UNIVERSIDADE FEDERAL DO AMAZONAS
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
DEPARTAMENTO DE APOIO A PESQUISA
PROGRAMA INSTITUCIONAL DE INICIAÇÃO CIENTÍFICA

# TÉCNICAS DE PROCESSAMENTO DE CONSULTAS PARA SISTEMAS DE BUSCA EM TEXTO

Bolsista: Victor Costa de Oliveira, CNPq

MANAUS
2012

UNIVERSIDADE FEDERAL DO AMAZONAS
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
DEPARTAMENTO DE APOIO A PESQUISA
PROGRAMA INSTITUCIONAL DE INICIAÇÃO CIENTÍFICA

# RELATÓRIO FINAL
PIB-E/0098/2011
# TÉCNICAS DE PROCESSAMENTO DE CONSULTAS PARA SISTEMAS DE BUSCA EM TEXTO

Bolsista: Victor Costa de Oliveira, CNPq
Orientador: Prof$^{o}$. Dr. Edleno Silva de Moura

MANAUS
2012

# RESUMO

Máquinas de Busca para WEB operam bases de informações de tamanho gigantesco e precisam responder às consultas dos usuários ávidos por informações. Neste cenário, é comum cientistas e pesquisadores procurarem desenvolver técnicas para acelerar o processamento de consultas. Geralmente, usuários dessas máquinas não olham todos os documentos resultantes da sua consulta e isso faz com que possamos utilizar técnicas de poda para diminuir o conteúdo de informações da máquina de busca acelerando o processo. Outra forma de processar as consultas de maneira mais rápida e eficaz é utilizar um processador de duas camadas, onde se utiliza um índice diferente em cada uma das camadas. Nosso objetivo é fazer uma máquina de busca de duas camadas, onde a primeira utiliza amostras de entradas do índice completo para fazer um processamento simples e enxuto que seleciona candidatos à resposta para cada consulta, enquanto que a segunda camada faz um processamento mais robusto apenas nos candidatos selecionados na primeira etapa. Dessa forma, mostramos, experimentalmente, que é possível acelerar o processamento como um todo, bem como trazer respostas com qualidades praticamente iguais às qualidades se fosse feito o processo robusto em toda a base de dados.

SUMARIO

# 1. INTRODUÇÃO

As máquinas de busca para a Web operam uma base de dados muito grande [7] e ocupam um cenário em que usuários precisam buscar cada vez mais essas informações. Além de operar grandes quantidades de páginas, as máquinas de busca também devem responder a centenas - ou até milhares – de consultas por segundo sendo necessário processá-las com uma grande rapidez.

Máquinas de Busca utilizam um conceito chamado *lista invertida*. *Lista invertida* é uma estrutura que fornece informações sobre a localidade dos termos nos documentos, bem como o seu impacto em cada um destes. Como esta estrutura armazena muitas informações, as máquinas de busca utilizam algumas técnicas para reduzir este espaço de armazenamento. Uma das técnicas é utilizar *métodos de poda* que consistem em descartar entradas de termos na lista invertida durante a indexação (poda estática) ou em tempo de processamento (poda dinâmica).

Trabalhos recentes [5] indicam que aproximadamente 80% dos usuários visualizam apenas os primeiros documentos de uma consulta. Assim, utilizar métodos de poda responde às consultas, diminui o espaço de armazenamento e acelera o processamento. No entanto, alguns documentos considerados importantes podem não estar na lista invertida depois da poda e, consequentemente, não estará na resposta, comprometendo, dessa forma, o desempenho da máquina de busca. Neste contexto, é necessário saber quais documentos devem estar presentes na lista invertida antes de podá-la para que tenhamos um aumento na eficiência sem, contudo, ter perda de qualidade.

Existem duas maneiras padrões de selecionar as respostas para as consultas na literatura [5]: o processo Documento a Documento (DAAT, do inglês *Document at a time)* e o processo Termo a Termo (TAAT, do inglês Term at a Time). Processando no modo DAAT, avalia-se a contribuição de todos os termos da consulta com um único documento antes de processar o próximo documento. Enquanto que no TAAT, processam-se os termos da consulta um por vez e acumula-se a pontuação parcial dos termos em cada documento à medida que os termos são computados.

Tendo em vista as considerações delineadas acima, propõe-se, neste sentido, criar um sistema de busca de duas camadas, onde o objetivo é acelerar o processamento de consultas obtendo uma lista de candidatos para a resposta de uma consulta usando o processo *TAAT* com índices podados na primeira camada e, depois, na segunda camada, processar o modo *DAAT* apenas nesta lista de candidatos, fazendo com que este processo tenha uma maior velocidade, quando comparado com os demais processadores encontrados na literatura.

Estudaremos, portanto, algumas técnicas para escolher o melhor conjunto de candidatos à resposta das consultas utilizando técnicas de poda estática e um valor de impacto unificado proposto em [4] em que o impacto é obtido a partir de uma função gerada por programação genética. O objetivo é saber se, utilizando este valor de impacto, podemos obter o conjunto de candidatos com o menor índice de poda possível.

## 2. REVISÃO BIBLIOGRÁFICA

Com o aumento do número de informações na WEB, bem como o número de usuários que buscam por estas informações, as máquinas de busca precisam processar muitas consultas ao mesmo tempo e fornecer aos usuários respostas rápidas e precisas. Neste cenário, muitos trabalhos estão surgindo para acelerar este processo. Técnicas de poda para não ter que processar as listas completas e máquinas de busca que dividem o processamento em várias fases estão surgindo como forma de se processar consulta de maneira eficiente.

O trabalho de Alexandros Ntoulas [Ntoulas and Cho. Pruning policies for two-tiered inverted index with correctness guarantee. [SIGIR '07 (2007), pp. 191-198.] será um dos trabalhos que vamos ter como comparação dos resultados. Ele propôs uma política de poda em duas camadas com garantia de exatidão, onde cria um algoritmo que processa a consulta com o índice podado caso esta resposta seja a mesma que a resposta (mesmos documentos e mesma ordem) usando toda a base, mas quando estas respostas não são as mesmas, utiliza toda a base para processar a consulta.

Nosso projeto diferencia-se do proposto por Ntoulas, pois este processa a segunda etapa apenas se a primeira não der resultados satisfatórios, enquanto que, no projeto que está sendo proposto, as duas etapas são necessárias e a primeira acelera o processamento da segunda. Nossa hipótese é que esta mudança sutil pode resultar em ganhos de *performance* sobre o método proposto por Ntoulas, que representa hoje o estado-da-arte em termos de processamento eficiente de consultas em máquinas de busca.

Outro trabalho que levaremos em conta quanto à comparação de resultados é o proposto por Ding e Suel [5], onde se utiliza um conceito de *Block-Max* para saltar mais documentos utilizando o processo DAAT. O autor afirma que utilizar o processador TAAT custa muito caro quando comparado com o DAAT, mas nosso desafio é provar que gerando os candidatos usando um processo TAAT com impacto unificado e depois processando apenas os candidatos com DAAT, o processo como um todo se torna bem mais rápido e preciso, pois a primeira etapa retorna documentos relevantes para a segunda etapa utilizando apenas um pequeno índice da lista invertida.

O trabalho de Broder [3] utiliza também um processador de duas camadas, cujo objetivo é diminuir o tempo de resposta, gerando candidatos na primeira etapa com o número mínimo de *features* possíveis para que na segunda etapa restem apenas alguns documentos para serem avaliados por completo e ter o *score* computado. Em nosso trabalho, utilizamos os algoritmos do LePref [4] para gerar funções que geram o *score* final já com todas as *features* combinadas. Mostramos experimentalmente que a utilização desta combinação permite podar mais e com isso ter um ganho na eficiência.

Em nosso trabalho, temos que usar técnicas que possam acelerar o processamento sem, contudo, ter perda na qualidade das respostas. Comparamos, portanto, o índice de poda que devemos ter para selecionar os candidatos na primeira etapa do processo utilizando o UTI com o índice necessário para obter os mesmos candidatos utilizando o BM25 e o TF-IDF.

Mostraremos que podemos podar as listas invertidas cerca de 10 vezes mais utilizando o modelo de impacto unificado do que utilizando as outras técnicas da literatura.

Trabalhos recentes sobre Métodos de Poda baseado em localidade [6, 8 e 9], revelam que podemos obter documentos relevantes ao podar listas em que estas tem informações sobre a localidade do termo no documento. Quando duas palavras estão em um mesmo parágrafo [8], estas, geralmente, estão muito relacionadas e, assim, podemos fazer a poda incluindo documentos em que os termos aparecem num mesmo parágrafo.

Para nosso trabalho, utilizamos uma poda estática em que processamos as listas invertidas das palavras até certo valor de k% da lista e assim analisamos o percentual necessário para obter os documentos candidatos à resposta. Utilizar outras técnicas de poda estática e até poda dinâmica faria o sistema mais robusto e provavelmente traria melhores resultados, sendo, portanto, um desafio a ser cumprido em trabalhos futuros.

# 3. MÉTODOS UTILIZADOS

Quando uma consulta possui mais de uma palavra, podemos destacar dois principais métodos de escolher os documentos que irão compor a resposta. Esses métodos vieram da álgebra booleana e são conhecidos como ‘OU’ e ‘E’. Considerando uma consulta com dois termos, podemos entender que, no método ‘OU’, os documentos que comporão a resposta estarão ou na lista invertida da primeira palavra ou na lista da segunda (subtende-se que os documentos que estarão nas duas listas, ao mesmo tempo, também estarão na resposta). O método ‘E’ indica que apenas os documentos que estão nas duas listas invertidas são os que comporão a resposta. Geralmente utilizamos o método ‘OU’ para processamento *term-at-a-time* e o método ‘E’ para *document-at-a-time*.

Para ter a lista de candidatos à resposta de uma consulta, foi feito um processador de consultas no modo TAAT. As listas invertidas foram salvas em ordem decrescente pelo impacto. Assim, fizemos um processo de poda estática para obter os k% da lista de cada palavra da consulta, onde k é o percentual de poda. Avaliamos o valor de k necessário para obter todos os documentos candidatos que são considerados relevantes na resposta utilizando o índice unificado (UTI) e os demais índices vistos na literatura, como, por exemplo, o BM25 e o TF-IDF.

Esta primeira etapa deve gerar os candidatos de maneira rápida, pois fazemos ainda, na segunda etapa, um processo DAAT com os candidatos gerados na primeira etapa. Para esta segunda etapa, foi considerado o trabalho de Ding e Suel proposto em [5], onde utiliza-se um processo WAND (proposto inicialmente por Broder at all[3]) com o conceito de *Block-Max*. O processo WAND combina o processo OR e o processo AND utilizando a técnica DAAT. Este processo analisa se somatório dos impactos dos termos de uma consulta para um documento alcança um limite mínimo. Se alcançar, este documento estará na resposta. O *Block-Max* diz respeito a esse limite mínimo, que é calculado para cada bloco de documentos em que se faz a descompressão dos documentos da lista invertida. O resultado é aceleração do processador.

Para saber se os documentos candidatos à resposta da consulta encontrados na primeira etapa são satisfatórios, comparamos estes documentos com os 20 primeiros documentos que fariam parte da resposta se fosse usado o processador que utilizasse as listas invertidas das palavras da consulta sem poda (índice completo). Assim, garantimos que, se recuperarmos uma boa parte desses 20 documentos de forma rápida e utilizarmos o processador padrão na segunda etapa do processo apenas nesses candidatos, o processador ficará mais rápido que o processador padrão, pois não perderemos tempo processando documentos não relevantes para a consulta.

A ideia de usar apenas 20 documentos como comparação surgiu pelo fato de que o usuário de uma máquina de busca geralmente só vê os primeiros resultados em uma consulta, ou seja, dificilmente visualiza os documentos após a vigésima posição [5] e quando não acha o documento desejado, ele muda a sua consulta.

# 4. RESULTADOS E DISCUSSÕES

Os experimentos dizem respeito à seleção dos candidatos, onde fizemos testes de eficiência e eficácia em que avaliamos se os candidatos gerados estão sendo satisfatórios. Utilizamos para os experimentos duas bases: a GOV2 e a WT10G. Estas bases têm tamanhos diferentes e isso nos ajuda a provar que a seleção de candidatos com o índice unificado (UTI) é confiável para qualquer tamanho. A base Gov2 possui 25 milhões de documentos enquanto que a WT10g possui apenas 2 milhões de documentos.

Para os experimentos a seguir, utilizamos um processador que utiliza o TAAT para processar as podas de k%, a fim de obter os candidatos, enquanto que para o gabarito, utilizamos o processador DAAT com o índice completo. Assim, o experimento foi verificar se conseguimos recuperar todos os documentos que são importantes para a consulta como candidatos. Utilizamos um conjunto de 10000 consultas para a GOV2 e WT10G.

Utilizamos os níveis de k em 5%, 2%, 1% e 0,5% para podar os índices de cada palavra da consulta e assim calculamos o tempo de processamento, bem como a porcentagem de consultas que tiveram todos os documentos recuperados. Além desses quatro níveis de poda, modificamos o algoritmo para que, se acharmos uma palavra cuja lista invertida seja muito grande (maior que um milhão de entradas), processamos apenas as N primeiras entradas da sua lista invertida. Consideramos para nossos experimentos, N = 1000 entradas.

## 4.1 Base GOV2

A base da *GOV2* possui mais de 25 milhões de documentos e, para a mesma, avaliamos 10000 consultas onde obtemos a tabela 1.

A tabela 1 relaciona cada processo com o tempo de execução, a porcentagem de consultas que recuperaram todos os seus documentos e o número de consultas que não tiveram êxito (não recuperaram todos os documentos). Podemos perceber, assim, que gerando os candidatos, com o índice do UTI, temos uma qualidade melhor dos resultados.

Percebemos, pela tabela 1, que em 2% do índice UTI apenas 2 (duas) consultas não tiveram todos os seus documentos gerados, enquanto que no índice TF-IDF, 20 consultas não tiveram êxito. Essa proporção se repete para as outras porcentagens dos índices, como, por exemplo, 1%, onde no UTI tivemos 10 consultas sem sucesso, enquanto que no TF-IDF 101 consultas tiveram falhas. Isso nos leva a comprovar que processar os candidatos utilizando o índice unificado tem 10 vezes menos perdas que processando no outro índice.

| | UTI | | | Whole Document (TF-IDF) | | |
|---|---|---|---|---|---|---|
| Processo (k%) | Tempo (s) | % Consultas c/ Sucesso | Qtd. Consultas com falhas | Tempo (s) | % Consultas c/ Sucesso | Qtd. Consultas com Falhas |
| 100% (Gabarito) | 1812 | GABARITO | GABARITO | 2277 | GABARITO | GABARITO |
| 5% | 2719 | 100,00% | 0 | 3469 | 99,989% | 1 |
| 2% | 1337 | 99,98% | 2 | 1215 | 99,8% | 20 |
| 1% | 628 | 99,90% | 10 | 573 | 98,98% | 101 |
| 0.5% | 252 | 99,51% | 49 | 223 | 96,18% | 382 |
| 5% (Mod) | 400 | 99,33% | 67 | 376 | 97,82% | 218 |
| 2% (Mod) | 115 | 99,00% | 100 | 118 | 96,17% | 383 |
| 1% (Mod) | 62 | 98,56% | 144 | 61 | 93.15% | 685 |
| 0.5% (Mod) | 37 | 97,80% | 220 | 37 | 88.75% | 1125 |

Tabela 1: Relacionando os Processos, tempos de execução das 10000 consultas, porcentagem das consultas que retornaram todos os documentos e a quantidade de consultas que não obtiveram êxito (que não recuperaram todos os documentos).

O gráfico da figura 1 mostra a relação tempo e qualidade avaliando as 10000 consultas, resumindo a tabela 1. As linhas vermelhas mostram o tempo em segundos e a qualidade do processador com o índice UTI, enquanto que as verdes dizem respeito ao tempo e qualidade da resposta com o índice TF-IDF. Assim, podemos destacar ainda que o UTI tem um tempo de processamento menor que o índice TD-IDF e ainda assim, possui uma qualidade melhor.

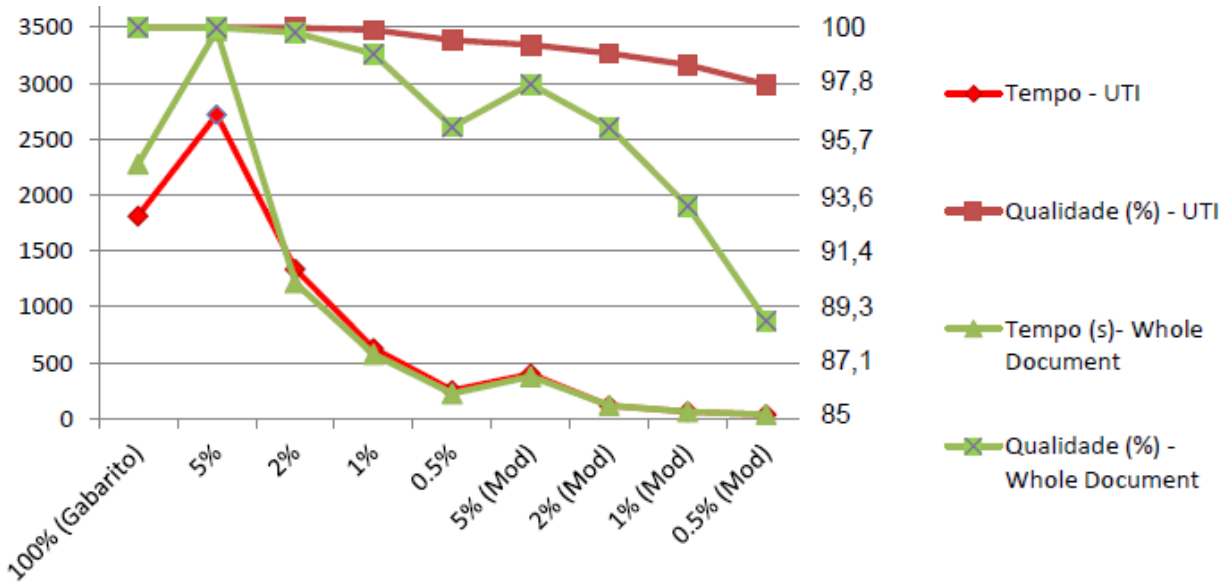


Figura 1 – Gráfico relacionando os processos com o tempo e qualidade de cada um para o índice UTI e TF-IDF.

Outro teste que fizemos na base GOV2 foi calcular a quantidade de documentos recuperados em cada posição. Ou seja, quando comparamos o resultado olhando o índice por inteiro com o resultado com a poda, procuramos descobrir quantas consultas conseguiram ter todos os seus documentos recuperados.

A figura 2 nos mostra a cobertura em cada posição processando 5%, 2%, 1% e 0,5% das listas de cada termo.

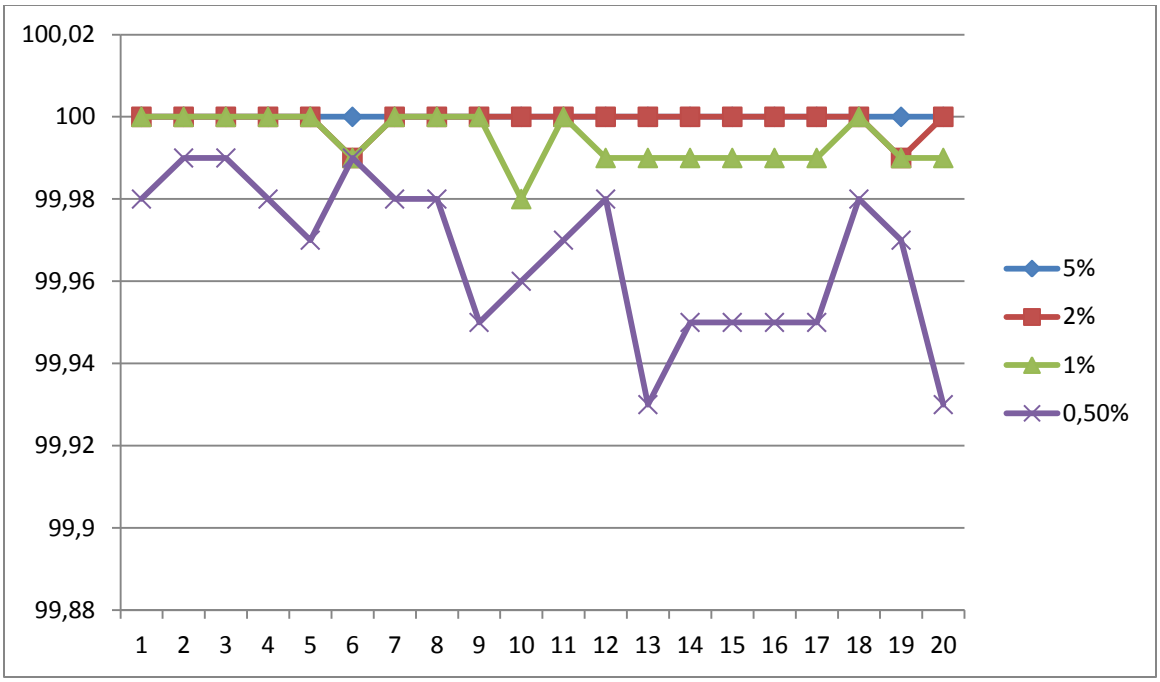


Figura 2: Quantidade de Consultas que obtiveram êxitos para cada posição (1 ~ 20) em diferentes níveis de poda.

Analisando a figura 2, percebemos que a poda de 5% retornou todos os documentos de todas as consultas avaliadas, mas o tempo de processamento é superior ao tempo do processo utilizando 100% do índice. O processador que utilizamos para gerar o índice em 5% é o processador TAAT que nos dá os melhores resultados no topo das listas invertidas, no entanto é mais lento, o que justifica o tempo ser maior que o tempo utilizando o processador DAAT.

Podemos também perceber que 1% do índice é uma ótima porcentagem para gerar os candidatos, pois ele demora 1/3 do tempo do processador completo e nos proporciona que, em média, 99,99% das consultas conseguem ter todos os documentos que a consulta exigiria ter se fosse processado no índice completo. Se fôssemos ver o top10, esse resultado seria ainda maior, onde praticamente 100% das consultas teriam encontrado os candidatos certos.

Vimos, portanto, que processar utilizando UTI na Gov2 nos dá uma melhor qualidade e menor tempo de processamento ao gerar candidatos à resposta para as consultas. Um trabalho futuro seria calcular o tempo total, gerando os candidatos na primeira fase e processando a segunda parte para obter os resultados finais.

## 4.2 Base WT10G

A base WT10G possui quase dois milhões de documentos e, por isso, é considerada uma base pequena para a avaliação de consultas. Decidimos avaliar a nosso processador nesta base, pois ela nos permite mostrar que a utilização do índice unificado para geração de candidatos

obtém resultados melhores do que utilizando os índices tradicionais da literatura mesmo com poucas páginas para fazer o treino das fórmulas pela programação genética.

A tabela 2 mostra o mesmo estudo feito para a base da GOV2, onde comparamos o tempo de execução e a qualidade utilizando o UTI e TF-IDF (Whole Document).

| | UTI | | | Whole Document | | |
|---|---|---|---|---|---|---|
| Nível de Poda | Tempo (s) | Qtd Doc não Recuperados. | Consultas c/ Sucesso | Tempo (s) | Qtd Doc não Recuperados. | Consultas c/ Sucesso |
| 100% | 216 | GABARITO | | 403 | GABARITO | |
| 5% | 26 | 0 | 100% | 25 | 534 | 98,80% |
| 2% | 13 | 0 | 100% | 14 | 544 | 98,77% |
| 1% | 10 | 0 | 100% | 10 | 544 | 98,77% |
| 0,5% | 8 | 0 | 100% | 9 | 544 | 98,77% |

Tabela 2: Relacionando os Processos, tempos de execução das 10000 consultas, a quantidade de documentos não recuperados e a porcentagem das consultas que retornaram todos os documentos.

Na tabela 2, percebemos que, pelo fato da base ser pequena, conseguimos recuperar todos os documentos mesmo com um nível de poda baixo quando usando o índice unificado. Já utilizando o *Whole Document*, percebemos que se tem uma perda de qualidade apesar de o tempo ser o mesmo para processar a primeira etapa nos dois índices. Podemos concluir, portanto, que o UTI é melhor em termos de qualidade quando a base que estamos analisando é pequena.

## 5. CONCLUSÃO

Tendo em vista que o número de usuários que utilizam máquinas de busca na WEB vem crescendo e dependem, no seu dia-a-dia, cada vez mais da utilização das mesmas, um processador rápido em uma máquina de busca é de extrema importância nos dias atuais. Além de ter que processar rápido, as máquinas de busca também devem trazer respostas relevantes para o usuário tornando o processo como um todo muito complexo de ser desenvolvido. O trabalho que foi desenvolvido trouxe resultados interessantes para o campo de Recuperação de Informação, pois conseguimos desenvolver um processador em que, quando comparado com os processadores vistos na literatura, possui um ganho na velocidade de processamento e também vimos que a qualidade nos resultados obtidos por esse processador é a praticamente a mesma obtida por processadores que levam em conta todas as entradas dos índices.

O aumento da velocidade do processador foi obtido ao separar o processo em duas etapas, onde a primeira etapa selecionou os candidatos a resposta para cada consulta e a segunda fez o processamento mais robusto e completo apenas nestes candidatos. O índice unificado nos ajudou na hora de podar as listas invertidas no processo da primeira etapa, pois quanto mais a poda aumenta, mais rápido se torna o processo devido a menor quantidade de documentos a serem processados. Concluímos que processar os candidatos utilizando o índice unificado é mais rápido e tem resultados mais precisos quando comparado com outros índices.

Os resultados obtidos no trabalho têm um grande significado para usuários de máquinas de busca, pois se consegue aumentar a velocidade do processamento de consultas e, assim, mais consultas podem ser processadas ao mesmo tempo. Nossos desafios futuros é fazer com que possamos ter um novo método para podar índices na primeira etapa do processador, e fazer esta etapa ficar mais rápida e robusta a diferentes consultas.

## 6. REFERÊNCIAS BIBLIOGRÁFICAS


[1] Ricardo A. Baeza-Yates, Vanessa Murdock, Claudia Hauff: *Efficiency trade-offs in two-tier web search systems*. SIGIR 2009: pp. 163-170. {B}

[2] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press Addison-Wesley, 1999. {B}

[3] Andrei Z. Broder, David Carmel, Michael Herscovici, Aya Soffer, Jason Zien. *Efficient Query Evaluation using a Two-Level Retrieval Process*. In *Proceedings of the 27th IEEE International Conference on Data Engineering (ICDE)*, 2011, pp 426-434. {B}

[4] Carvalho, André; de Moura, Edleno Silva; Rossi, Cristian; Silva, Altigran Soares da; Fernandes, David . *LePrEF: Learn to Pre-compute Evidence Fusion for Efficient Query Evaluation*. Journal of the American Society for Information Science and Technology, 2012. [Online]. {C}

[5] Shuai Ding, Torsten Suel. Faster Top-k Document Retrieval Using Block-Max Indexes*. SIGIR'11,* July 24–28, 2011, Beijing, China. {D}

[6] Claudia Hauff: *Predicting the effectiveness of queries and retrieval systems*. SIGIR 2010 PhD Thesis. {H}

[7] A. Gulli and A. Signorini. *The indexable web is more than 11.5 billion pages*. In *WWW*, 2005, pp. 902 e 903. {G}

[8] E. S. De Moura, C. F. Dos Santos, B. S. De Araujo, and A. S. Da Silva. *Locality-Based Pruning Methods for Web Search*. In ACM Transactions on Information Systems (TOIS). Vol. V, N$^{o}$ N, April 2007, Pages 1–32. {M}

[9] Edleno Silva de Moura, Célia Francisca dos Santos, Daniel R. Fernandes, Altigran Soares da Silva, Pável Calado, Mario A. Nascimento: *Improving Web search efficiency via a locality based static pruning method*. World Wide Web Conference – WWW 2005: 235-244. {M}

[10] Ntoulas and Cho. *Pruning policies for two-tiered inverted index with correctness guarantee.* SIGIR '07 (2007), pp. 191-198. {N}

[11] Aya Soffer, David Carmel, Doron Cohen, Ronald Fagin, Eitan Farchi, Michael Herscovici, Yoëlle S. Maarek: *Static Index Pruning for Information Retrieval Systems*. SIGIR 2001: 43-50. {S}

[12] Yohannes Tsegay, Andrew Turpin, Justin Zobel: *Dynamic index pruning for effective caching.* Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management – CIKM 2007: 987-990. {T}

7. CRONOGRAMA

| | Ago/11 | Set/11 | Out/11 | Nov/11 | Dez/11 | Jan/12 | Fev/12 | Mar/12 | Abr/12 | Mai/12 | Jun/12 | Jul/12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Revisão Bibliográfica** | X | X | | | | X | X | | | X | X | |
| **Disciplina RI** | X | X | X | X | X | | | | | | | |
| **Implementação** | X | X | X | X | X | X | | | | | | |
| **Avaliação & Experimentos** | | | | X | X | X | X | | X | X | X | |
| **Nova Implementação** | | | | | | X | X | X | X | | | |
| **Testes Finais** | | | | | | | | X | X | X | | |
| **Relatório Final** | | | | | | | | | | | X | X |
| **Reunião com o Orientador** | X | X | X | X | X | X | X | X | X | X | X | X |