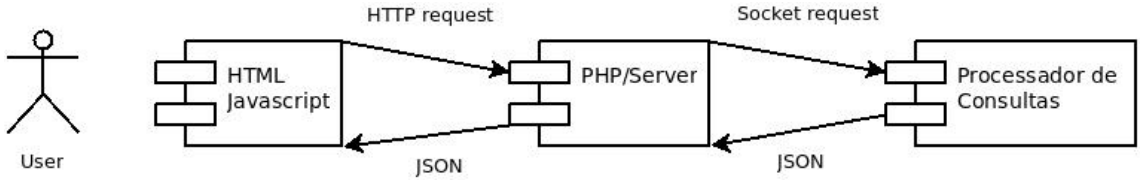


Figura 2. Esquema simplificado da comunicação entre a interface Web e o processador de consultas.

A interface Web está disponível em `http://greenwich.lbd.dcc.ufmg.br/lizardo/ri/`.

6. Implementação

Este trabalho foi implementado em *C++11*. Os arquivos com cabeçalho e códigos fontes estão organizados em pastas dentro do diretório *src*. Os três arquivos principais de execução dos programas estão na raiz desta pasta. Os arquivos de códigos da interface Web estão na pasta *www*.

As bibliotecas de terceiros necessárias para a compilação dos programas são fornecidas juntamente com o código fonte. Elas estão na pasta *lib* e seus cabeçalhos na pasta *include*. As bibliotecas são: Gumbo Parser, riCode e CMPH citadas anteriormente, e a biblioteca zlib[1].

6.1. Programas

Os seguintes programas estão disponíveis para execução:

- *index_writer_sorter*: Faz o *parsing* da coleção e cria o arquivo temporário de triplas ordenado.
- *index_query*: Constrói o índice comprimido e processa as consultas escritas em um arquivo passado como parâmetro da execução. Cada linha do arquivo é processada como uma consulta independente. Os resultados de cada consulta são salvos em arquivos separados.
- *index_server*: Constrói o índice comprimido e espera por requisições via *socket*. Cada requisição processada tem os resultados respondidos no formato JSON. Quando este programa é executado, ele primeiramente constrói o índice, que pode ser um processo demorado, para somente depois atender as requisições.

6.2. Compilação

Para compilar os programas e fazer o *link* as bibliotecas execute no Linux:

```
make all
```

6.3. Execução

Para executar um teste dos programas utilizando a coleção *toyExample*, execute no Linux:

```
make run
```

Para executar o *index_server*, execute:

```
make server
```

Para executar os programas individualmente, os seguintes argumentos devem ser passados:

```
./index_writer_sorter [-d -m -n -i -c]
./index_query [-d -t -q -n]
./index_server [-d -t -n]
```

onde as opções são,

-d: diretório de saída, onde os arquivos gerados pela execução serão criados.
-t: nome do arquivo temporário de triplas.

[1] zlib: http://www.zlib.net/

-m: tamanho da memória em MB, se não especificado, 1 MB será utilizado. O tamanho da memória durante a execução tem que ser o mesmo para todos os programas.
-n: número máximo de documentos a serem indexados, se não especificado, no máximo 1 milhão de documentos serão indexados.
-i: diretório da coleção comprimida de documentos.
-c: índice da coleção comprimida de documentos.
-q: caminho completo do arquivo contendo as consultas. Uma consulta por linha.

7. Avaliação Experimental

Os testes foram realizados no sistema operacional Ubuntu 13.10, rodando um notebook com processador 3rd Generation Intel® Core™ i7-3630QM (2.40GHz 6MB Cache), 8GB RAM, HHD 1 TB S-ATA (5,400 rpm). Foram feitos 5 execuções para cada teste e retirado a média. A coleção de documentos utilizada é a mesma do TP1.

7.1. Compressão

O método de compressão Elias γ foi utilizado na construção do índice invertido. Esse arquivo é formado por uma lista de pares $(documento, frequência)$. No trabalho, documento e frequência são inteiros positivos de 4 bytes cada. O algoritmo de compressão tem complexidade de tempo linear em relação ao tamanho da lista de pares. Para comprimir, foi calculado a diferença nos valores dos documentos em relação ao anterior. As figuras 3 e 4 comparam o índice, com e sem compressão, em relação ao tempo de construção e ao tamanho do arquivo final invertido.

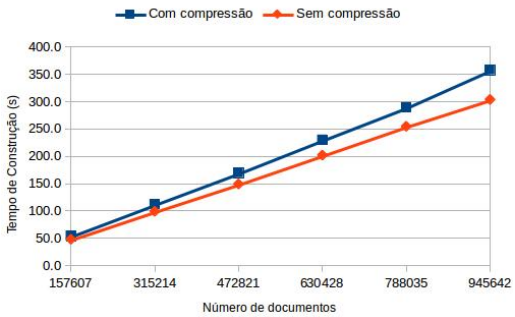


Figura 3. Tempo de construção do índice por número de documentos.

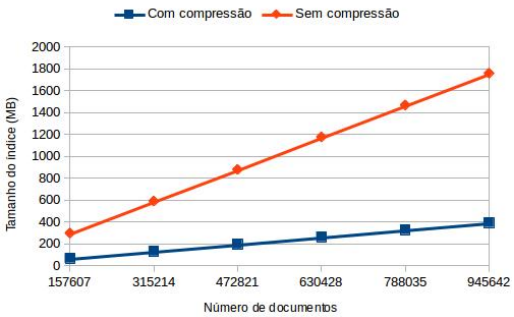


Figura 4. Tamanho do arquivo do índice invertido por número de documentos.

É possível notar nos gráficos que a construção do índice sem compressão é na média 13% mais rápida que a com compressão. No entanto, a compressão reduz o arquivo final invertido a aproximadamente 22% do tamanho do arquivo final sem compressão. Os resultados também mostraram que o tempo de compressão foi linear em relação ao número de documentos.

7.2. Consulta

Neste experimento é avaliado o tempo do processador de consulta utilizando os modelos BM25, Cosseno, Booleano e o Custom. Todos esses modelos são linearmente sensíveis

ao número de documentos encontrados como resposta, e ao número de termos pesquisados. O gráfico da Figura 5 mostra o desempenho dos modelos em relação ao tamanho da resposta encontrada para consultas de 1 termo. Os pontos representam o tempo de consulta e as linhas são as tendências. Neste experimento não é contabilizado o tempo de acesso ao arquivo contendo as informações dos documentos, mas somente o acesso ao índice e o ranqueamento dos documentos.

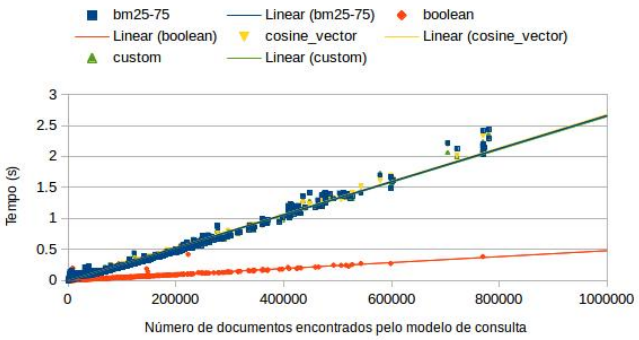


Figura 5. Tempo médio de consulta pelo tamanho da resposta.

Os resultados mostraram que o modelo Booleano é o mais rápido, o que é justificado pelo fato deste modelo não precisar contabilizar TF-IDF dos documentos encontrados. As operações realizadas por este modelo, união e interseção, são lineares no tamanho da menor lista da interseção. Os outros modelos são mais lentos, pois precisam calcular a similaridade de cada documento. Para ordenar os documentos, estes modelos, utilizam um *map*, com inserção de elementos em tempo linear.

No gráfico também é possível observar que o pior caso para um processador de consultas é quando é feita uma pesquisa por termo muito comum, como uma *stop-word* por exemplo. Consultas desse tipo levaria em média 3 segundos para ser processada por este programa e para uma coleção de 900 mil documentos.

7.3. Precisão x Revocação

Nestes experimentos os modelos foram avaliados utilizando 34 consultas pré-determinadas. Como métricas de avaliação foram utilizados curvas de precisão x revocação com a mesma interpolação definida em [Baeza-Yates et al. 1999].

7.3.1. Modelo Vetorial

O Modelo Vetorial foi testado com as seguintes normalizações:

- Normalização vetorial (*Cosine-vector*);
- Normalização pelo tamanho do documento em número de termos (*Cosine-terms*);
- Sem normalização. (*Cosine-no*).

O gráfico da Figura 6 apresenta a Precisão x Revocação média do modelo para as três normalizações testadas. Os melhores resultados foram obtidos pelo modelo com normalização igual a 1 (24,7% de precisão média). Seguido do modelo vetorial (13,4% de

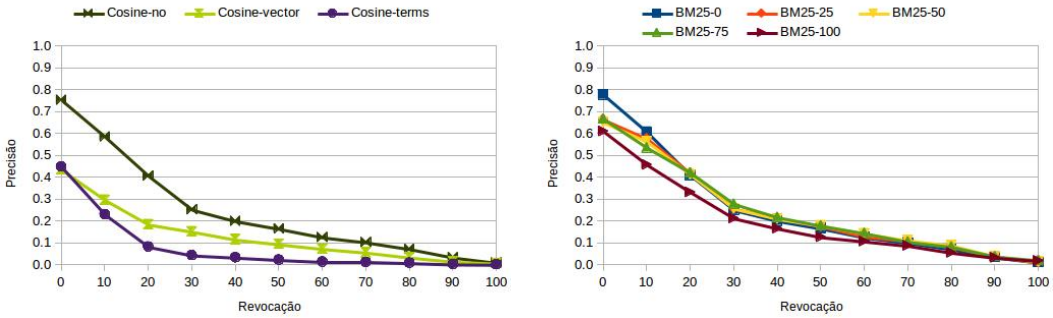


Figura 6. Precisão x Revocação média do Modelo Vetorial para normalizações diferentes. *Cosine-vector* apresenta normalização vetorial; *Cosine-terms* é normalizado pelo número de termos do documento; e *Cosine-no* não possuí normalização.

Figura 7. Precisão x Revocação média do Modelo BM25 para valores da constante b iguais a: $0,00$, $0,25$, $0,50$, $0,75$ e $1,00$.

precisão média). O modelo com normalização pelo número de termos apresentou precisão média de 8,2%.

7.3.2. Modelo BM25

O Modelo BM25 possui em sua função duas constantes K_1 e b. Neste trabalho, K_1 foi mantido igual a 1 e não foram experimentados outros valores. Já a constante b foi avaliada para os valores iguais a: 0.00, 0.25, 0.50, 0.75 e 1.00.

O gráfico da Figura 7 mostra a Precisão x Revocação média do BM25 para as 34 consultas e para valores diferentes da constante b. O BM25 com $b = 0$ (BM15) apresentou resultados ligeiramente superiores aos demais. Sua precisão média foi de 25,1%. Todos os demais apresentaram precisão média de abaixo de 24,6%.

7.3.3. Comparação dos Modelos

Neste experimento são comparados os quatro modelos (BM25, Cosine, Booleano e Custom) entre si. O gráfico da Figura 8 apresenta os resultados obtidos de Precisão x Revocação dos quatro modelos, com $b = 0,75$ para o Modelo BM25 e normalização vetorial para o Modelo Vetorial. O modelo Custom também foi testado com $b = 0$. Os resultados mostram que o BM25 apresentou resultados muito superiores aos demais, 24,4% de precisão média, contra 13,4% do Modelo Vetorial, 12,0% do Modelo Custom, e 2,0% do Modelo Booleano.

O gráfico da Figura 9 apresenta os resultados dos modelos com contantes $b = 0$ para o modelo BM25, e modelo vetorial sem normalização. Esses modelos foram os que apresentaram os melhores resultados, conforme explicado nas Seções 7.3.1 e 7.3.2. No gráfico é possível notar que os modelos BM25 e vetorial, com essas configurações, apresentaram resultados médios semelhantes.

As Figuras 10 e 11, 12 e 13, no final do documento, apresentam os gráficos

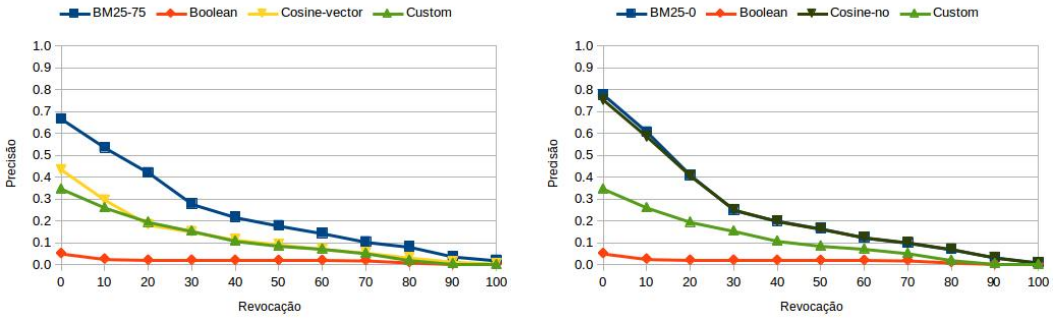


Figura 8. Precisão x Revocação média dos modelos BM25 (com $b = 0,75$), Cosseno (normalização vetorial), Booleano, e *Custom* (modelo vetorial adaptado com normalização baseada no BM25, dada por $(1 - b) + b * (doclen/avg\_doclen)$ e $b = 0,75$).

Figura 9. Precisão x Revocação média dos modelos BM25 (com $b = 0$), Cosseno (sem normalização), Booleano, e *Custom* (modelo vetorial adaptado com normalização baseada no BM25, dada por $(1-b)+b*(doclen/avg\_doclen)$ e $b = 0,75$).

de Precisão x Revocação para cada um dos 34 termos consultados durante os testes. A consulta pelo termo 'pânico' foi a que apresentou os melhores resultados.

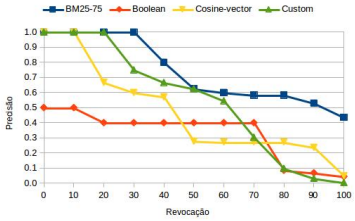
8. Conclusão

Neste trabalho foram implementados e avaliados quatro modelos de consultas: BM25, Vetorial, Booleano e o Custom, este desenvolvido a partir da combinação dos modelos BM25 e Vetorial. Também foi desenvolvido uma interface Web que permite aos usuários realizarem consultas utilizando os modelos implementados. Esta interface comunica com o processador de consulta por meio de *sockets*.

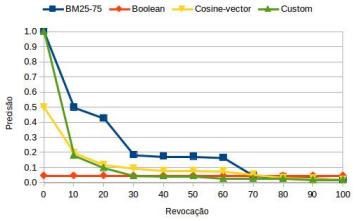
Uma avaliação experimental mostrou que o modelo BM25 com constante $b = 0$ apresentou os melhores resultados de precisão x revocação. O modelo vetorial, quando não utilizado normalização, também apresentou resultados satisfatórios. Também foi avaliado o desempenho dos algoritmos dos modelos, que apresentaram complexidade de tempo e espaço lineares.

Referências

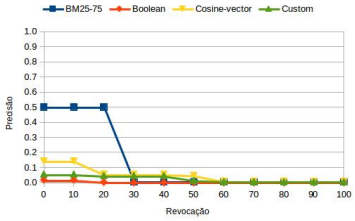
Baeza-Yates, R., Ribeiro-Neto, B., et al. (1999). *Modern information retrieval*, volume 463. ACM press New York.



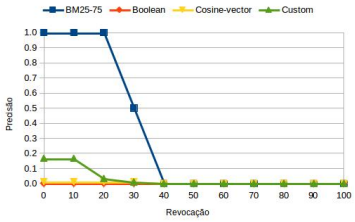
(a) ana maria braga



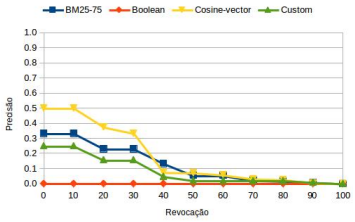
(b) baixaki



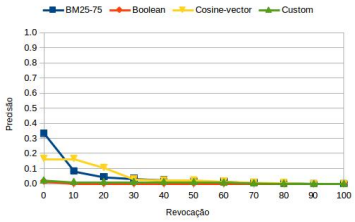
(c) caixa economica federal



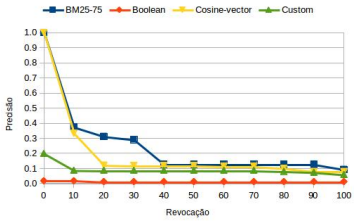
(d) casa e video



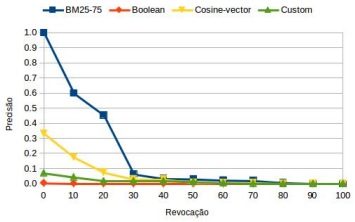
(e) claro



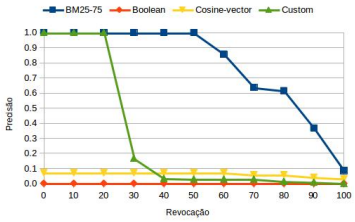
(f) concursos



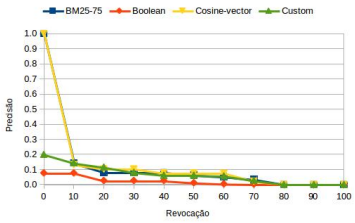
(g) detran



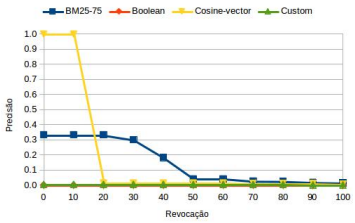
(h) esporte



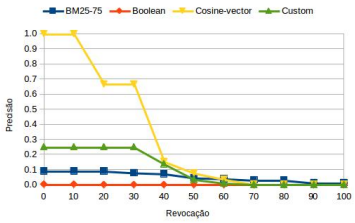
(i) frases de amor



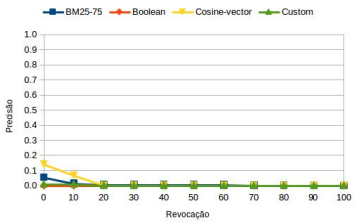
(j) funk



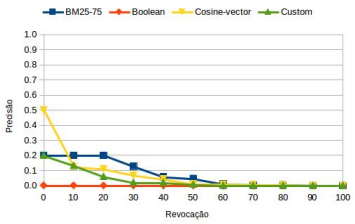
(k) globo



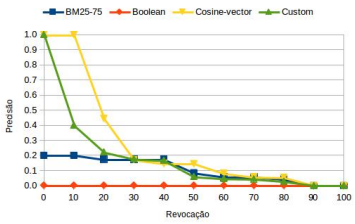
(l) gmail



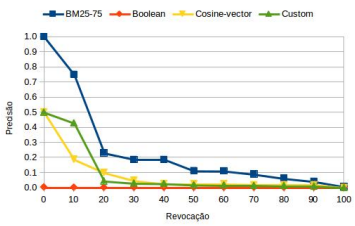
(m) google



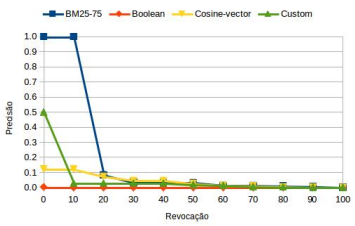
(n) hotmail



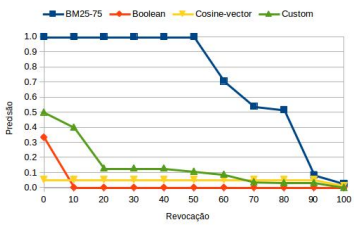
(o) ig



(p) jogos de meninas

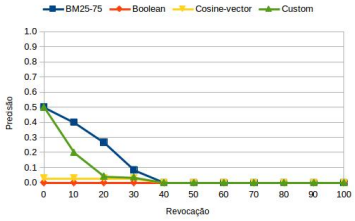


(q) jogos online

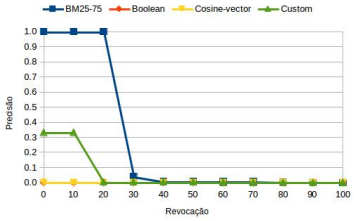


(r) mario

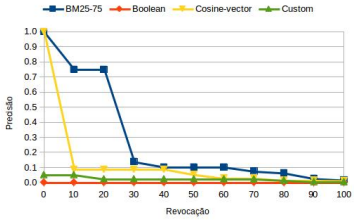
Figura 10. Gráficos de Precisão x Revocação.



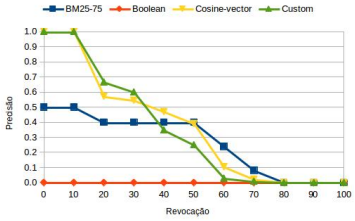
(a) mercado livre



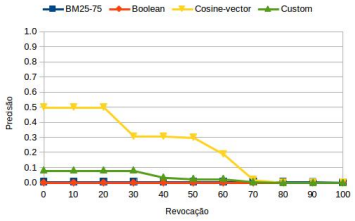
(b) msn



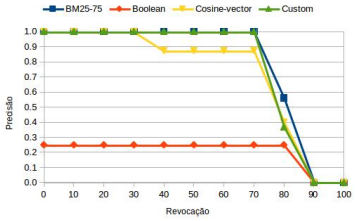
(c) naruto



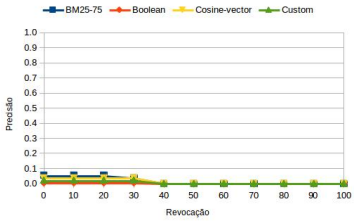
(d) oi



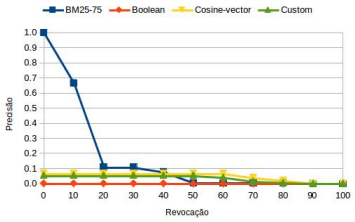
(e) orkut



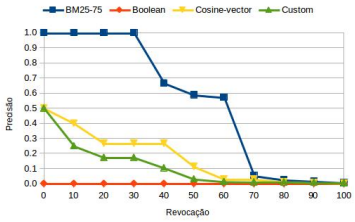
(f) panico



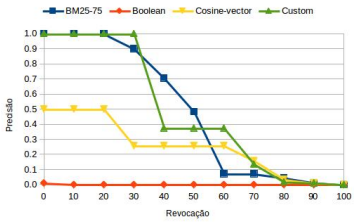
(g) poquer



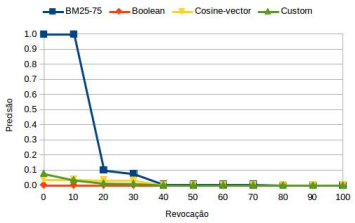
(h) previsao do tempo



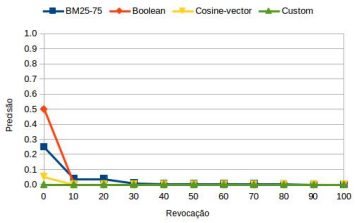
(i) receita federal



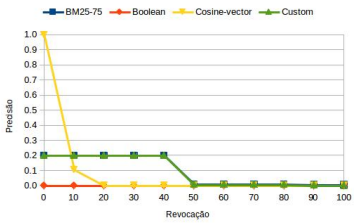
(j) record



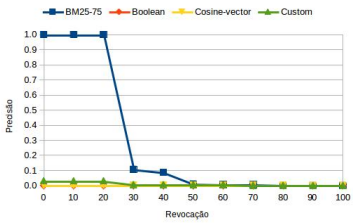
(k) rio de janeiro



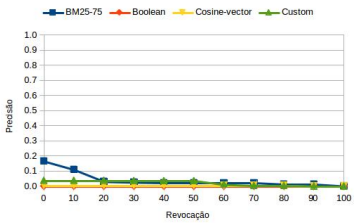
(l) terra



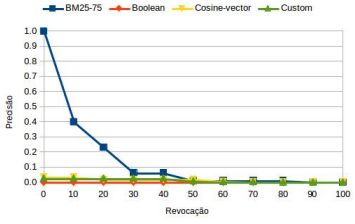
(m) uol



(n) vivo

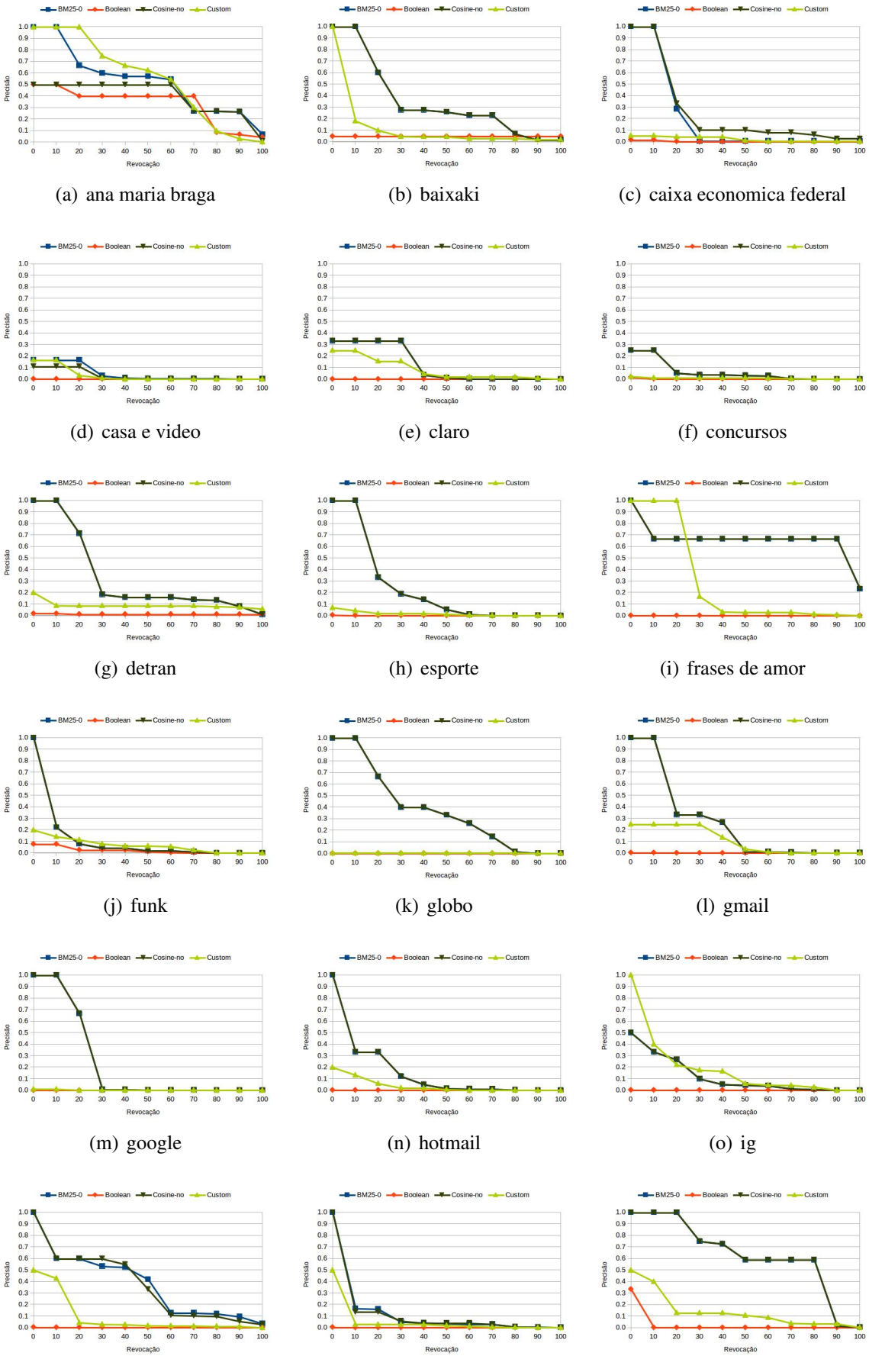


(o) yahoo



(p) youtube

Figura 11. Gráficos de Precisão x Revocação. (Continuação)



(a) ana maria braga (b) baixaki (c) caixa economica federal

(d) casa e video (e) claro (f) concursos

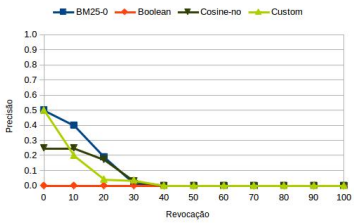
(g) detran (h) esporte (i) frases de amor

(j) funk (k) globo (l) gmail

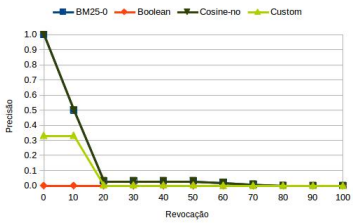
(m) google (n) hotmail (o) ig

(p) jogos de meninas (q) jogos online (r) mario

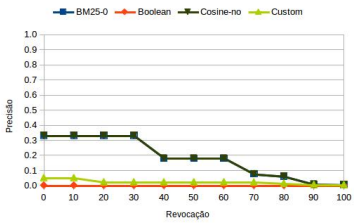
Figura 12. Gráficos de Precisão x Revocação com a melhor configuração dos modelos.



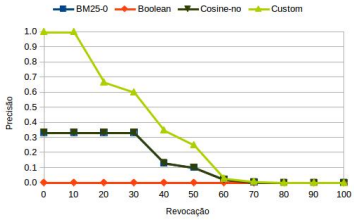
(a) mercado livre



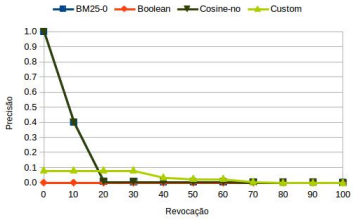
(b) msn



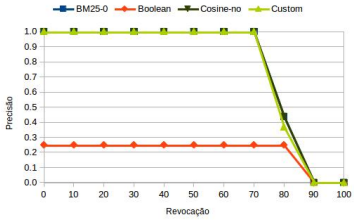
(c) naruto



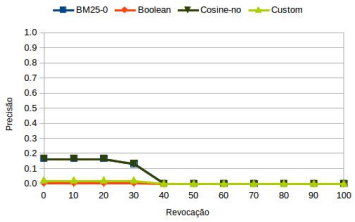
(d) oi



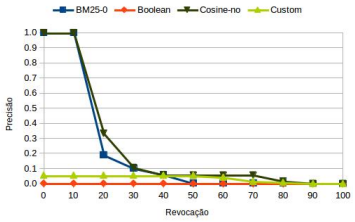
(e) orkut



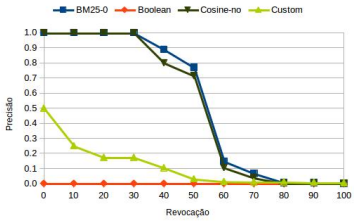
(f) panico



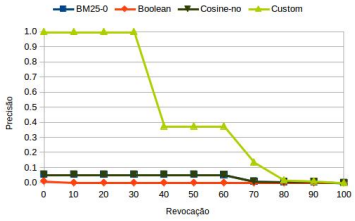
(g) poquer



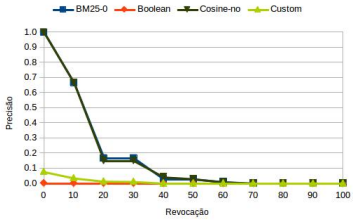
(h) previsao do tempo



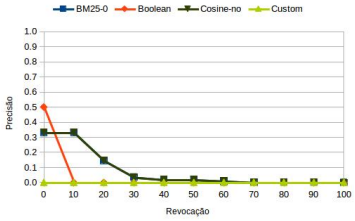
(i) receita federal



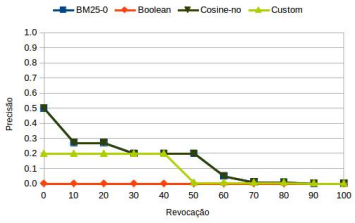
(j) record



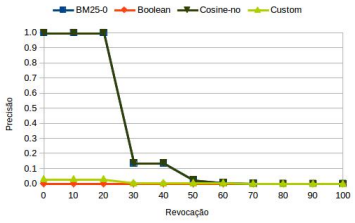
(k) rio de janeiro



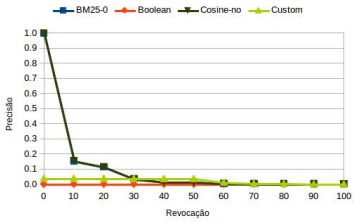
(l) terra



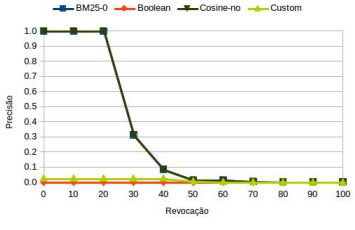
(m) uol



(n) vivo



(o) yahoo



(p) youtube

Figura 13. Gráficos de Precisão x Revocação com a melhor configuração dos modelos. (Continuação)