Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uma Abordagem de Componentes Combinados para a Geração de Funções de Ordenação usando Programação Genética

Humberto Mossri de Almeida

Belo Horizonte
Junho 2007

Humberto Mossri de Almeida

Uma Abordagem de Componentes Combinados para a Geração de Funções de Ordenação usando Programação Genética

Dissertação apresentada ao Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Belo Horizonte
Junho 2007

Resumo

Com o advento da Web e de outros repositórios de informação, como Bibliotecas Digitais, a tarefa de recuperação de informação transformou-se em um problema extremamente complexo e desafiador. Neste contexto, as máquinas de busca surgiram como ferramentas fundamentais para a tarefa de recuperação de informação em uma coleção de documentos. Estas ferramentas são baseadas em modelos de recuperação de informação, cujo principal objetivo é definir a ordem na qual os documentos são retornados para os usuários em resposta a uma consulta, através de uma função de ordenação. Diversas funções de ordenação têm sido investigadas ao longo dos anos. No entanto, a maioria delas tem um caráter geral, isto é, são projetadas para serem efetivas em qualquer coleção.

Neste trabalho é proposto um novo método para descobrir funções de ordenação adaptadas a uma coleção baseado em Programação Genética (GP). O processo evolutivo da Abordagem de Componentes Combinados (CCA), proposta por este trabalho, diferentemente de outras abordagens baseadas em GP, utiliza componentes de diferentes funções de ordenação comprovadamente eficazes e conhecidas da literatura de recuperação de informação. Parte-se da hipótese de que estes componentes são individualmente representativos e ricos de significado e podem ser combinados para a geração de uma nova função de ordenação mais efetiva e específica para uma determinada coleção.

Os resultados experimentais mostram que a abordagem CCA foi capaz de superar em até 40% as abordagens clássicas da literatura tais como tf-idf, BM25 e outra abordagem baseada em GP (denominada FAN-GP) em duas coleções diferentes. O processo evolutivo CCA também foi capaz de reduzir o problema do "treinamento exagerado", geralmente encontrado em métodos de aprendizado de máquina, especialmente programação genética.

Palavras-Chave: Recuperação de Informação, Funções de Ordenação, Ponderação de Termos, Programação Genética, Aprendizado de Máquina.

Abstract

Due to the advent of the Web and other textual repositories, such as digital libraries, the information retrieval task has become a very complex and challenging problem. In this context, search engines became valuable tools for the information retrieval task in document collections. These tools are based on information retrieval models whose main goal is to produce, given a query, a set of documents ranked by relevance as an answer. For doing so, the so-called ranking functions are employed. Several ranking functions have been investigated throughout the years. However, most of them attempt to be very general in nature, i.e., they were designed to be effective in any type of collection.

In this work, we propose a new method to discover collection-adapted ranking functions based on Genetic Programming (GP). The evolution process of our *Combined Component Approach* (CCA), differently from other approaches based on GP, uses several components extracted from effective and well-known ranking functions. Our assumption is that these components are representative and meaningful and can be combined for generating a more effective and specific new ranking function for a given document collection.

Experimental results show that our approach was able to outperform in more than 40% standard TF-IDF, BM25 and other GP-based approach (named FAN-GP) in two different collections. The CCA evolution process also was able to reduce the overtraining, commonly found in machine learning methods, especially genetic programming.

Keywords: Information Retrieval, Ranking Functions, Term-weighting, Genetic Programming, Machine Learning.

Dedico este trabalho a meus pais por tudo o que fizeram por mim e, em especial, à Fátima e Clara, meus grandes amores.

Agradecimentos

Primeiro de tudo, agradeço a Deus por permitir que este trabalho pudesse ser realizado e pela força dada nos momentos mais difíceis desta caminhada, renovando minha fé para que eu pudesse seguir adiante com dedicação e perseverança.

Agradeço de todo o meu coração a meus pais, Lima e Eridam, por não terem medido esforços para a minha educação, pelos exemplos e pelo dom da vida. Fisicamente não podem compartilhar esta vitória, mas sinto a emoção e o abraço carinhoso, assim como a presença em pensamento e espírito ao longo da eternidade.

Serei eternamente grato ao meu grande amor, Fátima, por compartilhar todos os momentos da minha vida, tornando-os mais alegres. Obrigado por sempre incentivar as minhas jornadas, por compreender as noites e finais de semana sem a minha companhia e por, juntamente com a Clarinha, encher meu coração de amor. Agradeço à Clara, ainda tão pequena, por renovar as minhas forças para dar-lhe motivos de exemplo e orgulho. Seria ingratidão não citar a querida Nina, que me fez conciliar o trabalho com uma brincadeira e outra, e aos meus pés se deitava para que não me sentisse só durante as horas de estudo.

Aos meus irmãos, obrigado pelo amor incondicional e por me fazerem sentir especial. Aos demais familiares — tios e tias, primos e primas, sobrinhos e sobrinhas, cunhados e cunhadas, agradeço por todo carinho e atenção. Em especial, agradeço ao Tio Antônio, Marlene, Tia Edméa, Tia Graciette, Tia Maria, Tio Pedro e ao saudoso Tio Flamarion, pelo carinho de toda vida.

Agradeço aos meus sogros, Wellington e Marina, por me acolherem como membro da família e por também me incentivarem a ingressar na vida acadêmica.

Agradeço aos meus amigos que compartilham comigo todos os momentos da minha vida. Agradeço ao Marcelo Nassau, pela amizade diária e por tornar a rotina de trabalho mais suave. Roger, agradeço o apoio e por preencher de alegria nossa amizade. Wlad, obrigado pelo incentivo e pelo carinho comigo. Cícero, obrigado pelo apoio de sempre e pelas orações; Guiga, Evands e Gusta, obrigado pela amizade. Agradeço aos amigos-primos Marcelo, Paula, Jederson, Patrícia, Emerson e Marina pela companhia e pela acolhida como familiar e amigo. Agradeço também a Giovanni Marins pelo incentivo inicial para a busca do mestrado. Aos amigos de todos os planos da vida, meu agradecimento pelo incentivo e inspiração. Compartilho esta vitória com todos vocês.

Agradeço aos professores Fábio Horácio Pereira e Humberto Torres Marques Neto, do DATAPUC, por permitirem a flexibilização do meu horário para dedicação ao mestrado. Ao Prof. Humberto, agradeço ainda pelo incentivo e apoio constantes nesta jornada acadêmica. Aos amigos do DATAPUC, obrigado pela agradável convivência diária.

Meu sincero agradecimento aos professores Nívio Ziviani e Berthier Ribeiro-Neto pelos primeiros passos em recuperação de informação, pela orientação inicial e, principalmente, pelo apoio para o meu ingresso no mestrado. Ao Prof. Nívio, agradeço por me acolher no LATIN e por me ensinar, observando sua experiência, duas lições: a necessidade de envolvimento em ações coletivas e a lição de que resultado é conseqüência de trabalho e dedicação.

Aos amigos e colegas da UFMG e do LATIN, muito obrigado pelo apoio e pela acolhida. Em especial a Fabiano Botelho pelas dúvidas esclarecidas e pela amizade; a Thierson pela atenção e presteza; a Denilson, David e Max pelos momentos compartilhados; a Anísio pelas orientações iniciais em programação genética; a Guilherme e Hendrickson pelo apoio no LATIN. Agradeço ainda a Marco Cristo e Pável pela participação e pelas sugestões e comentários no trabalho.

Agradeço aos membros da banca, professores Nívio Ziviani, Alberto Henrique Frade Laender e André Ponce de Leon F. de Carvalho por aceitarem o convite de examinar, avaliar e colaborar com este trabalho.

Finalmente, plagiando o dito popular de que "os últimos serão os primeiros", agradeço sinceramente ao Professor Marcos André Gonçalves por esta oportunidade e por confiar no meu trabalho. Obrigado por não descartar de imediato nenhuma das minhas idéias infundadas e por incentivar e acrescentar muito àquelas que tinham potencial. Espero continuar compartilhando com você esta desafiadora e prazeirosa aventura chamada pesquisa.

Sumário

Lista de Siglas e Abreviaturas

AEs — Algoritmos Evolucionários

CCA — Abordagem de Componentes Combinados (*Combined Component Approach*)

CFC — *Collection Frequency Component*

GAs — Algoritmos Genéticos (*Genetic Algorithms*)

GP — Programação Genética (*Genetic Programming*)

IDF — *Inverse Document Frequency*

MAP — Média das Precisões Médias (*Mean Average Precision*)

NC — *Normalization Component*

PAVG— Precisão Média (*Average Precision*)

RI — Recuperação de Informação

SRI — Sistema de Recuperação de Informação

SVM — Máquinas de Vetor de Suporte (*Support Vector Machines*)

TF — *Term Frequency*

TFC — *Term Frequency Component*

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivação

O desenvolvimento intelectual, social e científico da humanidade está intimamente relacionado ao registro físico do conhecimento humano. Desde os primórdios, através da escrita, passando pela evolução das técnicas gráficas, até os dias atuais da informação digital, o homem vem registrando, estocando, compartilhando e recuperando o conhecimento acumulado ao longo dos séculos de existência. Na interseção entre a Ciência da Informação e a Ciência da Computação, insere-se a Recuperação de Informação (RI), lidando principalmente com a representação, o armazenamento, a organização e o acesso a itens de informação.

Com o advento e o crescimento da *World Wide Web* ou simplesmente Web, no início da década de 90, considerada como um grande repositório universal do conhecimento humano, e de outros repositórios de informação, como Bibliotecas Digitais, a tarefa de pesquisar informação, por um lado, tornou-se mais fácil, principalmente devido à disponibilização e ao compartilhamento de um grande volume de informação de caráter geral e especializado, que passou a estar acessível a milhões de pessoas em todo o mundo a um custo muito baixo. Uma grande diferença existente entre a Web e outros repositórios de informação é, sobretudo, a liberdade de publicação, isto é, não existe controle sobre o que as pessoas podem publicar na Web. Por outro lado, porém, em meio a bilhões de documentos[1],

[1]Segundo informação do sítio "BOUTELL.COM" o tamanho da Web em fevereiro de 2007 foi estimado em quase 30 bilhões de páginas. Além disso, também foi publicado neste sítio que precisar o tamanho exato da Web nos dias de hoje é uma pergunta muito difícil de se responder (disponível em http://www.boutell.com/newfaq/misc/sizeofweb.html).

recuperar informação pertinente a uma necessidade de informação do usuário transformou-se em um problema extremamente complexo e desafiador. Segundo John Battelle [6], a tarefa de busca é um dos problemas mais desafiadores e interessantes de toda a história da Ciência da Computação.

Neste contexto, as máquinas de busca (*search engines*) surgiram como ferramentas fundamentais para a tarefa de recuperação de informação em uma coleção de documentos. Estas ferramentas são baseadas em modelos de recuperação de informação, cuja finalidade principal é fornecer o arcabouço teórico para que a máquina de busca possa ser capaz de satisfazer às necessidades de informação dos usuários. Com o crescimento vertiginoso do volume de informação disponível, a tarefa de encontrar o conteúdo procurado tornou-se difícil, dispendiosa e, em muitos casos, complexa tanto para máquinas de busca quanto para o usuário final. Dessa forma, investigar sobre ferramentas de busca e modelos de recuperação de informação que possam atender à necessidade de informação de um usuário de forma mais efetiva, é um tema fundamental nos dias de hoje.

Os modelos de recuperação de informação têm percorrido um enorme caminho ao longo dos anos. Desde o modelo de espaço vetorial proposto por Salton [35, 37], muitas outras alternativas têm sido propostas tais como o modelo probabilístico [30, 31], o modelo de indexação semântica latente (*latent semantic indexing*) [16], as redes bayesianas de inferência (*inference networks*) [42] e de crenças (*belief networks*) [29], o modelo de rede neural (*neural network model*) [47], os modelos estatísticos de linguagem (*statistical language models*) [26], o *set-based model* [27, 28], dentre outros.

Basicamente, os modelos de recuperação de informação tentam encontrar o conjunto de documentos de uma coleção, que satisfaça à necessidade de informação de um usuário. Isto é, o usuário formula uma consulta, como sendo a expressão de sua necessidade de informação, submete ao processador de consultas de uma máquina de busca e espera como resultado uma lista ordenada de documentos (*ranking*), cuja ordem tenta expressar o grau de relevância de cada documento retornado frente à consulta submetida. A ordem dos documentos é calculada por uma função de similaridade ou função de ordenação (*ranking function*) que atribui um número real a cada documento retornado, representando o grau de similaridade entre o documento e a consulta.

Diferentes funções de ordenação têm sido investigadas ao longo dos anos. No entanto, a maioria delas geralmente tem um caráter geral, isto é, são projetadas para serem efetivas em qualquer tipo de coleção. Uma função de *ranking* mais efetiva é aquela cujas respostas retornadas para uma consulta, aproxima-se do conjunto de documentos relevantes para

esta consulta, também chamado por Robertson e Sparck-Jones [31] de conjunto ideal de respostas. O trabalho de Zobel e Moffat [50], por exemplo, apresenta mais de 1.000.000 de possibilidades diferentes de calcular uma função de ordenação. Entretanto, ao final dos experimentos, eles concluíram que nenhuma delas é consistentemente efetiva em todas as coleções. Isto é, uma função de ordenação pode ter sucesso em um determinado domínio, mas não ser efetivo em um outro. Outra conclusão importante daquele trabalho é que a exploração exaustiva do espaço de busca das funções de ordenação não é uma solução viável para verificar qual função tem melhor desempenho em uma determinada coleção.

Em função disso, pode-se concluir que investigar funções de ordenação mais efetivas é um tema muito relevante e complexo no contexto atual de busca por informação: (i) relevante, pois, como afirma Battelle [6], as ferramentas de busca na Web, como Google por exemplo, estão reinventando os negócios e transformando a vida da sociedade moderna; e (ii) complexo, pois desde o trabalho de Zobel e Mofat em [50], na década de 90, foi mostrado que o número estimado de funções de ordenação já publicadas na literatura até aquele momento havia ultrapassado a casa dos milhares. A principal motivação deste trabalho, portanto, é tentar melhorar a satisfação das necessidades de informação dos usuários de mecanismos de busca, buscando descobrir uma função de ordenação que possa retornar mais documentos relevantes em resposta a uma consulta do usuário.

1.2 Definição do Problema

O problema investigado por este trabalho é o da descoberta de funções de ordenação específicas para coleções de documentos. Não obstante existir um grande número de funções de ordenação de caráter geral na literatura, este trabalho investiga o problema da geração de uma função de ordenação mais específica para uma determinada coleção. As abordagens de geração de funções de ordenação partem da hipótese de que é possível melhorar a efetividade de um mecanismo de busca, quando se utiliza algum conhecimento prévio das características da coleção e, dessa forma, gerar uma função de ordenação que leva em consideração estas características particulares de uma dada coleção. Por exemplo, a diferença entre os tamanhos dos documentos de uma coleção pode ser bem mais acentuada do que em outra, assim como a diferença entre as freqüências de termos e documentos, a natureza das consultas a serem formuladas pode variar de uma coleção para outra de acordo com o seu propósito. Ou seja, uma função de ordenação que leva em consideração estas particularidades de cada coleção tende a ser mais efetiva do que uma outra de propósito

geral.

Em virtude disso, deseja-se encontrar uma função de ordenação $r(d_j, q) : D \times Q \rightarrow \Re$, onde d_j é um documento, q uma consulta, D é um conjunto de documentos de uma coleção e Q é um conjunto de consultas para uma necessidade de informação do usuário, que seja mais específica para uma determinada coleção. Uma vez descoberta, esta função poderá ser utilizada na tarefa de recuperação de informação nesta coleção.

1.3 Objetivos

Levando em consideração o problema definido na seção anterior, este trabalho levanta a seguinte hipótese de pesquisa: *"É possível melhorar a efetividade de uma máquina de busca através da descoberta de uma função de ordenação mais específica para uma determinada coleção, a partir de componentes significativos e extraídos de funções de ordenação bem conhecidas e comprovadamente efetivas?"*

Dessa forma, a partir dessa hipótese de pesquisa, define-se o principal objetivo deste trabalho como sendo a proposição de uma abordagem para encontrar funções de ordenação mais adaptadas a uma determinada coleção baseada em Programação Genética (GP - *Genetic Programming*) [21]. Diferentemente de outras abordagens baseadas em GP, o processo evolutivo da abordagem proposta por este trabalho baseia-se em componentes de diferentes funções de ordenação comprovadamente eficazes e conhecidas da literatura de recuperação de informação. Parte-se da hipótese de que estes componentes, que em verdade são partes de diversas funções de ordenação, são individualmente representativos e ricos de significado. Acredita-se, dessa forma, que o conhecimento humano por trás destas conhecidas funções é significativo e pode ser aproveitado para a geração de uma nova função de ordenação mais específica para uma coleção. Abordagens anteriores de geração de funções de ordenação usando GP [11, 12, 13, 14, 15, 24, 41] utilizam basicamente informações estatísticas brutas da coleção tais como freqüência de termos nos documentos, freqüência de documentos na coleção, tamanho dos documentos e tamanho médio dos documentos. A Abordagem de Componentes Combinados (CCA - *Combined Component Approach*), proposta por este trabalho, é descrita no Capítulo 3.

Programação genética é uma técnica evolutiva, proposta por John Koza [21], que realiza de forma otimizada a exploração do espaço de soluções de um problema. Esta técnica será discutida na Seção 2.2. A escolha de programação genética deve-se à natureza do problema, isto é, o espaço de possibilidades de funções de ordenação é muito grande,

conforme investigado inicialmente por Salton e Buckley [36] e depois por Zobel e Moffat [50]. Em função disso, GP sendo um paradigma de aprendizado de máquina (*machine learning*) [21, 23], pode descobrir funções de ordenação potencialmente mais efetivas, através de operações inspiradas em técnicas evolutivas, como cruzamento, mutação e reprodução. GP irá explorar o espaço de busca das soluções possíveis para um determinado problema, aprendendo combinações de diferentes características, representadas no arcabouço GP por meio de terminais, para a geração de indivíduos que resolvam bem o problema estudado. Outras razões para o uso de GP, incluem o fato de a solução descoberta aproximar-se do máximo global como investigado por Koza [21], pois, apesar de não se poder garantir que a solução seja globalmente ótima, resultados interessantes têm sido relatados na literatura [41, 14] de ganhos sobre modelos de recuperação de informação tradicionais.

1.4 Trabalhos Relacionados

Diferentes abordagens para descoberta de funções de ordenação baseadas em técnicas de aprendizado de máquina, tais como algoritmos genéticos e programação genética, têm sido propostas na literatura de recuperação de informação. Fan *et al.* [11] propõem uma nova abordagem para gerar automaticamente estratégias de ponderação de termos para diferentes contextos baseada em programação genética. Eles argumentaram que cada contexto específico exige um estratégia diferente de ponderação de termos, isto é, uma função de ordenação deve ser adaptada para diferentes coleções de documentos. Ao longo dos anos de pesquisa, Fan *et al.* [12, 13, 14, 15] têm demonstrado que GP tem se mostrado uma técnica eficaz para melhorar a efetividade da tarefa de recuperação de informação. Em [9], é estudado também o efeito de diferentes funções de utilidade, isto é, funções que, quando usadas como funções de aptidão[2], privilegiam a recuperação de documentos relevantes no topo da lista de documentos retornados. Em todos esses trabalhos, os terminais definidos para o arcabouço GP são baseados fundamentalmente em informações estatísticas brutas da coleção, documentos, termos e consultas, tais como freqüência do termo (tf), número total de documentos, tamanho em *bytes* de um documento, número de termos distintos em um documento etc. Em [12], Fan *et al.* comparam sua abordagem (referenciada de agora em diante por FAN-GP) com uma abordagem baseada em aprendizado de máquina que utiliza redes neurais. Nesse trabalho, os resultados mostram que FAN-GP supera substancialmente a estratégia baseada em redes neurais. Em [14], FAN-GP descobre funções

[2]Funções de aptidão e outros conceitos de programação genética serão apresentados na Seção 2.2.5

de ordenação para busca na Web, explorando a informação de estrutura dos documentos Web, superando uma outra abordagem baseada em máquinas de vetor de suporte (SVM) na tarefa de recuperação de informação.

O trabalho de Trotman [41] também apresenta uma abordagem de GP baseada em informações estatísticas de uma coleção de documentos. Adicionalmente, todavia, diferente do trabalho de Fan *et al.*, o trabalho de Trotman inclui na população inicial de indivíduos, algumas funções de ordenação usadas como linha de base para comparação, tais como [33, 36, 40]. De acordo com esse autor, esta adição garante que o pior desempenho possível durante a fase de treinamento ou aprendizado é pelo menos tão bom quanto a melhor função da linha de base incluída na população inicial. O fato de Trotman ter escolhido para representar os indivíduos, basicamente informações estatísticas, significa que não é possível garantir a manutenção dos componentes presentes nas funções de ordenação da linha de base. Isto é, durante o processo evolutivo as funções da linha de base podem ser distorcidas ou completamente destruídas pela formação de novos indivíduos. Na verdade, os melhores indivíduos reportados por Trotman não contêm componentes das funções de ordenação da linha de base. A abordagem CCA proposta neste trabalho, no entanto, garante a preservação dos bons componentes e de todo o conhecimento que eles representam, tendo (indiretamente) superado os ganhos obtidos por Trotman. Quando aplicada a uma coleção específica (FBIS), a abordagem de Trotman alcançou um ganho de 32,90% na média das precisões médias (MAP)[3]. Este ganho, no entanto, foi descrito por Trotman como sendo um resultado atípico. O ganho médio para todas as coleções testadas obtido pela melhor execução da abordagem de Trotman foi de cerca de 8%, com ganhos negativos para algumas coleções. Os resultados obtidos por CCA superam BM25[4] em mais de 40% na média das precisões médias para a coleção TREC-8[5], sugerindo um ganho substancial sobre a abordagem de Trotman. Finalmente, diferentemente de CCA, cuja abordagem sugere uma função de ordenação específica para cada coleção, Trotman tem o objetivo de encontrar uma única função genérica que seja efetiva para todas as coleções.

O trabalho de Oren [24] explora algumas funções tf-idf[6] geradas por uma abordagem de GP. São utilizadas como terminais do arcabouço evolutivo, informações estatísticas da coleção, o inverso da freqüência do documento (idf) e duas variações da freqüência do termo em um documento (tf). Os resultados obtidos por Oren, todavia, não foram significativos

[3] MAP e outras métricas estão descritas na Seção 2.1.5.

[4] BM25 e outras funções de ordenação usadas como linha de base são descritas na Seção 4.4.

[5] As coleções de referência utilizadas, TREC-8 e WBR99, são descritas na Seção 4.1.

[6] A estratégia de ponderação tf-idf está detalhada na Seção 2.1.2.

comparados a uma estratégia tf-idf clássica.

Ao contrário de todos esses trabalhos supracitados, a abordagem CCA, ao invés de informações estatísticas da coleção, usa partes de funções de ordenação bem conhecidas, significativas e comprovadamente eficazes como terminais do arcabouço GP, para representar componentes de um esquema de ponderação de pesos. Como mencionado anteriormente, a hipótese deste trabalho baseia-se na idéia de usar componentes ricos como terminais do arcabouço GP, para obter melhores funções de ordenação ao final do processo evolutivo. Na realidade, como os experimentos irão demonstrar, CCA também se mostrou estável e consistente na geração de efetivas funções de ordenação, se comparado com os trabalhos descritos anteriormente, baseados somente em informações estatísticas. Além disso, o processo evolutivo de CCA conseguiu convergir mais rapidamente em uma boa função de ordenação e foi possível atenuar o problema da função descoberta ficar muito especializada nos dados utilizados para treinamento.

Vale a pena mencionar a existência de outros trabalhos que também são, de certa forma, relacionados a este, como os trabalhos focados na combinação de resultados de diferentes funções de ordenação. Em [5], Bartell *et al.* mostram que a efetividade da tarefa de recuperação de informação pode ser melhorada significativamente pela combinação dos resultados de diferentes algoritmos de recuperação, na medida que cada um deles cobre diferentes aspectos de uma coleção de documentos. Pathak *et al.* [25] introduzem um método que utiliza Algoritmos Genéticos (GAs - *Genetic Algorithms*) em RI. Especificamente, é mostrado como os GAs podem ser utilizados para adaptar diversas funções de casamento, usadas para casar descrições de documentos a descrições de consultas. Vogt *et al.* [43] propõem o uso de combinação linear para fusão de resultados retornados por diferentes funções de ordenação. Esse modelo combina as listas de resultados de múltiplos sistemas de recuperação de informação realizando uma soma ponderada das pontuações individuais de cada documento nos diferentes sistemas. O trabalho de Fan *et al.* [10] também usa algoritmos genéticos para combinar diferentes funções de ordenação. A abordagem deste trabalho difere de todos os demais, pois CCA não realiza a fusão ou combinação de resultados de funções de ordenação, mas componentes de funções de ordenação de diferentes sistemas de RI são utilizados como terminais do arcabouço GP para gerar funções de ordenação completamente novas e diferentes.

1.5 Contribuições da Dissertação

Neste trabalho foi investigada a descoberta de estratégias de ordenação específicas para coleções de documentos. A abordagem CCA de componentes combinados [2], proposta por este trabalho, baseia-se em um arcabouço de Programação Genética que considera algumas características importantes e únicas de cada coleção de documentos, a fim de descobrir uma função de ordenação mais efetiva para esta coleção do que qualquer outra função genérica.

Dos pontos de vista teórico e experimental, este trabalho apresenta basicamente as seguintes contribuições:

- Do ponto de vista teórico, foi proposto um novo arcabouço baseado em Programação Genética para a descoberta de funções de ordenação específicas para coleções de documentos. Esse arcabouço é baseado em uma abordagem de componentes combinados que utiliza componentes de funções de ordenação conhecidas e comprovadamente eficazes como a base para todo o processo evolutivo. O uso de componentes ricos e significativos, como terminais do arcabouço GP, ao invés de informações estatísticas mostrou-se efetivo na geração de funções de ordenação específicas para uma coleção de documentos. A abordagem CCA pode ser adaptada para ser aplicada também em outros problemas de recuperação de informação tais como classificação e recuperação de imagens.

- Do ponto de vista experimental, a abordagem CCA foi implementada e diversos experimentos foram realizados para atestar a efetividade desta abordagem em comparação a outras funções de ordenação e outros métodos baseados em GP. De acordo com os experimentos realizados, que serão apresentados no Capítulo 4, os resultados obtidos pela abordagem de componentes combinados superou os resultados de outras abordagens importantes descritas na literatura. Portanto, os resultados obtidos pela abordagem CCA nos experimentos realizados podem também servir de linha de base para novos trabalhos de descoberta de funções de ordenação.

1.6 Organização da Dissertação

Esta dissertação está organizada em capítulos dos quais este, o primeiro, introduz o problema, os objetivos e os trabalhos relacionados a esta dissertação. O Capítulo 2 apresenta os conceitos básicos relativos a Recuperação de Informação e Programação Genética que são fundamentais para todas as idéias discutidas por este trabalho. O Capítulo 3 descreve

a Abordagem de Componentes Combinados proposta por este trabalho. Os experimentos realizados e os resultados obtidos são apresentados e analisados no Capítulo 4. Finalmente, o Capítulo 5 apresenta as conclusões deste trabalho de pesquisa e algumas perspectivas de trabalhos futuros.

Capítulo 2

Conceitos Básicos

Neste capítulo, é apresentada uma revisão conceitual dos principais tópicos referentes ao problema da geração de funções de ordenação para máquinas de busca usando programação genética. Serão discutidos, inicialmente, os conceitos e modelos de recuperação de informação, com enfoque especial para o modelo vetorial. Mais adiante, serão apresentados os conhecimentos básicos de programação genética necessários ao leitor deste trabalho.

2.1 Recuperação de Informação

O termo "Recuperação de Informação" (*Information Retrieval*) foi utilizado inicialmente por Calvin Mooers em 1951 [39], quando o escopo de atuação desta área de pesquisa foi definido.

> "*Recuperação de informação lida com os aspectos intelectuais da descrição da informação e sua especificação para busca, e também de qualquer sistema, técnicas ou máquinas que são empregadas para realizar esta operação.*" (Mooers *apud* Saracevic, 1999)

Para Baeza-Yates e Ribeiro-Neto [3], recuperação de informação lida com a representação, armazenamento, organização e acesso a itens de informação, de forma que os usuários possam acessar de forma fácil a informação na qual estão interessados. Recuperação de Informação navega na fronteira entre a Ciência da Informação e a Ciência da Computação.

Um tópico fundamental em RI é o conceito de relevância. Relevância está associada à necessidade de informação de um usuário. Mais do que isso, é uma prerrogativa do

usuário. Em 1964, Goffman [17] definiu relevância como uma medida da informação transmitida por um documento em relação a uma consulta. Já Rees [17], em 1966, definiu o conceito de relevância como um critério usado para quantificar a efetividade de Sistemas de Recuperação de Informação.

> "*Relevância é o critério usado para quantificar o fenômeno de quando os indivíduos (usuários) julgam o relacionamento, a utilidade, a importância, o grau de coincidência (match), a adequação (fit), a proximidade, a conveniência, a intimidade (closeness), a pertinência, o valor ou influência de documentos ou representações de documentos para uma necessidade de informação.*" (Rees *apud* Greisdorf, 2000)

No contexto dos Sistemas de Recuperação de Informação, a relevância descreve o grau de aceitação ou rejeição de um documento retornado por um sistema em relação a uma consulta fornecida por um usuário. Um Sistema de Recuperação de Informação (SRI) tem como principal objetivo a recuperação de todos os documentos que são relevantes a uma consulta do usuário, assim como, tanto quanto possível, evitar a recuperação de documentos não relevantes. Um SRI deve recuperar documentos em uma coleção e ordená-los de acordo com o grau de relevância para uma dada consulta. Neste contexto, existem dois problemas diferentes e igualmente desafiadores: recuperar informação de forma eficiente e estimar a relevância dos documentos recuperados para a ordenação do conjunto resposta.

Assim, como visto na Figura 2.1, o problema de recuperação de informação envolve um usuário, um SRI e uma coleção de documentos.

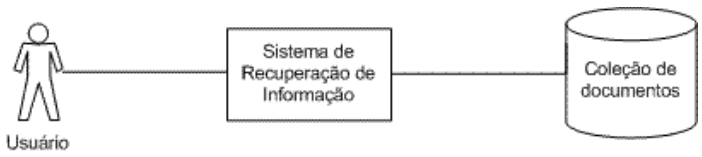

Figura 2.1: O problema de recuperação de informação.

O usuário tem uma necessidade de informação que é expressa sob a forma de uma consulta. Usualmente, consultas são expressas em uma ou mais palavras-chave. A consulta é então submetida ao SRI, cujo objetivo principal é pesquisar na coleção os documentos que satisfaçam a necessidade de informação do usuário, isto é, os documentos relevantes para a consulta. Uma vez recuperados os documentos, antes de enviar a resposta ao usuário, o SRI deve ordenar os documentos recuperados de acordo com a predisposição em satisfazer

a consulta. Por fim, o usuário então examina a lista ordenada de documentos (*ranking*) em busca de informação útil.

Os algoritmos de ordenação presentes nos Sistemas de Recuperação de Informação tentam predizer quais documentos recuperados são relevantes ou não a uma consulta do usuário. Os documentos que aparecem no topo da lista ordenada de documentos são aqueles com mais chance de serem relevantes. Por isso, os algoritmos de ordenação são considerados o coração dos Sistemas de Recuperação de Informação. A qualidade de um algoritmo de ordenação pode ser verificada através de medidas de avaliação. A Seção 2.1.5 aborda algumas destas medidas.

2.1.1 Modelos de Recuperação de Informação

Um algoritmo de ordenação é construído de acordo com um modelo de recuperação de informação, que representa uma possível abordagem para encarar o problema da relevância de documentos. Ao longo dos anos, surgiram muitos modelos de recuperação de informação para ordenação de documentos de uma coleção. Os mais conhecidos são, indubitavelmente, os modelos baseados em um espaço vetorial [35, 37, 38], os modelos probabilísticos [29, 30, 31, 42] e os modelos estatísticos de linguagem [26]. A Seção 2.1.2 irá tratar em detalhes do modelo de espaço vetorial. Além deste, existem ainda outros modelos algébricos tais como o modelo de indexação semântica latente [16] e o modelo de rede neural [47].

Um modelo de recuperação de informação atua, basicamente, sobre uma coleção de documentos e um conjunto de consultas. Além disso, possui uma função de ordenação que ordena os documentos retornados, de forma a colocar os mais relevantes no topo. Para se definir claramente o que é um modelo de recuperação de informação, Baeza-Yates e Ribeiro-Neto [3] propuseram uma caracterização formal dos modelos de RI.

Definição 1 Um **Modelo de Recuperação de Informação** é definido por uma quádrupla $[D, Q, \mathcal{F}, r(q_i, d_j)]$ onde

(1) D é um conjunto composto por representações para os documentos em uma coleção.

(2) Q é um conjunto formado por representações (consultas) para uma necessidade de informação do usuário.

(3) $\mathcal{F}$ é um arcabouço para modelagem de representações de documentos, consultas e seus relacionamentos.

(4) $r(q_i, d_j)$ é uma função de ordenação que associa um número real a uma consulta $q_i \in Q$ e uma representação de documento $d_j \in D$ para ordenar os documentos de acordo com a consulta.

Definição 2 Seja D uma coleção de documentos e $d_j \in D$ um documento pertencente à coleção D. Seja Q um conjunto de consultas e $q_i \in Q$ uma consulta para a qual existe um conjunto de documentos relevantes pertencentes à coleção D. Define-se **função de ordenação** como sendo a função r tal que $r(q_i, d_j) : Q \times D \rightarrow \Re$, representando a medida de similaridade entre uma consulta q e um documento d_j.

As funções de ordenação, geralmente, calculam uma medida de similaridade entre uma consulta q_i e um documento d_j, definindo uma ordem entre os documentos retornados em resposta a uma consulta. Mais especificamente, o cálculo da função de ordenação, normalmente, considera a importância ou peso de cada termo presente na consulta q_i em relação ao documento d_j. Ou seja, o cálculo da medida de similaridade para o documento d_j pode ser obtido a partir da soma dos pesos de cada termo presente na consulta para o documento d_j.

2.1.2 Modelo de Espaço Vetorial

O modelo de espaço vetorial [37, 35], ou simplesmente modelo vetorial, é, sem dúvida, o modelo mais conhecido em recuperação de informação. A idéia central do modelo vetorial é representar algebricamente, como vetores em um espaço euclidiano t-dimensional, onde t é o número de termos distintos da coleção[1], o conjunto de termos de uma consulta q e dos documentos de uma coleção D. Um documento d_j é associado a um vetor $\vec{d_j}$ representado por $\vec{d_j} = (w_{1,j}, ..., w_{i,j}, ..., w_{t,j})$, onde $w_{i,j} \geq 0$ é o peso de um termo k_i em um documento d_j. Da mesma forma, uma consulta q é associada a um vetor $\vec{q}$ representado por $\vec{q} = (w_{1,q}, ..., w_{i,q}, ..., w_{t,q})$, onde $w_{i,q} \geq 0$ é o peso de um termo k_i na consulta q. Um exemplo de como os vetores $\vec{d_j}$ e $\vec{q}$ podem ser representados em um espaço vetorial t-dimensional pode ser visto na Figura 2.2.

Uma vez representados os documentos e a consulta em um espaço vetorial é possível calcular o grau de similaridade de um documento d_j em relação a uma consulta q como sendo a similaridade entre os vetores $\vec{d_j}$ e $\vec{q}$. Esta similaridade pode ser calculada, por exemplo, pelo cosseno do ângulo θ formado entre estes dois vetores [3]. Esta fórmula é conhecida por medida do cosseno.

[1] O conjunto de termos distintos de uma coleção de documentos é conhecido por **vocabulário**.

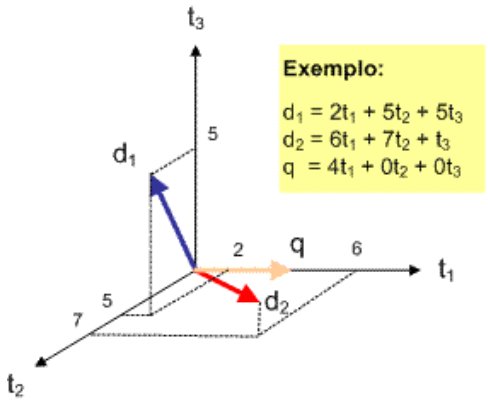

Figura 2.2: Exemplo de um espaço vetorial 3-dimensional.

$$sim(d_j, q) = \cos\theta = \frac{\vec{d_j} \bullet \vec{q}}{|\vec{d_j}| \times |\vec{q}|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{t} w_{i,q}^2}} \tag{2.1}$$

onde $|\vec{d_j}|$ e $|\vec{q}|$ são as normas dos vetores da consulta q e do documento d_j, sendo que o fator $|\vec{q}|$ não afeta o cálculo da similaridade, sendo geralmente simplificado, já que o valor de $|\vec{q}|$ é o mesmo para todos os documentos.

Existem diferentes estratégias para se calcular os pesos $w_{i,j}$ e $w_{i,q}$ [36, 50], mas a estratégia mais usual é conhecida por $tf \times idf$ [3, 48]. Onde tf (*term frequency*) é a freqüência de um termo em um documento ou o número de vezes que um termo k_i ocorre em um documento d_j. Já idf (*inverse document freqeuncy*) é o inverso da freqüência do documento ou o número de documentos nos quais um termo k_i é encontrado, considerando toda uma coleção de documentos. Dessa forma, uma maneira possível de se calcular os pesos $w_{i,j}$ e $w_{i,q}$ foi definida por Witten *et al.* [48] como se segue:

$$w_{i,j} = f\left(tf_{i,j}\right) \times idf_i = (1 + \log tf_{i,j}) \times \log\left(1 + \frac{N}{df_i}\right) \tag{2.2}$$

onde $tf_{i,j}$ é a freqüência de um termo k_i em um documento d_j, N é o número de documentos da coleção e dfi é o número de documentos onde um termo k_i ocorre;

$$w_{i,q} = f\left(tf_{i,q}\right) \times idf_i = (1 + \log tf_{i,q}) \times \log\left(1 + \frac{N}{df_i}\right) \tag{2.3}$$

onde N é o número de documentos da coleção, dfi é o número de documentos onde um

termo k_i ocorre e $tf_{i,q}$ é o número de ocorrências de um termo k_i em uma consulta q.

Existem diversas variações para $w_{i,j}$ e $w_{i,q}$ [3, 36]. Para $w_{i,j}$, por exemplo, a alternativa definida pela Equação 2.4 também pode ser encontrada.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right) \tag{2.4}$$

onde $tf_{i,j}$ é a freqüência de um termo k_i em um documento d_j, N é o número de documentos da coleção e dfi é o número de documentos onde um termo k_i ocorre.

2.1.3 Modelo Probabilístico

O modelo probabilístico [30, 31] tenta estimar a relevância de um documento baseada na idéia de que os termos das consultas têm diferentes distribuições nos documentos relevantes e não-relevantes. Basicamente, é estimada a probabilidade do usuário encontrar um documento d_j relevante para uma consulta q. O modelo assume que esta probabilidade de relevância depende apenas da consulta e dos documentos. Ou seja, para cada consulta há um determinado conjunto de documentos que contém apenas documentos relevantes. Este conjunto de documentos relevantes R é conhecido por conjunto ideal de resposta (*ideal answer set*). Dessa forma, assume-se que os documentos pertencentes ao conjunto R são potencialmente relevantes para uma consulta, enquanto que os documentos que não fazem parte deste conjunto são considerados potencialmente não-relevantes. Dada uma consulta q, o modelo probabilístico assinala para cada documento d_j, como uma medida de similaridade entre q e d_j, a razão entre a probabilidade de d_j ser relevante a q e a probabilidade de d_j não ser relevante a q, como visto na Equação 2.5.

$$sim(d_j, q) = \frac{P(R|d_j)}{P(\overline{R}|d_j)} \tag{2.5}$$

Existem diferentes alternativas para se computar as probabilidades $P(R|d_j)$ e $P(\overline{R}|d_j)$. A medida BM25 [32, 33] presente no sistema Okapi é uma das mais bem sucedidas funções de ordenação para o modelo probabilístico. O esquema de ponderação de BM25 é uma função do número de ocorrências de um termo em um documento, na coleção, e uma função do tamanho do documento. A variação mais conhecida de BM25 [32] é apresentada na Equação 2.6,

$$sim(d_j, q) = \sum_{t \in q} w^{(1)} \times \frac{(k_1 + 1) \times tf}{(k_1 \times ((1 - b) + b \times (dl/avgdl)) + tf} \times \frac{(k_3 + 1) \times qtf}{k_3 + qtf} \quad (2.6)$$

onde tf a freqüência do termo t no documento d_j, qtf é a freqüência do termo t em uma consulta q, dl e $avgdl$ são respectivamente o tamanho do documento d_j e o tamanho médio dos documentos da coleção, normalmente calculados em *bytes*, k_1, k_3 e b são parâmetros que permitem um ajuste fino da função de ordenação, de acordo com a consulta e a coleção de documentos. Normalmente, assumem os valores $1,2$, 1000 e $0,75$ respectivamente. A Equação 2.7 apresenta $w^{(1)}$, que é o peso de um termo t em uma consulta q, definido por Robertson/Sparck-Jones [31]:

$$w^{(1)} = \frac{(r + 0.5)/(R - r + 0.5)}{(n - r + 0.5)/(N - n - R + r + 0.5)} \quad (2.7)$$

onde N é o número de documentos de uma coleção, n é o número de documentos que possuem o termo t, R é o número de documentos relevantes para a consulta q, e r é o número de documentos relevantes contendo o termo t.

2.1.4 Ponderação de Termos

Um dos pontos mais importantes no cálculo da similaridade de um sistema baseado em ponderação de termos é a escolha adequada dos valores para os pesos $w_{i,j}$ e $w_{i,q}$. No estudo de Salton e Buckley [36] foram experimentadas cerca de 1.800 formas diferentes para o cálculo dos pesos. Estas variações foram baseadas na combinação de três componentes, que foram especificados por Salton e Buckley como partes integrantes de qualquer sistema baseado em ponderação de termos. São eles, o componente da freqüência do termo (*tfc — term frequency component*), o componente da freqüência na coleção (*cfc — collection frequency component*) e o componente de normalização (*nc — normalization component*).

Uma fórmula de ponderação de pesos é definida como sendo composta de duas triplas: $\langle tfc_q, cfc_q, nc_q \rangle$, que representa o peso de um termo em uma consulta q, e $\langle tfc_d, cfc_d, nc_d \rangle$, que representa o peso de um termo em um documento d. O componente *tfc* representa quantas vezes um termo ocorre em um documento ou consulta. O componente *cfc* considera o número de documentos nos quais um termo aparece. Baixas freqüências indicam que um termo é pouco comum e, dessa forma, mais importante para distinguir documentos. Finalmente, o componente *nc* tenta compensar a diferença existente entre o tamanho dos

documentos de uma coleção. Tipicamente, uma fórmula de ponderação de termos combina estes três componentes. Dessa forma, podemos expressar uma função de ordenação baseada em um sistema de ponderação de termos como

$$sim(q,d) = \sum_{t \in q} w_{tq} \times w_{td} \tag{2.8}$$

onde $sim(q,d)$ é a medida de similaridade entre uma consulta q e um documento d e o peso total de um termo t é definido pelos pesos w_{tq} e w_{td} que representam o peso do termo t na consulta q e no documento d respectivamente. O peso w_{tq} é definido por $w_{tq} = tfc_q \times cfc_q \times nc_q$ e w_{td} é definido por $w_{td} = tfc_d \times cfc_d \times nc_d$.

Mais tarde, após a proposta de Salton e Buckley, o trabalho de Zobel e Moffat [50] explorou esta taxonomia, adicionando oito tipos diferentes de função de similaridade. Desse modo, chegou-se a mais de 1.500.000 combinações diferentes para o cálculo da similaridade entre documentos e consultas, demonstrando que o espaço de possibilidades para personalização, ajuste e refinamento de funções de ordenação é extremamente extenso. Isto estimula a aplicação de técnicas eficazes de exploração do espaço de busca, como programação genética [21], para a descoberta de funções de similaridade específicas para uma determinada coleção de documentos.

2.1.5 Avaliação de Sistemas de Recuperação de Informação

Uma vez que os sistemas de Recuperação de Informação retornam documentos relevantes e não relevantes a uma necessidade de informação do usuário, é necessário avaliar o quão preciso é um conjunto resposta retornado para uma consulta. Segundo Baeza-Yates e Ribeiro-Neto [3], este tipo de avaliação é conhecido por **Avaliação de Desempenho de Recuperação**. As medidas de avaliação mais conhecidas são **Precisão** e **Revocação** que avaliam a qualidade de um conjunto desordenado de documentos retornados. Isto é, estas medidas são utilizadas em uma determinada posição do conjunto resposta. Usualmente, calcula-se e traça-se pontos de um gráfico de precisão versus revocação após cada documento retornado.

Para se definir formalmente os conceitos de precisão e revocação é necessário considerar uma coleção de documentos D, uma consulta q que expressa uma necessidade de informação do usuário, o conjunto A dos documentos retornados pelo SRI em resposta a esta consulta q, o conjunto R de documentos relevantes da coleção D para a consulta q e o conjunto R_a dos documentos relevantes retornados pelo SRI, formado pela interseção dos conjuntos

R e A. Estes conjuntos são ilustrados na Figura 2.3. A partir desta definição, seja $|A|$ o número de documentos em A, $|R|$ o número de documentos em R e $|R_a|$ o número de documentos em R_a.

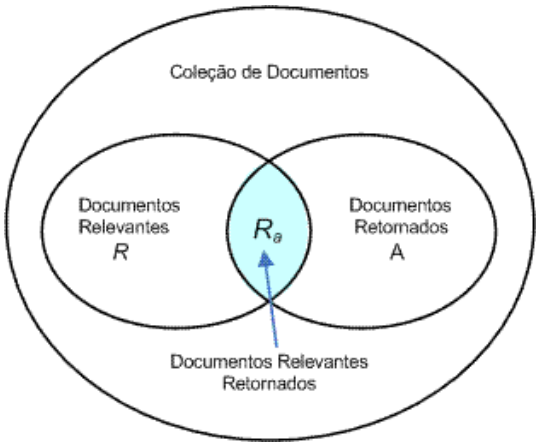

Figura 2.3: Conjuntos de documentos relevantes e retornados para uma consulta.

Definição 3 Precisão é a medida da habilidade de um sistema de retornar somente documentos relevantes. Isto é, precisão é a razão entre os documentos retornados que são relevantes $|R_a|$ e o total de documentos retornados $|A|$.

$$\text{Precisão} = \frac{|R_a|}{|A|} \tag{2.9}$$

Definição 4 Revocação é a medida da habilidade de um sistema de apresentar todos os documentos relevantes. Isto é, revocação é a razão entre os documentos retornados que são relevantes $|R_a|$ e o total de documentos relevantes $|R|$.

$$\text{Revocação} = \frac{|R_a|}{|R|} \tag{2.10}$$

Para melhor analisar e visualizar o desempenho de um Sistema de Recuperação de Informação, usualmente são utilizadas curvas de precisão em vários níveis de revocação. Ou seja, calcula-se a precisão para determinados valores de revocação. A Figura 2.4 ilustra um exemplo de curvas de precisão $\times$ revocação para dois SRI. Como pode ser observado, o sistema *SRI-1* é mais preciso do que o sistema *SRI-2* em baixa revocação, embora a partir de um certo nível de revocação *SRI-2* torna-se mais preciso do que *SRI-1*.

Como pode ser visto no exemplo da Figura 2.4, as curvas de precisão $\times$ revocação não são suficientes para determinar qual SRI é mais preciso na média. Para verificar se um

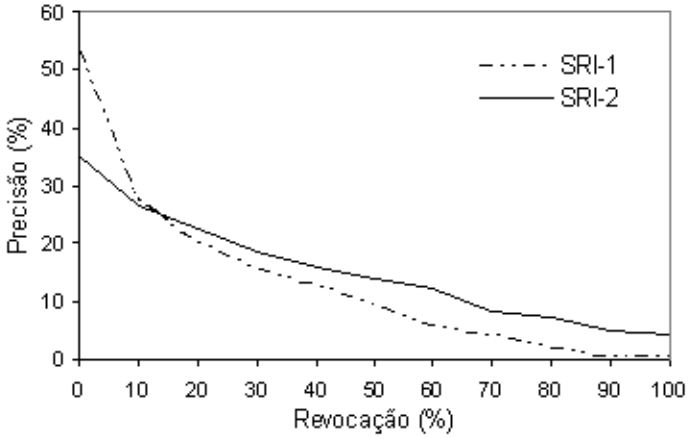

Figura 2.4: Exemplo de curvas de precisão × revocação para dois SRI.

sistema supera outro, é conveniente utilizar um único valor médio de precisão para cada consulta.

Definição 5 Precisão Média (*PAVG*) é uma medida que reflete o desempenho de um sistema sobre todos os documentos relevantes [44]. Esta medida é a média dos valores de precisão para uma consulta obtidos depois que cada documento relevante é recuperado (quando um documento relevante não é recuperado, assume-se que sua precisão é igual a zero).

$$\text{Precisão Média} = PAVG = \frac{\sum_{i=1}^{|D|} \left(r(d_i) \times \left(\frac{\sum_{j=1}^{i} r(d_j)}{i} \right) \right)}{|R|} \tag{2.11}$$

onde $r(d)$ assume o valor 1 se o documento d é relevante e 0 caso contrário. $|D|$ é o número total de documentos retornados e $|R|$ é o número total de documentos relevantes na coleção.

Definição 6 A **Média dos Valores de Precisão Média** (*MAP — Mean Average Precision*) é uma medida do desempenho médio de um sistema para N_q consultas, isto é, calcula-se a média dos valores de precisão média obtidos para cada consulta q.

$$MAP = \frac{\sum_{q=1}^{N_q} PAVG_q}{N_q} \tag{2.12}$$

onde $PAVG_q$ é a precisão média para uma consulta q e N_q é o número de consultas utilizadas.

Existem outras medidas utilizadas na avaliação de Sistemas de Recuperação de Informação, como *R-precision* (valor de precisão quando o R-ésimo documento é retornado, onde R é o total de relevantes), *P@10* (valor de precisão quando 10 documentos são retornados) etc. As definições destas e de outras medidas são descritas em [3] e [44].

2.2 Programação Genética

2.2.1 Origem

O paradigma de programação genética, introduzido por John Koza [21], pertence à família dos Algoritmos Evolucionários (AEs). Os algoritmos evolucionários são métodos de busca e otimização inspirados na idéia da seleção natural de Darwin [8], na qual ocorre o fenômeno de adaptação das espécies na luta pela sobrevivência. Durante o processo evolutivo, variações úteis ou favoráveis das espécies são preservadas, enquanto que variações prejudiciais são descartadas. Dessa forma, os algoritmos evolucionários simulam os mecanismos naturais de sobrevivência e reprodução das populações, nos quais os indivíduos mais aptos terão mais descendentes do que os menos aptos ao longo de todo processo evolutivo. programação genética surgiu a partir da extensão das idéias dos algoritmos genéticos, um outro tipo particular de algoritmo evolucionário, proposto por Holland [19]. No paradigma de algoritmos genéticos, assim como em programação genética, as possíveis soluções do problema são tratadas como indivíduos de uma população que irá evoluir ao longo de gerações. Os GAs modelam os indivíduos como estruturas de dados fixas, conhecidas por cromossomos e, geralmente, vistas como seqüências de bits de tamanho fixo. Essa representação apresenta restrições para a modelagem de uma solução para determinados problemas.

A partir da evolução das idéias dos algoritmos genéticos, principalmente a abordagem de algoritmos genéticos hierárquicos [20], que modela uma solução para um problema como sendo uma estrutura hierárquica de funções e átomos, Koza [21] propôs o paradigma de programação genética, definindo como um método automatizado para gerar geneticamente um programa de computador para resolver um dado problema, através da exploração otimizada do espaço de busca de todas as possíveis soluções para o problema. É a evolução direta de um conjunto de programas ou algoritmos para a finalidade de aprendizagem por indução [4]. O paradigma de programação genética provê um caminho para se gerar

geneticamente uma população de estruturas hierárquicas de programas de computadores para resolver toda sorte de problemas.

No paradigma de Programação Genética, populações de programas de computadores, representados por indivíduos, vão evoluindo ao longo de gerações, a partir dos princípios de Darwin de sobrevivência dos indíviduos mais aptos e do cruzamento de espécies. As estruturas utilizadas em Programação Genética são hierárquicas com forma e tamanho variáveis, eliminando, dessa forma, a restrição da estrutura fixa existente nos Algoritmos Genéticos. O processo de solução de problemas usando Programação Genética pode ser definido como sendo a busca pelo indivíduo mais apto, ou melhor indivíduo, no espaço de possibilidade das soluções para um dado problema. Em particular, este espaço de busca é o hiperespaço dos programas de computadores composto de funções e terminais apropriados para o domínio do problema.

Ao longo dos anos, muitas pesquisas têm sido realizadas em Programação Genética. Por exemplo, Gruau [18] propôs o uso de gramáticas para impor restrições sintáticas à geração de indivíduos, restringindo também, dessa forma, o espaço de busca. As restrições sintáticas são informadas como parte dos parâmetros utilizados no arcabouço genético. Estudos específicos sobre Programação Genética Orientada a Gramáticas vêm sendo conduzidos em diversos trabalhos tais como [34] e [46].

2.2.2 População e Indivíduos

A população de um algoritmo de programação genética é representada por um conjunto de indivíduos. Cada indivíduo, que representa um programa de computador ou uma possível solução de um problema, possui um conjunto de características próprias que o distingue dentre os demais indivíduos da população. Normalmente, utiliza-se uma estrutura de árvore para representar cada indivíduo de uma população. Um indivíduo é formado pela combinação de funções e terminais adequados ao domínio do problema. A Figura 2.5 ilustra um exemplo da representação em árvore de um indivíduo.

Definição 7 O conjunto dos **terminais** $T = \{t_1, t_2, ..., t_{nt}\}, nt \geq 1$, formado por variáveis, constantes e funções sem argumentos, é definido como sendo o conjunto que representa as características importantes de um dado problema que se deseja solucionar.

Definição 8 O conjunto $F = \{f_1, f_2, ..., f_{nf}\}, nf \geq 1$, é definido como sendo o conjunto das **funções**, formado por operadores aritméticos ($+, -, *$ etc.) e funções matemáticas

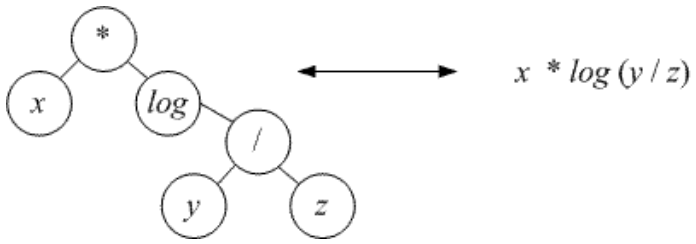

Figura 2.5: Representação em árvore de um indivíduo.

($log, sqrt$ etc.) dentre outros, a serem utilizadas para combinar terminais. Cada função $f \in F$ deve possuir pelo menos um argumento.

Definição 9 Define-se por **espaço de busca**, o conjunto $S = \{s_1, s_2, ..., s_{ns}\}, ns \geq 1$, de soluções possíveis para um problema. Cada $s \in S$ é criado pela livre combinação de elementos dos conjuntos T e F e representa uma possível solução para um dado problema.

Definição 10 O conjunto $P = \{i_1, i_2, ..., i_{np}\}, np \geq 1$, representa uma **população** que é definida como sendo um conjunto de indivíduos. Cada $i \in P$ é um indivíduo da população que representa uma possível solução para um dado problema, tal que $i \in S$.

2.2.3 A Escolha de Terminais e Funções

Uma solução encontrada por um arcabouço de Programação Genética depende, fundamentalmente, da qualidade dos terminais e das funções escolhidas para representar um determinado problema. Koza [21] define que a escolha do conjunto de terminais e de funções deve satisfazer os requisitos ou propriedades de Fechamento e Suficiência.

Fechamento

A propriedade de fechamento (*closure*) requer que cada função do conjunto F seja capaz de aceitar como argumento qualquer valor retornado por uma outra função do conjunto F e qualquer valor possível de ser assumido por um terminal do conjunto de terminais T definidos. Isto é, cada função do conjunto de funções F deve ser bem definida e "fechada" para qualquer combinação de argumentos que puder encontrar.

Um exemplo típico de uma função modificada para atender a propriedade de Fechamento é a operação de divisão. Como a divisão por zero não é possível matematicamente, modifica-se esta operação para retornar o valor 1 (um), por exemplo, onde seria feita uma divisão por zero.

Suficiência

A propriedade de suficiência requer que o conjunto de terminais T e o conjunto de funções F sejam capazes de expressar uma solução para o problema. Ou seja, ao definir um conjunto de funções e terminais deve-se ter em mente que uma composição de quaisquer funções e terminais irá produzir uma solução. Portanto, deve-se ter fortes evidências de que os terminais e funções escolhidas possam representar características importantes do problema abordado e, além disso, possam ser capazes de gerar uma boa solução.

2.2.4 População Inicial

Geralmente, o conjunto de indivíduos que forma a população inicial é gerado aleatoriamente a partir dos conjuntos de funções F e de terminais T. O processo é realizado escolhendo-se funções e terminais para a composição da árvore, até que se tenha atingido a profundidade máxima definida para a árvore representando cada indivíduo. Geralmente a profunidade máxima de um indivíduo é um parâmetro do arcabouço GP. A geração aleatória dos indivíduos da população inicial pode ser feita de acordo com os métodos [21], descritos a seguir:

Método *Full*

O método *full* de geração aleatória de indivíduos para a população inicial cria árvores completas, nas quais todos os nós-folha possuem a mesma distância até o nó-raiz. Esta distância é igual à profundidade máxima definida para os indivíduos.

Método *Grow*

O método *grow* cria árvores de profundidade variável. A escolha dos nós é feita de forma aleatória entre funções e terminais; porém, a profundidade de um nó qualquer até o nó-raiz está restrita à profundidade máxima definida para um indivíduo.

Método *Ramped-half-and-half*

O método *ramped-half-and-half* é uma combinação balanceada dos métodos *full* e *grow*, isto é, cada um dos métodos é escolhido 50% das vezes. Uma faixa de valores para a profundidade dos indivíduos pode ser também definida. Um intervalo de 2-5, significa que

25% das árvores serão geradas com profundidade 2, 25% com profunidade 3 e assim por diante.

2.2.5 Função de Aptidão

No paradigma de programação genética, a função de aptidão ou *fitness* retorna a aptidão de um indivíduo ou o quão boa é a solução representada por ele para um dado problema. Os melhores indivíduos terão maiores valores de *fitness* e, por conseguinte, terão mais chance de se reproduzirem. A escolha da função de *fitness* depende diretamente do domínio do problema. Como todo o processo evolutivo é baseado na função de *fitness*, torna-se fundamental escolher uma função apropriada para um determinado domínio.

Definição 11 Uma **função de aptidão** ou *fitness* para um indivíduo $i \in P$ é definida como sendo a função a que retorna um número real de acordo com a aptidão de um indivíduo, tal que $a(i) : P \rightarrow \Re$, onde P é a população de indivíduos e $\Re$ o conjunto dos números reais.

2.2.6 Seleção

Para escolher os indivíduos sobre os quais serão aplicados os operadores genéticos, é necessário realizar a etapa de seleção. Nesta fase, de acordo com os valores de aptidão de cada indivíduo, será realizada a escolha daqueles que irão sofrer as transformações genéticas e conseqüentemente compor a nova geração. Existem diferentes métodos de seleção, sendo os principais seleção proporcional (*fitness-proporcionate selection*) e seleção por torneio (*tournament selection*) [21].

Seleção Proporcional

A seleção de um indivíduo é proporcional ao valor da aptidão normalizada. Ou seja, quanto maior o valor de aptidão de um indivíduo em relação aos demais, maior a chance de ser selecionado.

Seleção por Torneio

Para se selecionar um indivíduo, primeiro seleciona-se t indivíduos onde t é o tamanho do torneio. Após isso, escolhe-se o melhor dentre os t indivíduos. Repete-se este processo até que se tenha uma nova população.

2.2.7 Operadores Genéticos

Indivíduos evoluem geração após geração através dos operadores genéticos, que são aplicados nos indivíduos selecionados ao final de uma geração. Os principais operadores, segundo Koza [21], são reprodução (*reprodution*), mutação (*mutation*) e cruzamento (*crossover*).

Reprodução

A operação de reprodução simplesmente copia um indivíduo de uma geração g para a próxima geração g+1 sem modificar a sua estrutura. Reprodução é o mecanismo através do qual se garante a evolução dos melhores indivíduos de uma geração para a geração subseqüente durante o processo evolutivo. A Figura 2.6 ilustra uma operação de reprodução.

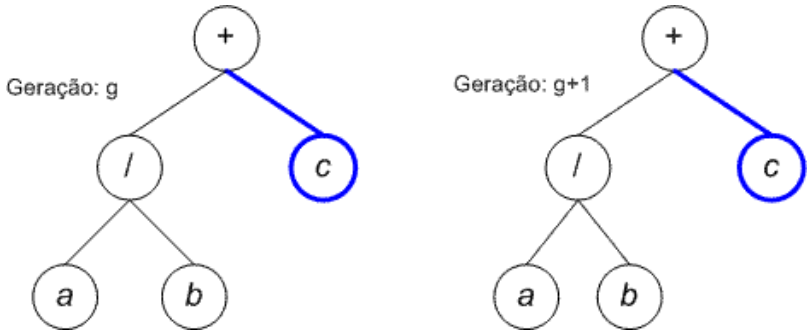

Figura 2.6: Exemplo de reprodução de um indivíduo.

Cruzamento

A operação de cruzamento seleciona dois indivíduos (pais) e realiza uma combinação de suas características (sub-árvores) gerando dois novos indivíduos. O objetivo desta operação é garantir a variedade das espécies e a diversidade da população, através da combinação de indivíduos que trocam material genético. A Figura 2.7 ilustra uma operação de cruzamento.

Mutação

A operação de mutação, ilustrada pela Figura 2.8, simula os desvios que podem ocorrer em um processo evolutivo. Uma sub-árvore de um indivíduo, escolhida aleatoriamente, é eliminada e então substituída por uma nova sub-árvore também gerada aleatoriamente. Esta operação tem o objetivo de garantir a diversidade genética da população, prevenindo a convergência prematura do processo evolutivo.

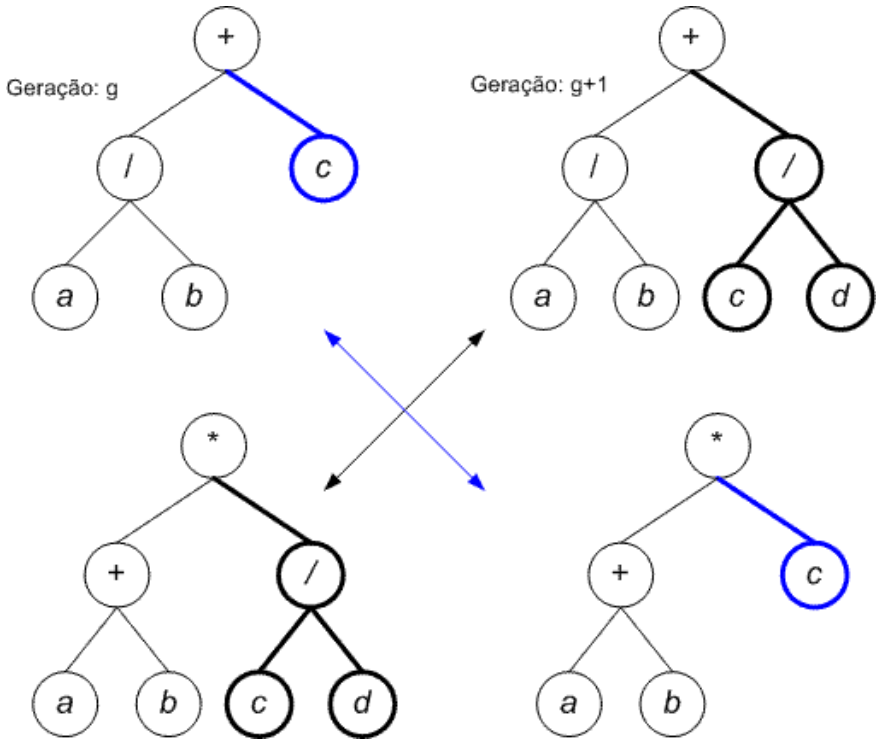

Figura 2.7: Exemplo de cruzamento entre dois indivíduos.

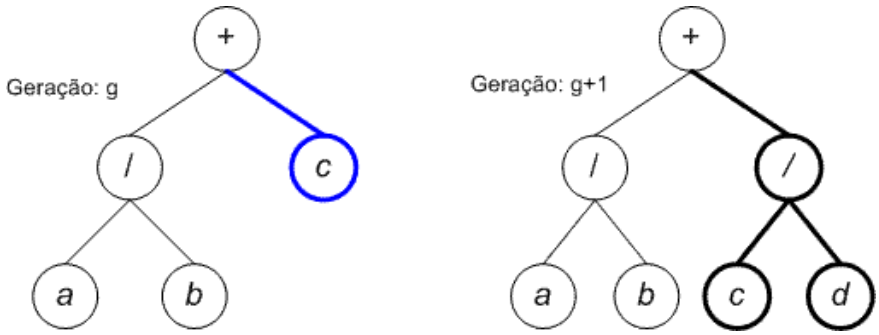

Figura 2.8: Exemplo de mutação de um indivíduo.

2.2.8 Critério de Terminação

O critério de terminação é responsável por interromper o processo evolutivo. Normalmente, usa-se um número máximo de gerações ou algum valor pretendido de *fitness* [21].

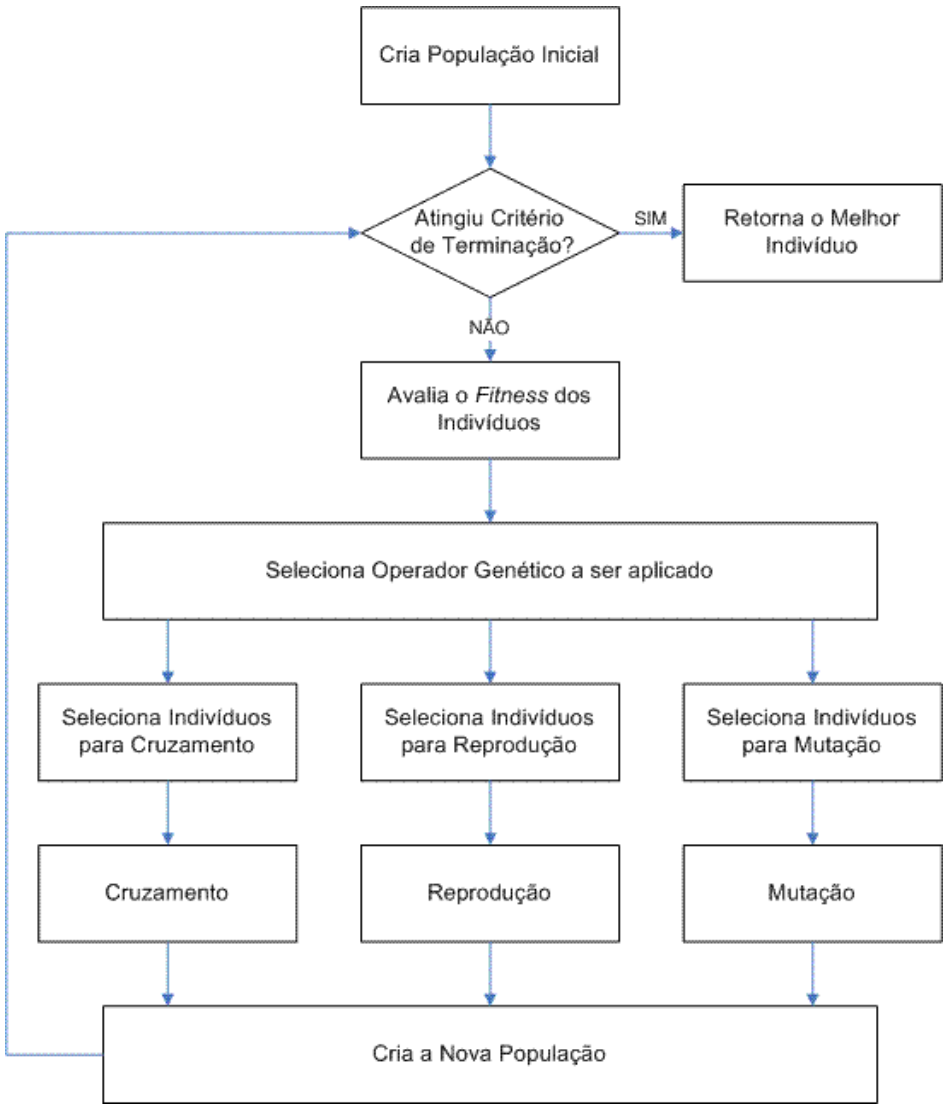

Figura 2.9: Visão Geral do Algoritmo de Programação Genética.

2.2.9 Algoritmo de Programação Genética

A Figura 2.9 apresenta uma visão geral do algoritmo de programação genética. O algoritmo começa pela criação aleatória de uma população inicial. Após isso, executa-se os seguintes passos até que o critério de terminação seja atingido.

- Avalia-se cada indivíduo através da função de aptidão;
- Seleciona-se os indivíduos através de um método de seleção para aplicação dos operadores genéticos;

- Aplica-se os operadores genéticos, criando-se uma nova população.

Ao final do processo evolutivo, após o critério de terminação ter sido satisfeito, retorna-se o melhor indivíduo descoberto.

2.2.10 O Problema do *Overfitting*

O problema de um indivíduo ficar específico demais (*overfitting*) para um determinado conjunto de dados utilizado para treino é muito comum em aplicações de aprendizado de máquina (*machine learning*). Tom Mitchell [23] formalizou o problema do *overfitting* da seguinte maneira:

Definição 12 Dado um espaço de hipóteses H e uma hipótese $h \in H$, é dito haver **"superajuste"** (*overfitting*) para os dados utilizados no treino, se há alguma hipótese alternativa $h' \in H$, tal que h apresenta um erro menor que h' sobre um conjunto de exemplos de treinamento, mas h' apresenta um erro menor que h sobre a distribuição total de exemplos, incluindo as de treinamento e qualquer outra utilizada para validação ou testes.

Ocorre *overfitting*, portanto, quando um indivíduo começa a ficar especializado demais para o conjunto de treino, devido a um "treinamento exagerado", isto é, uma solução é muito boa para os dados da amostra de treinamento, mas não generaliza para outras amostras.

Capítulo 3

Abordagem de Componentes Combinados

A Abordagem de Componentes Combinados (CCA - *Combined Component Approach*) proposta por este trabalho é uma abordagem baseada em programação genética para a descoberta de efetivas funções de ordenação. O objetivo principal de CCA é descobrir novas funções de ordenação que sejam mais adaptadas às características de uma coleção de documentos específica. Como anteriormente mencionado na Seção 1.4, diferentemente de outros trabalhos de geração de funções de ordenação baseado em GP, a idéia de CCA consiste em examinar importantes funções de ordenação presentes em diferentes modelos e sistemas de recuperação de informação descritos na literatura [1, 7, 33, 40], e extrair destas funções os componentes de um esquema de ponderação de termos, como aqueles descritos na Seção 2.1.4. Estes componentes podem ser apenas partes das funções de ordenação. Uma vez identificados, podem ser utilizados como terminais do arcabouço GP, onde serão combinados e utilizados para a geração de novas funções de ordenação mais específicas para uma determinada coleção de documentos.

Neste capítulo, será apresentado em detalhes o arcabouço de geração de funções de ordenação baseado na Abordagem de Componentes Combinados. A Seção 3.1 explica a modelagem CCA para a descoberta de funções de ordenação usando GP e a Seção 3.2 apresenta em detalhes o arcabouço GP usado por CCA.

3.1 Modelagem CCA para Descoberta de Funções de Ordenação

A idéia da abordagem CCA é descobrir uma função de ordenação $r(q_i, d_j)$, como apresentado na Definição 1, que associe um número real a cada documento d_j da coleção de acordo com a sua similaridade em relação a uma consulta q_i. Como apresentado na Seção 2.1.4, uma função de ordenação baseada em um sistema de ponderação de termos, que representa a similaridade entre uma consulta q e um documento d, pode ser expressa pelo somatório do produto dos pesos de cada termo t de uma consulta em relação a consulta q e ao documento d. Esta expressão pode ser simplificada na Equação 3.1 que define a similaridade entre uma consulta q e um documento d de acordo com a abordagem CCA.

$$sim(q, d) = \sum_{t \in q} w_{tdq} \tag{3.1}$$

onde w_{tdq} é uma função que retorna o peso de um termo t presente na consulta em relação a um documento d e uma consulta q.

Dessa forma, a abordagem CCA tenta descobrir através do arcabouço de programação genética o peso w_{tdq} que faça com que a função de ordenação para uma determinada coleção de documentos se aproxime da solução ótima. Assim como a abordagem proposta por Fan *et al.* [12, 13, 14, 15], CCA usa uma estrutura de dados de árvore para representar o peso w_{tdq}. A representação baseada em árvores permite fácil implementação, caminhamento e interpretação. A Figura 3.1 ilustra um indivíduo representando um esquema de ponderação tf-idf. Os nós-folha assim como em árvores são chamados de **terminais** e representam unidades de informações básicas que serão utilizadas para criar uma nova fórmula de ponderação para o peso w_{tdq}. Terminais são combinados através de **funções** que são representadas nos nós-internos. Em trabalhos anteriores de descoberta de funções de ordenação baseados em GP [11, 12, 13, 14, 15, 41], os terminais sempre refletem informações estatísticas básicas diretamente derivadas de um coleção, tais como freqüência de um termo em um documento ou tamanho de um documento.

A abordagem CCA, no entanto, difere de todas as demais por usar terminais mais significativos. A Figura 3.2 exemplifica um outro indivíduo representando um esquema de ponderação tf-idf. Neste caso, a informação de idf é por si mesma um terminal e não o resultado de uma combinação de terminais. Nos trabalhos anteriores citados, esta informação é uma sub-árvore que deve ser explicitamente descoberta pelo processo evolutivo do

arcabouço GP, podendo até mesmo não ser descoberta. Desse modo, a abordagem CCA tira vantagem do uso de informações previamente conhecidas — componentes de funções de ordenação tais como *idf* e componentes de normalização pivoteada [40] — para a geração de novas funções de ordenação mais efetivas baseadas nestes componentes, explorando de maneira otimizada e orientada o espaço de busca de soluções.

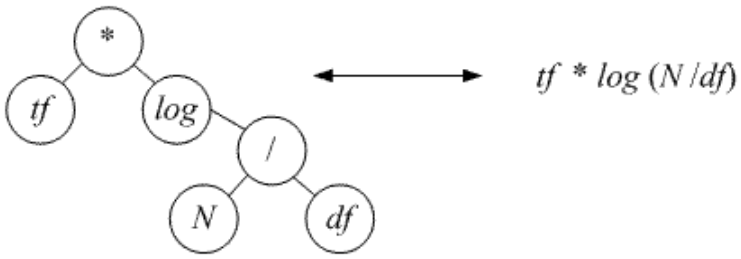

Figura 3.1: Um exemplo de uma árvore para um indíviduo tf-idf baseado em informações estatísticas de um coleção.

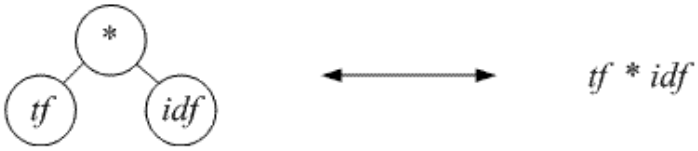

Figura 3.2: Um exemplo de uma árvore para um indíviduo tf-idf baseado na abordagem CCA de componentes combinados, onde tf e idf são componentes.

O uso de terminais mais significativos extraídos de funções de ordenação conhecidas está também fundamentado na definição da propriedade de suficiência do conjunto de terminais e funções. A propriedade de suficiência, como definido na Seção 2.2.3, requer que terminais e funções, quando combinados, consigam expressar uma solução para o problema. Dessa forma, diferentemente dos terminais baseados apenas em informações estatísticas que isoladamente não representam possíveis soluções para o problema, os terminais CCA, por construção, já representam possíveis funções de ordenação ou parte delas e, com isso, têm mais chance de atender a propriedade de suficiência e de convergir em uma solução para o problema investigado.

3.2 Visão Geral do Arcabouço GP usado em CCA

O arcabouço GP usado em CCA é basicamente um processo iterativo de duas fases ou etapas: treino e validação. Para cada fase, são selecionados um conjunto de consultas e documentos da coleção, que são chamados **conjunto de treino**, para a fase de treino, e **conjunto de validação**, para a fase de validação.

O arcabouço começa com a criação de uma população inicial de indivíduos que evoluem geração por geração. Em cada fase, treino e validação, é verificada a aptidão de cada indíviduo através da aplicação de uma função de aptidão (*fitness*). Uma vez que cada indivíduo representa um esquema de ponderação, aplicar a função de aptidão significa calcular a função de ordenação para um indivíduo de acordo com o conjunto de documentos e consultas de treino. O valor obtido para a função de aptidão é uma medida de qualidade do *ranking* obtido em relação a relevância dos documentos retornados.

Após cada geração, são escolhidos os melhores indivíduos que serão avaliados mais adiante na fase de validação. São escolhidos também, dentre os melhores indivíduos, aqueles nos quais são aplicados os operadores genéticos, comumente cruzamento, reprodução e mutação, a fim de que seja gerada a nova população de indivíduos da próxima geração. O processo evolutivo continua até ser atingido o critério definido para terminação, geralmente um número máximo de gerações

No final do processo evolutivo, depois que a última geração é criada, aplica-se a fase de validação, na qual os melhores indivíduos da fase de treino são avaliados também no conjunto de validação. Após isso, de acordo com o método de seleção utilizado, escolhe-se o melhor indivíduo considerando o desempenho nas fases de treino e validação. A visão geral do arcabouço CCA é apresentado mais detalhadamente no algoritmo listado na Figura 3.3.

3.2.1 Indivíduos

Um indivíduo, geralmente representado por um estrutura de árvore, é composto pela combinação entre terminais e funções. Um típico indivíduo CCA foi apresentado na Figura 3.2. Terminais são visualizados nos nós-folha e funções estão presentes nos nós internos.

Terminais

Os terminais CCA são baseados em componentes presentes em conhecidas e efetivas funções de ordenação descritas na litetura de recuperação de informação [1, 7, 33, 40]. Estes componentes, quando utilizados como terminais, colaboram com o processo evolutivo, pois são

```
1  Seja T o conjunto de consultas de treino;
2  Seja V o conjunto de consultas de validação;
3  Seja P a população de indivíduos;
4  Seja B_t o conjunto dos melhores indivíduos do treino e a respectiva aptidão;
5  Seja B_v o conjunto dos indivíduos da validação e a respectiva aptidão;
6  Seja F_t o conjunto dos valores de aptidão dos indivíduos no treino;
7  Seja N_g o número de gerações;
8  Seja N_b o número de melhores indivíduos;
9
10 // Inicializações
11 P ← População inicial de indivíduos gerada aleatoriamente;
12 B_t ← ∅;
13 B_v ← ∅;
14
15 // FASE DE TREINO
16 // Laço principal de gerações
17 Para cada geração g de N_g gerações execute {
18     F_t ← ∅;
19
20     // Laço de cada indivíduo em uma geração
21     Para cada indivíduo i ∈ P execute
22         F_t ← F_t ∪ {g, i, calcularAptidao(i, T)};
23
24     // Recuperar melhores indivíduos e aplicar operações genéticas
25     B_t ← B_t ∪ recuperarMelhoresIndividuos(N_b, F_t);
26     P ← aplicarOperacoesGeneticas(P, F_t, B_t, g);
27 }
28
29 // FASE DE VALIDAÇÃO
30 // Laço de cada indivíduo para aplicar conjunto de consultas de validação
31 Para cada indivíduo i ∈ B_t execute
32     B_v ← B_v ∪ {i, calcularAptidao(i, V)};
33
34 // Escolher melhor indivíduo
35 MelhorIndividuo ← aplicarMetodoSelecao(B_t, B_v);
```

Figura 3.3: Arcabouço de Programação Genética usado por CCA.

mais representativos e significativos do que os terminais simples baseados em informações estatísticas, utilizados em trabalhos anteriores de descoberta de funções de ordenação usando GP.

O conjunto de terminais utilizados no arcabouço de programação genética definido pela

abordagem CCA é apresentado na Tabela 3.2.1. Além destes, foram também utilizados valores constantes na faixa de [0..100].

3.2.2 Funções

Para combinar os terminais na formação dos indivíduos, foram utilizadas as seguintes funções: adição (+), multiplicação (∗), divisão (/) e logaritmo protegido (log). Logaritmo protegido é uma modificação da função de logaritmo natural para prevenir o retorno de valores negativos. O logaritmo protegido irá retornar 0 (zero) se o argumento for menor do que 1 (um) ou o logaritmo natural do valor absoluto do argumento passado para a função. Essa modificação é necessária para atender a propriedade de fechamento, conforme Seção 2.2.3.

3.2.3 Operadores Genéticos

Baseado no trabalho de Koza [21], foram utilizados os seguintes operadores genéticos: reprodução, cruzamento e mutação. Como mostrado na Seção 2.2, o operador de reprodução copia um indivíduo para a próxima geração sem executar nenhuma modificação no indivíduo. O operador de cruzamento cria um novo indivíduo combinando partes de dois outros indivíduos chamados "pais". Finalmente, o operador de mutação substitui uma sub-árvore de um indivíduo por uma nova sub-árvore criada aleatoriamente a partir de outros terminais e funções.

3.2.4 Função de Aptidão

Como os indivíduos representam esquemas de ponderação a serem utilizados em uma função de ordenação de documentos, a função de aptidão deve medir a qualidade do *ranking* produzido por um determinado indivíduo. Foram experimentadas duas diferentes funções: (1) a função MAP sendo a média dos valores de precisão média (PAVG) como descrita em [44] e [9] e apresentada na Equação 2.12, e (2) a função de utilidade FFP4 como definida em [9] e apresentada na Equação 3.2, como sendo uma função baseada na idéia de que a utilidade de um documento relevante diminui à medida que o documento se afasta do topo do *ranking*. FFP4 foi escolhida por ter tido um desempenho melhor do que outras funções de utilidade. Como descrito em [9], essa função é definida pela equação

$$FFP4 = \sum_{i=1}^{|D|} r(d_i) \times k_8 \times k_9^i \tag{3.2}$$

onde $r(d_i) \in (0, 1)$ assume 1 se um documento retornado em resposta a uma consulta é relevante, e 0 caso contrário, $|D|$ é o número total de documentos retornados em resposta a uma consulta, e k_8 e k_9 são parâmetros de ajuste assumindo os valores 7 e 0,982 respectivamente.

3.2.5 Escolha do Melhor Indivíduo

Como anteriormente mencionado na Seção 3.2, foi utilizado um conjunto de validação para ajudar na escolha de funções de ordenação, levando em consideração o resultado obtido na fase de treino e de validação, de modo a escolher um indivíduo capaz de generalizar para consultas não conhecidas e não escolher um indivíduo muito especializado para o conjunto de treino.

Geralmente, como visto nos trabalhos de Fan *et al.* [12, 13, 14, 15], escolhe-se o melhor indivíduo como base no desempenho obtido na fase de validação. No entanto, uma abordagem mais balanceada deveria considerar também o desempenho obtido por um indivíduo na fase de treino. O trabalho de Lacerda *et al.* [22], apresenta uma estratégia para a escolha do melhor indivíduo, levando em consideração o desempenho médio de um indivíduo obtido em ambos os conjuntos de treino e validação, considerando também o desvio padrão da média aritmética destes valores, conforme descrito na equação

$$AVG_\sigma = \underset{i}{argmax}(\overline{f}_i - \sigma_i) \tag{3.3}$$

onde $\overline{f}_i$ é a média entre os desempenhos de um indivíduo i na fase de treino e validação, assim como σ_i é o desvio padrão correspondente. A este método foi atribuído o nome de AVG_σ. O indivíduo com o valor mais alto de AVG_σ será escolhido como sendo o melhor.

Não obstante ser uma proposta balanceada para a escolha do melhor indivíduo, alguns experimentos realizados neste trabalho de pesquisa mostraram que este método pode levar a um indivíduo que não possui o melhor desempenho. Por exemplo, supondo-se o desempenho obtido por três indivíduos A, B e C na fase de treino como sendo 25,0, 30,0 e 50,0 respectivamente, e na fase de validação como sendo 25,0, 25,0 e 25,0 para os mesmos indivíduos A, B e C, o valor de AVG_σ para todos os indivíduos seria de 25,0, já que o

desempenho médio dos indivíduos A, B e C seria de 25,0, 27,5 e 37,5, e o desvio padrão de 0,0, 2,5 e 12,5 respectivamente.

Por causa deste comportamento, que assinalaria o mesmo valor para os indivíduos A, B e C na escolha do melhor indivíduo, foi proposto um método similar, que também considera a dispersão entre os valores de treino e validação, mas usa a soma destes valores ao invés do valor médio. Este método foi denominado SUM_σ. Mais formalmente, seja t_i o desempenho de um indivíduo i na fase de treino, seja v_i o desempenho do indivíduo i na fase de validação, e seja σ_i o desvio padrão correspondente a esses dois valores. A escolha do melhor indivíduo é feita de acordo com a equação

$$SUM_\sigma = \underset{i}{argmax}((t_i + v_i) - \sigma_i) \tag{3.4}$$

Dessa forma, considerando o mesmo exemplo apresentado acima, SUM_σ assumiria os valores de 50,0, 52,5 e 62,5 para os mesmos indivíduos A, B e C, que levaria à escolha do indivíduo C como sendo o melhor indivíduo.

Os experimentos descritos no Capítulo 4 foram realizados com ambos os métodos AVG_σ e SUM_σ e foram reportados os resultados obtidos com o melhor método em cada execução.

Id	Terminal	Descrição
t_{01}	tf	Freqüência bruta do termo (número de vezes que um termo ocorre em um documento) [36]
t_{02}	$1 + \log(tf)$	Logaritmo natural da freqüência bruta do termo como apresentado em [48], usado para suavizar a influência da freqüência do termo
t_{03}	$0.5 + \frac{0.5+tf}{\max tf}$	Freqüência do termo em um documento normalizada pela freqüência máxima de um termo em um documento, e mais adiante ajustada para ficar situada entre 0,5 e 1,0 [3, 36]
t_{04}	$\frac{1+\log(tf)}{1+\log(\text{avg}tf)}$	Freqüência do termo em um documento normalizada pela freqüência média de termos em um documento, como definido em [40]. Parte da fórmula do esquema de ponderação do sistema SMART como definido em [7]
t_{05}	$\frac{(k_1+1)tf}{(k_1((1-b)+b\times dl/\text{avg}dl)+dl)+tf}$	Parte da função de ordenação do sistema Okapi BM25 com os componentes de freqüência do termo *tfc* e de normalização *nc* [32]
t_{06}	$\log\left(\frac{N}{df}\right)$	Inverso da freqüência do documento (*idf*) [36]
t_{07}	$\log\left(\frac{N}{df}+1\right)$	Uma alternativa para o inverso da freqüência do documento (*idf*) como apresentado em [48]
t_{08}	$\log\left(\frac{N-df+0.5}{0.5}\right)$	Uma variação para o peso definido por Robertson-Sparck Jones [31], que suaviza as diferenças entre os valores de *df*.
t_{09}	$w^{(1)} = \log\left(\frac{N-df+0.5}{df+0.5}\right)$	Peso definido por Robertson-Sparck Jones [31, 32, 33]
t_{10}	$\log\left(\frac{N-df}{df}\right)$	Uma alternativa probabilística para o inverso da freqüência do documento [31, 36]
t_{11}	$\frac{\log\frac{N+0.5}{df}}{\log N+1}$	Parte da função de ordenação do sistema INQUERY para calcular a crença de um termo dentro de um documento como definido em [1]
t_{12}	$\frac{1}{\sqrt{\sum_{i=0}^{2} w_{i,j}^2}}$	Normalização do cosseno onde $w_{i,j}^2$ is $t_{01} \times t_{07}$ [36, 48]
t_{13}	$\frac{1}{\sqrt{\sum_{i=0}^{2} w_{i,j}^2}}$	Normalização do cosseno onde $w_{i,j}^2$ is $t_{02} \times t_{07}$ [36, 48]
t_{14}	dl	Normalização do tamanho do documento (em *bytes*). Técnica de normalização usada em coleções de documentos antigas tais como os repositórios baseados em OCR (digitalizados) [40]
t_{15}	$\frac{1}{(1-slope)+slope\times\frac{\text{avg}t_{13}}{t_{13}}}$	Normalização pivoteada do cosseno com o terminal t_{13} como definido em [40]
t_{16}	$\frac{1}{(1-slope)\times\text{avg}dl+slope\times dl}$	Normalização pivoteada do tamanho do documento como definido em [40]
t_{17}	$\frac{1}{(1-slope)\times pivot+slope\times\#\text{ of unique terms}}$	Normalização pivoteada baseada nos termos distintos de um documento [40], onde *pivot* é a média do número de termos distintos de um documento em toda a coleção
t_{18}	$\frac{1}{\left(k_1\times(1-b)+b\times\frac{dl}{\text{avg}dl}\right)+tf}$	Fator de normalização presente na função de ordenação do sistema Okapi BM25, como definido em [32, 33]
t_{19}	$\frac{(k_3+1)\times qtf}{k_3+qtf}$	Fator relativo à consulta presente na função de ordenação do sistema Okapi BM25, como definido em [32, 33]
t_{20}	$0.5 + \frac{0.5+qtf}{\max qtf}$	Freqüência do termo na consulta normalizada, como definido em [3, 36]

Tabela 3.1: Terminais utilizados pela abordagem CCA no arcabouço GP.

Capítulo 4

Experimentos

Neste capítulo são apresentadas as coleções de referência utilizadas para avaliação da abordagem proposta por este trabalho, assim como a configuração e os parâmetros utilizados nos experimentos, as abordagens utilizadas como linha de base de comparação e os resultados obtidos a partir dos experimentos realizados.

O ambiente de execução utilizado nos experimentos é composto de duas máquinas Intel(R) Xeon(TM) CPU 3.20GHz, uma com 4Gbytes e a outra com 3Gbytes de memória principal.

4.1 Coleções de Referência

Para avaliação da abordagem de componentes combinados proposta neste trabalho, foram utilizadas as coleções TREC-8 [45] e WBR99[1]. Algumas características destas coleções são apresentadas na Tabela 4.1.

As seções 4.1.1 e 4.1.2 abaixo descrevem mais detalhadamente as coleções TREC-8 e WBR99.

4.1.1 TREC-8

A coleção TREC-8 tem aproximadamente 528 mil documentos e 737 mil termos distintos. Os documentos são extraídos de diversas fontes tais como *The Financial Times* (1991-1994), *Federal Register* (1994), *Foreign Broadcast Information Service* e *LA Times*.

[1]Disponível em http://www.linguateca.pt/Repositorio/WBR-99/

Característica	TREC-8	WBR99
Número de Documentos	528.155	5.939.061
Número de Termos Distintos	737.833	2.669.965
Número de Consultas	50	49
Média de Termos por Consulta	10,8	1,94
Tamanho	2Gb	16Gb

Tabela 4.1: Características das Coleções de Referência.

Nossos experimentos foram realizados utilizando-se 50 consultas, chamadas tópicos, numeradas de 401 a 450. Há na coleção TREC-8, para cada tópico, um conjunto de documentos relevantes que podem ser utilizados para testar novos algoritmos de ordenação. As consultas em nossos experimentos foram geradas automaticamente usando o título, a descrição e a narrativa de cada tópico, tais como "*what is happening in the field of behavioral genetics the study of the relative influence of genetic and environmental factors on an individuals behavior or personality*", "*what other countries besides the United States are considering or have approved women as clergy persons*" e "*identify instances of attacks on humans by Africanized killer bees*". As consultas foram divididas em três grupos: tópicos 401-420 foram usados na fase de treino, tópicos 421-430 foram utilizados na fase de validação e, finalmente, tópicos 431-450 foram usados na fase de teste para comparação das funções de ordenação descobertas com as funções utilizadas como linha de base.

4.1.2 WBR99

A coleção WBR99, que foi previamente utilizada em trabalhos como [28], contém um repositório de páginas Web, um conjunto de consultas e um conjunto de documentos relevantes associados a cada consulta. Os documentos relevantes foram gerados através da avaliação manual de documentos retornados por um processador de consultas baseado no modelo vetorial. A coleção possui quase 6 milhões de páginas Web, coletadas da Web brasileira (o domínio *.br*) e quase 2,7 milhões de termos distintos.

Esta coleção representa uma fotografia interessante da Web brasileira, que é provavelmente tão variada em conteúdo e estrutura de ligações entre as páginas quanto a própria Web. Dessa forma, acredita-se que a coleção WBR99 representa um cenário realista para nossos experimentos. As consultas 1-20 foram utilizadas na fase de treino, as consultas de

21-30 foram usadas na fase de validação e as consultas 31-50, exceto consulta 35, foram utilizadas na fase de teste para avaliação da abordagem. As expressões “detran”, “emprego”, “filmes”, “história do brasil”, “receita federal” e “vestibular” são exemplos de consultas existentes na coleção WBR99.

4.2 Configuração dos Experimentos

Quase todos os parâmetros e configurações utilizadas no arcabouço GP foram baseadas nos trabalhos de Koza [21] e de Fan *et al.* [13]. Foi definida uma população inicial de 200 indivíduos gerados aleatoriamente através do método *Ramped-half-and-half*, descrito na Seção 2.2.4. Foram realizados testes com as coleções TREC-8 e WBR99, utilizando-se duas funções de aptidão descritas na Seção 3.2.4: MAP e FFP4. A função MAP foi mais efetiva para a coleção TREC-8, enquanto que FFP4 foi a melhor escolha para a WBR99.

Devido à estabilidade dos resultados após 30 gerações, verificada nos primeiros experimentos, este valor foi definido como critério de terminação do processo evolutivo. Foram utilizados os operadores genéticos de cruzamento, reprodução e mutação, definidos com taxas de 90%, 5% e 5% respectivamente. A semente aleatória utilizada foi o valor 1234567890. Os terminais e funções utilizados foram aqueles definidos na Seção 3.2.1.

A profundidade máxima da árvore representando um indivíduo foi variada de 3 a 12, isto é, para cada coleção, 10 experimentos foram realizados, um para cada valor de profundidade máxima. Além disso, na fase de validação, foram selecionados em cada geração os 20 melhores indivíduos, totalizando 600 indivíduos ao final do processo evolutivo, candidatos ao melhor indivíduo descoberto.

4.3 Avaliação dos Experimentos

Os resultados obtidos nos experimentos realizados com a abordagem CCA foram comparados com outras três abordagens: (i) Okapi BM25 [32], como descrito na Equação 2.6 (usando os parâmetros k_1=1.2, k_3=1000, b=0.75), (ii) modelo de espaço vetorial clássico com esquema de ponderação tf-idf ($w_{i,q}$=1 e $w_{i,j}$ definido conforme Equação 2.4), e (iii) a abordagem GP proposta por Fan *et al.* em [13] (denominada de FAN-GP neste trabalho).

Inicialmente, foram escolhidas, como linha de base de comparação, duas abordagens para cada coleção. Para a TREC-8, os dois melhores resultados foram obtidos com BM25 e FAN-GP. Já para a coleção WBR99, as abordagens utilizadas como linha de base foram

TF-IDF e FAN-GP.

Apesar do trabalho original de Fan *et al.* não ter variado a profundidade máxima da árvore que representa um indivíduo no arcabouço GP, assim como realizado para CCA, este parâmetro também foi variado para FAN-GP para garantir uma comparação justa. Além disso, todos os demais parâmetros do arcabouço genético foram definidos da mesma forma para CCA e FAN-GP, como descrito na Seção 4.2. A taxa de mutação utilizada foi de 5%, apesar do trabalho original de Fan *et al.* não ter utilizado o operador de mutação. Os experimentos com FAN-GP para as coleções de referência utilizadas, todavia, mostraram que um fator de mutação de 5% produzia melhores resultados do que desconsiderar esta operação genética. Em virtude disso, foi adotado o operador de mutação, com uma taxa de 5%, também para FAN-GP.

Os resultados descritos na Seção 4.4 para CCA e FAN-GP foram os obtidos com a melhor função de ordenação descoberta nas 10 execuções realizadas, considerando diferentes valores para a profundidade máxima da árvore (de 3 a 12). A seleção das melhores funções de ordenação foi realizada conforme descrito na Seção 3.2.5. Foi escolhido o melhor resultado entre os indivíduos obtidos pelos métodos de seleção AVG_σ e SUM_σ.

Para avaliação da efetividade da abordagem CCA contra as linhas de base, foram utilizadas, como em [44], a média das precisões médias (MAP), a precisão obtida em determinadas posições do *ranking* (*document cutoff values*) e R-precision. Foram também traçadas as curvas de precisão × revocação.

4.4 Resultados

Nesta seção, serão apresentadas as melhores funções de ordenação descobertas, bem como todos os resultados obtidos nos experimentos realizados para as coleções TREC-8 e WBR99. Foram utilizadas as consultas da fase de teste, que foram processadas pela melhor função de ordenação descoberta por CCA para cada coleção e pelas abordagens da linha de base, para avaliação e comparação dos resultados.

4.4.1 Melhores Funções de Ordenação Descobertas

Ao final dos experimentos, foi escolhida a melhor função de ordenação descoberta para cada coleção. Em ambas as coleções, TREC-8 e WBR99, a melhor função de ordenação foi escolhida através do método SUM_σ.

Para a coleção TREC-8, foi considerado como melhor indivíduo aquele que obteve o melhor desempenho global, considerando a efetividade média (MAP) e ao mesmo tempo a efetividade no topo do *ranking*. A melhor função de ordenação, apresentada na Figura 4.1, foi descoberta com a profundidade máxima para a árvore de um indivíduo configurada para o valor 5.

```
(*
    (* (log t08) (+ t05 t07))
    (+ (+ (* (+ t19 t05) (+ t07 t06))
          (* (+ t06 t02) (* t16 t18)))
       (/ t07 t19))
)
```

Figura 4.1: Melhor função de ordenação descoberta por CCA para a coleção TREC-8.

Já para a coleção WBR99, também considerando o melhor desempenho no geral e no topo do *ranking*, a melhor função de ordenação, apresentada na Figura 4.2, foi descoberta com o valor 8 para a profundidade máxima da árvore de um indivíduo. Um detalhe interessante é que, diferentemente da TREC-8, o indivíduo com melhor desempenho médio (MAP) para a WBR99 não foi aquele de melhor efetividade no topo do *ranking*.

```
(+ (+ (+ 99.09 t11)
   (+ (* (* t07 t10)
      (* t05 (* (+ (* t07 t10) (+ t08 t10)) (* t12 t01))))
      (* (* t07 t10)
      (* t05 (* (+ (* t02 t04) (+ t08 t10)) (* t12 t01))))))
   (+ (* t12 t01)
      (* (* t07 t10)
      (* t05 (* (+ (/ t08 t20) (+ t08 t10)) (* t12 t01)))))
)
```

Figura 4.2: Melhor função de ordenação descoberta por CCA para a coleção WBR99.

Cada função descoberta possui um terminal relativo ao componente de freqüência do termo (tfc) para consultas — t_{19} para TREC-8 e t_{20} para WBR99 —, pelo menos três

diferentes terminais relativos ao componente de freqüência na coleção (cfc) — t_{06}, t_{07}, t_{08}, t_{09}, t_{10}, ou t_{11} — e, além disso, considerando o componente de normalização (nc), dois terminais baseados em normalização pivoteada para a coleção TREC-8 — t_{16} e t_{18} — e um terminal baseado na normalização do cosseno para a WBR99 — t_{12}. A Figura 4.1 apresenta também algumas partes da função de ordenação BM25 — t_{05}, t_{18} e t_{19} — comprovadamente efetiva na tarefa de recuperação de informação baseada em conteúdo. E a Figura 4.2 apresenta componentes presentes em esquemas de ponderação tf-idf tais como t_{01}, t_{02}, t_{07} e t_{12}. Isto mostra que CCA foi capaz de reusar bons componentes da melhor função de ordenação usada como linha de base para cada coleção e combiná-los com outros componentes para gerar uma função de ordenação ainda mais efetiva do que as funções utilizadas como linha de base.

4.4.2 Efetividade da Recuperação

A Figura 4.3 apresenta as curvas de precisão × revocação da abordagem CCA e das abordagens utilizadas como linha de base de comparação, para as coleções TREC-8 e WBR99. Como é possível notar, CCA alcança valores de precisão melhores do que tf-idf, BM25 e FAN-GP ao longo de quase todos os níveis de revocação, isto é, CCA tem um desempenho médio melhor do que todas as outras abordagens nas duas coleções avaliadas. Para a coleção WBR99, a abordagem FAN-GP obtém um melhor desempenho do que CCA, quando os primeiros documentos são recuperados no topo do *ranking*. A efetividade de FAN-GP, todavia, piora sensivelmente à medida que os demais documentos são recuperados.

A Tabela 4.2 apresenta detalhadamente os resultados obtidos pela abordagem CCA para a coleção TREC-8. Como pode ser visto, CCA alcança melhores resultados do que BM25 e FAN-GP. Na precisão média, CCA atingiu 16,40%, superando BM25 em 40,87% e FAN-GP em 14,00%. A significância estatística destes ganhos foi comprovada através do teste t de Student com p<0,06 e os níveis de confiança apresentados correspondem a (1-p). Além disso, ganhos também são observados para a precisão nos 5 primeiros documentos do topo do *ranking* (p@5): CCA superou BM25 e FAN-GP em 18,52%. A abordagem CCA também conseguiu melhorar o desempenho na *R-precision* em mais de 26% sobre BM25 e 15% sobre FAN-GP.

Para a coleção WBR99, também foram obtidos ganhos em relação às abordagens utilizadas como linha de base para comparação. A Tabela 4.3 apresenta o valor de 16,68% obtido por CCA na precisão média. Este valor supera em 21,67% tf-idf e em 24,85%

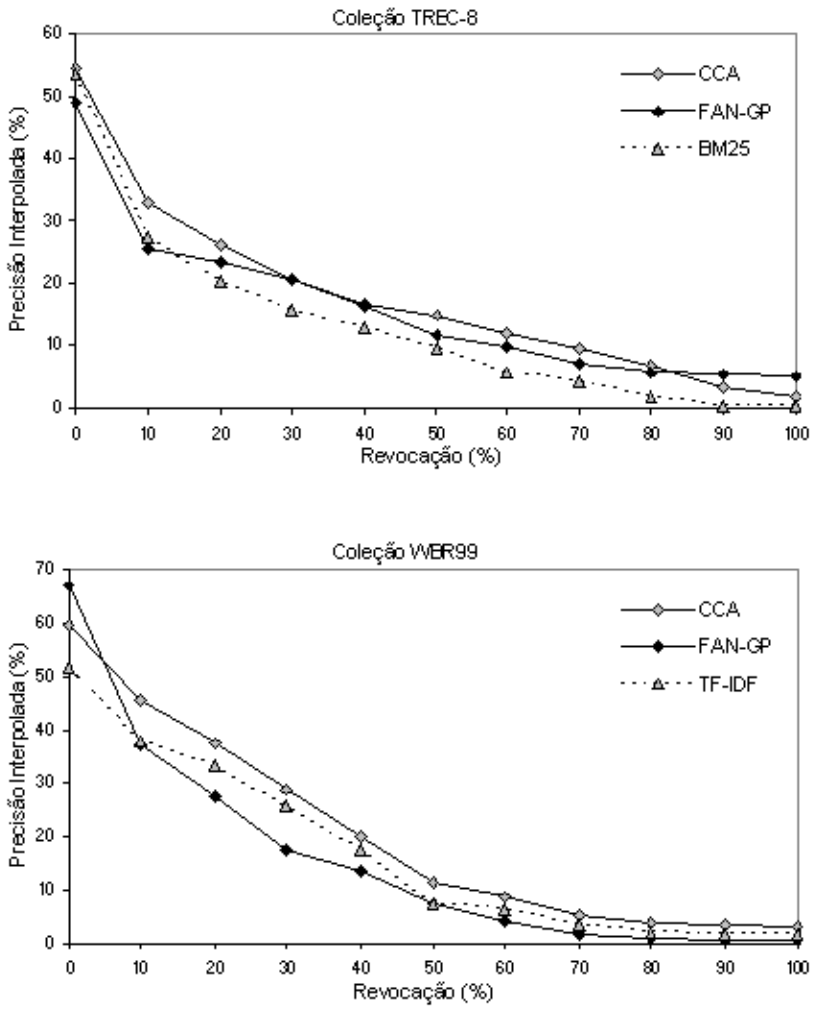

Figura 4.3: Curvas de precisão × revocação para as coleções TREC-8 e WBR99.

FAN-GP. A significância estatística destes ganhos foi comprovada através do teste t de Student com $p<0{,}05$ e os níveis de confiança apresentados correspondem a (1-p). Pode ser observado também que FAN-GP não conseguiu superar tf-idf para esta coleção. No topo do *ranking*, CCA superou tf-idf e FAN-GP em 59,10% e 2,94% respectivamente. A abordagem CCA superou tf-idf e FAN-GP em mais de 8% e 12% respectivamente na *R-precision*.

A Figura 4.4 apresenta a precisão média para cada consulta. Para a coleção TREC-8, pode ser observado que os resultados de CCA melhora os resultados sobre FAN-GP em algumas consultas, nas quais FAN-GP não havia ganhado de BM25, como por exemplo, nas consultas 438, 443, 445, 449 e 450. A figura também mostra que CCA somente apresenta

Nível	TREC-8				
	Linhas de Base		CCA		
	BM25	**FAN-GP**	*Precisão*	*Ganho sobre BM25*	*Ganho sobre FAN-GP*
Em 5 documentos	27,000	27,000	32,000	+18,52%	+18,52%
Em 10 documentos	29,000	25,500	31,500	+8,62%	+23,53%
Em 15 documentos	25,333	24,667	27,000	+6,58%	+9,46%
Em 20 documentos	23,750	22,750	25,000	+5,26%	+9,89%
Em 30 documentos	20,167	21,333	22,167	+9,92%	+3,91%
Em 100 documentos	15,050	15,150	16,050	+6,64%	+5,94%
Em 200 documentos	11,900	11,775	12,025	+1,05%	+2,12%
Em 500 documentos	7,410	7,160	7,650	+3,24%	+6,84%
Em 1000 documentos	4,910	4,505	5,000	+1,83%	+10,99%
R-precision	16,116	17,703	20,449	+26,89%	+15,51%
Precisão Média	11,643	14,388	16,402	+40,87%	+14,00%
Nível de Confiança				*94,05%*	*98,19%*

Tabela 4.2: Dados de precisão da abordagem CCA para a coleção TREC-8

Nível	WBR99				
	Linhas de Base		CCA		
	TF-IDF	**FAN-GP**	*Precisão*	*Ganho sobre TF-IDF*	*Ganho sobre FAN-GP*
Em 5 documentos	23,157	35,790	36,842	+59,10%	+2,94%
Em 10 documentos	26,315	32,632	32,632	+24,00%	0,00%
Em 15 documentos	30,175	29,474	29,123	-3,49%	-1,19%
Em 20 documentos	32,105	26,842	27,368	-14,75%	+1,96%
Em 30 documentos	25,263	21,579	25,965	+2,78%	+20,33%
Em 100 documentos	13,421	10,737	13,632	+1,57%	+26,96%
Em 200 documentos	9,000	7,553	8,790	-2,34%	+16,38%
Em 500 documentos	3,726	3,516	3,737	+0,28%	+6,29%
Em 1000 documentos	1,968	1,879	2,005	+1,87%	+6,73%
R-precision	20,607	19,830	22,294	+8,19%	+12,43%
Precisão Média	13,710	13,361	16,681	+21,67%	+24,85%
Nível de Confiança				*96,96%*	*96,04%*

Tabela 4.3: Dados de precisão da abordagem CCA para a coleção WBR99

pior desempenho do que BM25 e FAN-GP em consultas onde a precisão média já é muito baixa tais como 432, 435, 437, 440 e 448. Para a coleção WBR99, CCA obtém melhor desempenho do que as abordagens *tf-idf* e FAN-GP para as consultas 31, 32, 33, 34, 44,

47, 48, 49 e 50. CCA fica próximo dos melhores resultados nas consultas 39, 40 e 46. A abordagem CCA não consegue superar FAN-GP para as consultas 37, 39, 41, 42, 43 e 45.

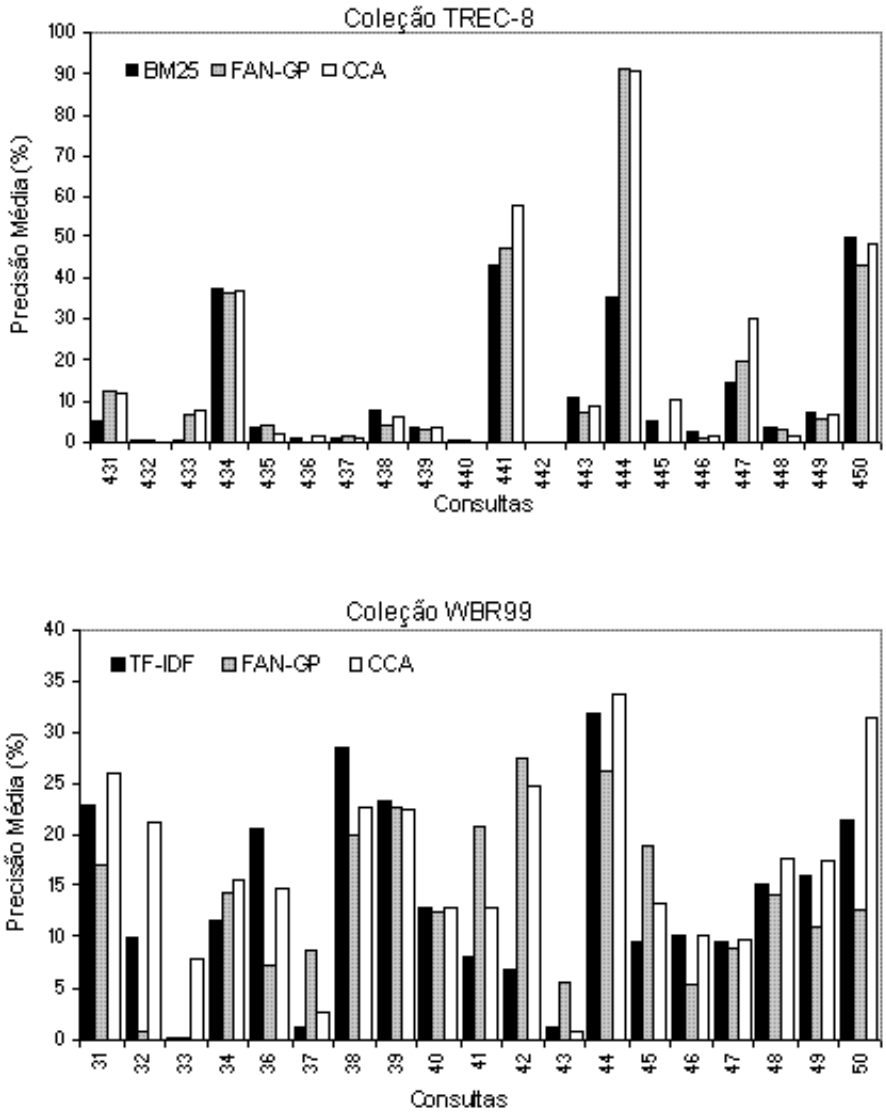

Figura 4.4: Histograma de comparação da efetividade consulta-a-consulta para as coleções TREC-8 e WBR99.

4.4.3 Profundidade Máxima da Árvore

A Tabela 4.4 apresenta a comparação dos resultados obtidos de precisão média entre as abordagens FAN-GP e CCA para as coleções TREC-8 e WBR99. Como pode ser observado,

a abordagem CCA obteve o melhor resultado médio, bem como os melhores resultados no geral: profundidades 5 e 7 para a TREC-8 e profundidades 8 e 11 para a WBR99.

Profundidade Máxima	**TREC-8**		**WBR99**	
	FAN-GP	**CCA**	**FAN-GP**	**CCA**
Profundidade 3	7,767	15,982	10,457	10,883
Profundidade 4	13,800	15,095	7,568	13,255
Profundidade 5	9,844	16,402	7,884	13,162
Profundidade 6	11,477	15,244	9,856	8,819
Profundidade 7	12,887	16,297	13,361	13,703
Profundidade 8	11,022	15,631	12,485	16,681
Profundidade 9	14,125	14,869	13,344	9,762
Profundidade 10	13,429	14,434	11,324	15,262
Profundidade 11	13,986	15,007	9,086	17,200
Profundidade 12	14,388	14,886	12,441	13,110
Média	12,273	15,385	10,781	13,184

Tabela 4.4: Comparativo dos resultados de precisão média entre as abordagens CCA e FAN-GP para as coleções TREC-8 e WBR99

4.5 Análise dos Experimentos

Nesta seção é realizada uma análise mais detalhada dos experimentos e da abordagem CCA. O processo evolutivo CCA é analisado em relação ao problema de *overfitting*, bem como é verificada também a ocorrência de cada terminal CCA nas funções descobertas.

4.5.1 Processo Evolutivo

A Figura 4.5 mostra o processo evolutivo CCA ao longo de 30 gerações, considerando os 20 melhores indivíduos, para as coleções TREC-8 e WBR99. Para cada geração, foram avaliados, de acordo com a função de aptidão utilizada, os 20 melhores indivíduos. Como pode ser observado, as funções descobertas pela abordagem CCA convergem mais rapidamente do que as descobertas pela abordagem FAN-GP. As figuras também mostram que as curvas CCA comportam-se de modo mais similar. Isto é, apesar do fato dos conjuntos de treino, validação e teste apresentarem diferentes valores de aptidão, as curvas de validação e teste tendem a seguir o comportamento da curva de treino. Na abordagem FAN-GP, as curvas de validação e teste não seguem o comportamento da curva de treino, indicando

overfitting. Isto pode ser observado tanto na coleção TREC-8 quanto na coleção WBR99. Outro indicador da ocorrência de *overfitting* é a distância entre as curvas de treino e as demais curvas. Ou seja, quanto maior a distância entre as curvas, maior o *overfitting*. Observando as duas figuras, pode ser comprovado que as curvas CCA estão mais próximas uma das outras do que as curvas da abordagem FAN-GP, indicando que na abordagem CCA o problema do *overfitting* ocorre de modo menos acentuado do que em FAN-GP.

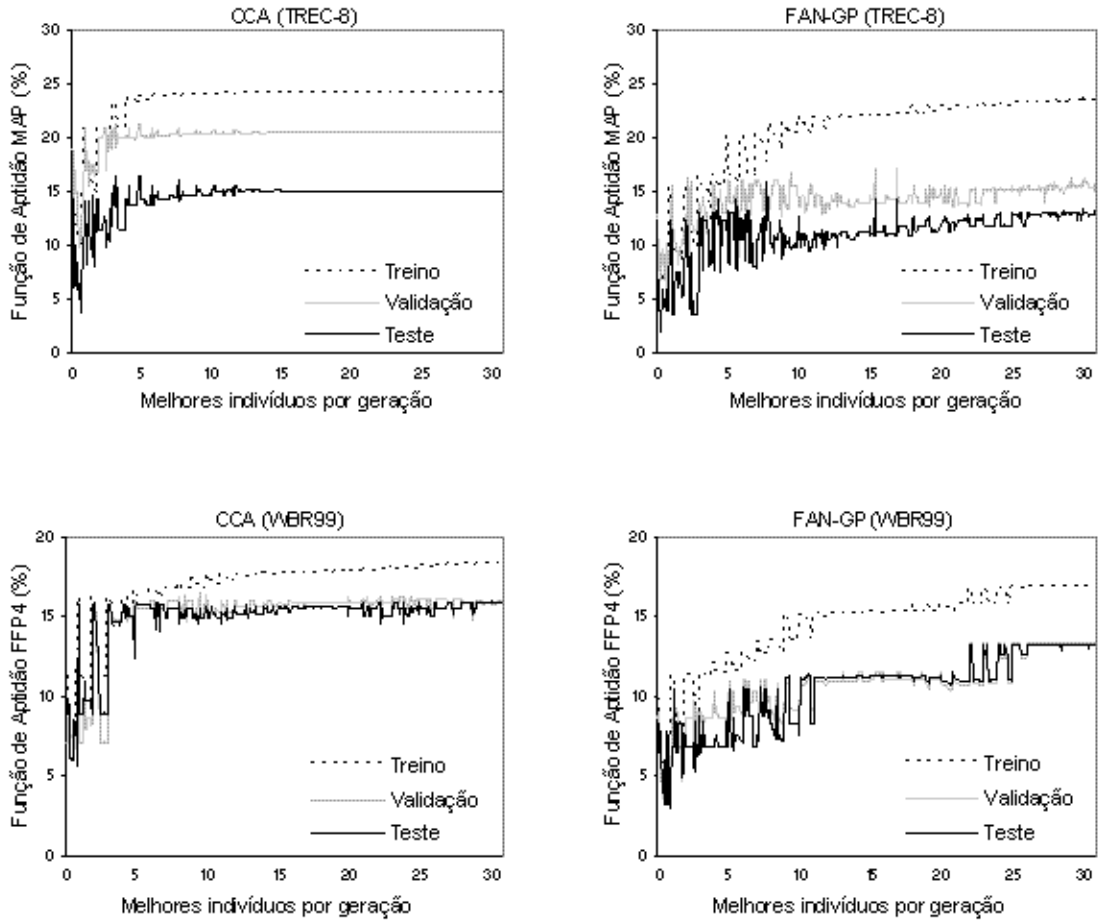

Figura 4.5: Processo evolutivo CCA considerando os 20 melhores indivíduos em 30 gerações para as coleções TREC-8 e WBR99.

Além disso, considerando-se especificamente os experimentos que produziram os melhores indivíduos para as TREC-8 e WBR99, com profundidade máxima de 5 e 8 respectivamente, foi possível comprovar através da investigação do processo evolutivo que além do melhor indivíduo descoberto para cada coleção — Figuras 4.1 e 4.2 — no mínimo outros 30 indivíduos superaram as abordagens utilizadas como linha de base. Isto indica que

CCA, em um mesmo experimento, é capaz de encontrar diversas funções de ordenação efetivas para uma determinada coleção. A Figura 4.6 ilustra este resultado apresentando o desempenho dos 20 melhores indivíduos descobertos em cada experimento que produziu a melhor função de ordenação de cada coleção (profundidade máxima do indivíduo configurada como 5 para a TREC-8 e 8 para a WBR99). É possível notar que todos os 20 melhores indivíduos descobertos superaram as linhas de base em ambas as coleções.

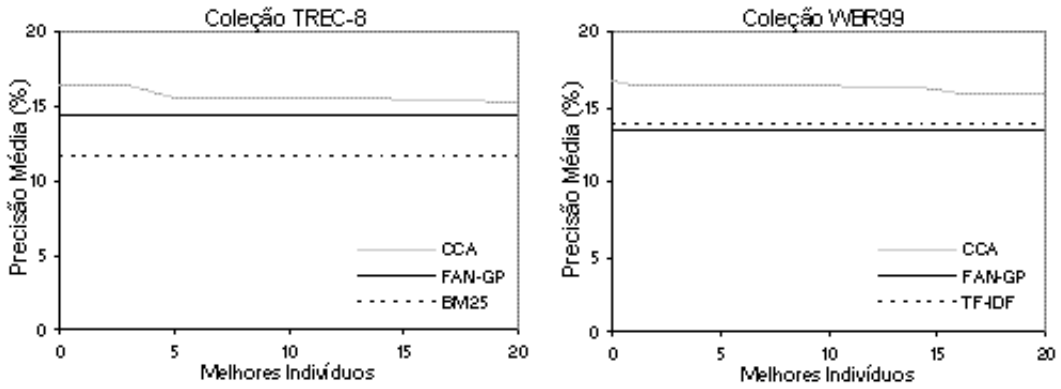

Figura 4.6: Distribuição dos valores de precisão média para os 20 melhores indivíduos dos experimentos que produziram a melhor função de ordenação para cada coleção (TREC-8 e WBR99).

4.5.2 Análise dos Terminais

No geral, os terminais significativos extraídos de funções de ordenação conhecidas que foram utilizados pela abordagem CCA, são capazes de produzir boas funções. No entanto, é necessário analisar a contribuição de cada terminal nas funções de ordenação descobertas, para que se possa verificar qual papel é desempenhado por cada terminal. Isto é, deseja-se saber quais terminais estão sendo realmente utilizados nas funções geradas pela abordagem CCA. Para realizar esta análise, a melhor função de ordenação descoberta em cada um dos 10 experimentos de cada coleção foi examinada para se verificar a ocorrência de cada terminal, visando-se descobrir os mais freqüentes. Supõe-se, dessa forma, que o número de ocorrências dos terminais nas funções geradas, possa indicar uma medida de avaliação da qualidade ou influência de cada terminal. Esta análise pode subsidiar, em novos experimentos no futuro, a redução do conjunto de terminais, visando simplificar e otimizar o processo de descoberta de funções de ordenação.

As Tabelas 4.5 e 4.6 apresentam uma análise preliminar da importância de cada terminal, baseada na ocorrência de cada um nas melhores funções de ordenação descobertas pelos 10 experimentos realizados em cada coleção. A Tabela 4.5 apresenta a análise para a coleção TREC-8 e a Tabela 4.6 para a coleção WBR99. São apresentados o número de ocorrências de um terminal em cada experimento de acordo com a profundidade máxima configurada, assim como o número total de ocorrências de um terminal em todos os experimentos (*Ocorrências / Total*) e o número de experimentos nos quais um terminal foi encontrado (*Ocorrências / Exps.*).

Coleção TREC-8												
Terminais	**Profundidade Máxima**										**Ocorrências**	
	03	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**Total**	**Exps**
t_{01}	-	-	-	-	-	-	-	-	1	-	1	1
t_{02}	-	-	1	-	-	-	-	-	-	-	1	1
t_{03}	1	-	-	-	1	-	-	-	1	-	3	3
t_{04}	-	2	-	1	1	1	1	1	1	3	11	8
t_{05}	1	1	2	1	2	1	-	1	5	2	16	9
t_{06}	-	-	2	3	-	2	-	-	2	-	9	4
t_{07}	1	-	3	-	1	-	1	-	5	2	13	6
t_{08}	-	-	1	-	-	1	-	1	-	-	3	3
t_{09}	1	1	-	-	2	1	1	1	-	5	12	7
t_{10}	-	-	-	-	-	2	-	2	6	1	11	4
t_{11}	-	4	-	2	-	1	-	1	-	1	9	5
t_{12}	-	-	-	-	-	-	-	-	-	-	-	-
t_{13}	-	-	-	-	1	-	-	-	-	2	3	2
t_{14}	-	-	-	-	-	-	-	1	-	-	1	1
t_{15}	-	-	-	-	-	-	1	-	-	1	2	2
t_{16}	-	-	1	-	-	-	-	-	-	1	2	2
t_{17}	1	-	-	-	-	-	-	-	-	-	1	1
t_{18}	-	-	1	1	-	-	-	1	-	-	3	3
t_{19}	-	-	2	-	1	-	-	-	2	2	7	4
t_{20}	1	-	-	-	-	-	-	-	4	1	6	3

Tabela 4.5: Total de ocorrências dos terminais CCA nos experimentos realizados para a coleção TREC-8 variando-se a profundidade máxima dos indivíduos.

Para a coleção TREC-8 o terminal mais freqüente foi t_{05}, que é uma parte importante da função de ordenação do sistema Okapi BM25, comprovadamente efetiva em coleções TREC. O terminal ocorreu 16 vezes em 9 dos 10 experimentos realizados com a coleção TREC-8. Em segundo lugar, o terminal t_{04}, que faz parte da função de ordenação do

sistema SMART, ocorreu em 8 dos 10 experimentos.

Coleção WBR99												
Terminais	**Profundidade Máxima**										**Ocorrências**	
	03	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**Total**	**Exps**
t_{01}	-	1	1	-	-	4	-	1	2	-	9	5
t_{02}	-	-	-	2	1	1	-	-	-	7	11	4
t_{03}	-	-	-	-	-	-	1	-	-	2	3	2
t_{04}	-	-	1	-	2	1	1	-	2	2	9	6
t_{05}	1	-	1	-	1	3	-	-	1	-	7	5
t_{06}	-	-	1	-	-	-	3	-	-	-	4	2
t_{07}	1	-	-	-	3	4	1	-	-	2	11	5
t_{08}	-	2	1	-	1	4	1	-	-	2	11	6
t_{09}	-	-	-	5	-	-	-	2	-	1	8	3
t_{10}	-	-	1	3	-	7	2	-	5	4	22	6
t_{11}	-	-	-	-	1	1	-	-	-	2	4	3
t_{12}	1	2	-	-	-	4	1	1	3	-	12	6
t_{13}	1	1	2	-	1	-	-	-	-	1	6	5
t_{14}	-	-	-	-	-	-	-	-	-	-	-	-
t_{15}	-	-	-	4	3	-	-	-	-	-	7	2
t_{16}	-	-	-	-	-	-	1	-	-	-	1	1
t_{17}	-	2	-	-	-	-	-	-	-	-	2	1
t_{18}	1	2	-	3	-	-	-	-	-	-	6	3
t_{19}	-	-	1	-	-	-	-	-	-	6	7	2
t_{20}	-	-	-	-	-	1	1	-	1	2	5	4

Tabela 4.6: Total de ocorrências dos terminais CCA nos experimentos realizados para a coleção WBR99 variando-se a profundidade máxima dos indivíduos.

Já nos experimentos da coleção WBR99, o terminal mais freqüente foi t_{10}, que é uma alternativa probabilística para *idf*. O terminal t_{10} ocorreu 22 vezes em 6 dos 10 experimentos. Além deste, os terminas t_{04}, t_{08} e t_{12} também ocorreram em 6 dos 10 experimentos da coleção WBR99.

As Figuras 4.7 e 4.8 comparam a ocorrência dos terminais em cada coleção. Como pode ser notado na Figura 4.7, o terminal t_{12}, que foi um dos mais freqüentes para a WBR99, nem figurou nas melhores funções geradas para a coleção TREC-8. Estas comparações mostram que a abordagem CCA foi capaz de escolher os terminais mais apropriados para as características de cada coleção e utilizá-los de forma efetiva para gerar boas funções de ordenação. Já na Figura 4.8, pode ser observado que os terminais t_{04}, t_{05}, t_{07} e t_{09} apareceram em 60% ou mais dos experimentos realizados com a coleção TREC-8. No

entanto, isto não se repetiu para a WBR99. Dois terminais tiveram o mesmo percentual de ocorrência nas duas coleções: t_{15} e t_{18}.

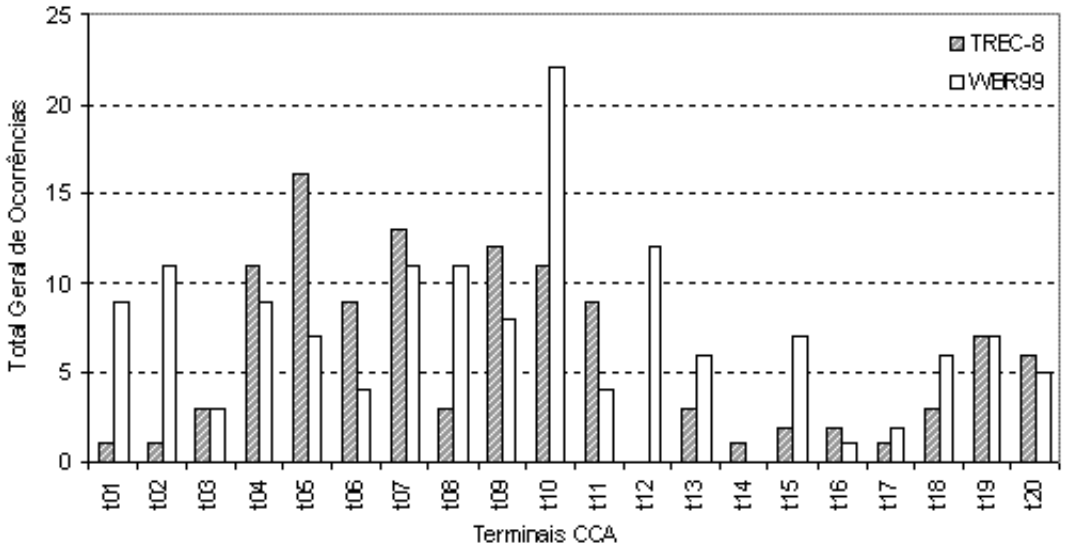

Figura 4.7: Histograma comparativo do total geral de ocorrências dos terminais CCA nas melhores funções de ordenação descobertas para as coleções TREC-8 e WBR99.

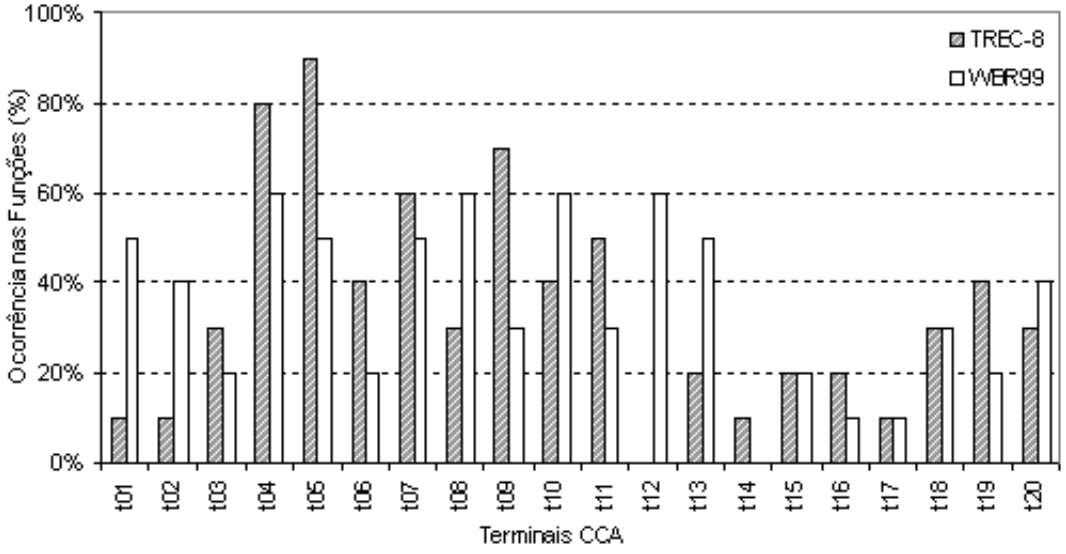

Figura 4.8: Histograma comparativo do número de funções de ordenação nas quais um terminal CCA foi encontrado nos experimentos realizados com as coleções TREC-8 e WBR99.

Capítulo 5

Conclusões e Trabalhos Futuros

Neste capítulo apresentamos um resumo das conclusões desta dissertação e as direções a serem seguidas em trabalhos futuros.

5.1 Conclusões

Neste trabalho, foi apresentada uma nova abordagem para a geração de funções de ordenação, usando programação genética, baseada na combinação de componentes de ponderação de termos, extraídos de sistemas de recuperação de informação bem conhecidos da literatura e comprovadamente efetivos. A esta abordagem foi dado o nome de Abordagem de Componentes Combinados (CCA). A abordagem CCA melhorou a efetividade da tarefa de recuperação de informação quando comparada às abordagens tradicionais tais como *tf-idf*, Okapi BM25 e outra abordagem GP (FAN-GP) [13, 41] que utiliza apenas informações estatísticas básicas derivadas da coleção de documentos. Foram utilizadas as coleções TREC-8 e WBR99 para a validação da abordagem CCA.

Usando-se a coleção TREC-8, os experimentos mostraram que CCA levou a uma significante melhora na efetividade da tarefa de recuperação de informação. A média das precisões médias (MAP) foi melhorada em cerca de 40% e 14% em relação a BM25 e FAN-GP respectivamente. Foram também observados outros ganhos na *R-precision* e na precisão no topo do *ranking*. Para a coleção WBR99, foram obtidos ganhos de cerca de 21% e 24% na média das precisões médias sobre *tf-idf* e FAN-GP respectivamente. Todos os ganhos na média das precisões médias foram estatisticamente significantes.

Examinando o processo evolutivo da abordagem CCA, foi observado que as funções descobertas convergiram mais rapidamente do que FAN-GP. Ou seja, CCA é capaz de

encontrar um boa função evoluindo os indivíduos através de um número menor de gerações se comparada com a abordagem FAN-GP. A abordagem CCA também conseguiu reduzir o problema do "treinamento exagerado", no qual as funções ficariam muito especializadas para o conjunto utilizado no treino.

Os resultados obtidos pela abordagem CCA permitem a conclusão de que o uso de terminais significativos, como os componentes extraídos de outras funções de ordenação, ao invés de usar simplesmente informações estatísticas básicas, melhora a qualidade do processo de descoberta de funções de ordenação usando GP. Isto responde a hipótese de pesquisa formulada na Seção 1.3. Esta conclusão é possível, pois os resultados indicaram ganhos significativos na média das precisões médias, redução do *overfitting* e convergência mais rápida do processo evolutivo na geração de um boa função de ordenação. GP também pode ser considerada um técnica efetiva para a tarefa de encontrar um esquema de ponderação, a partir de componentes de ponderação de termos já conhecidos.

5.2 Trabalhos Futuros

Nesta seção são apresentadas algumas sugestões para a continuidade deste trabalho.

Utilização de Novos Terminais

Um trabalho futuro interessante é o de investigar outros modelos de recuperação de informação não considerados neste trabalho, como por exemplo o *Set-based Model* [28] e a abordagem *Language Modeling* [26], na busca de novas instâncias dos componentes apresentados na Seção 2.1.4.

Busca na Web

Um ponto não coberto por este trabalho de pesquisa foi o uso de outras evidências comumente encontradas em documentos Web, tais como as ligações (*links*) existentes entre os documentos e informações estruturais como título, texto âncora etc. Por isso, uma outra sugestão de trabalho futuro seria a de adequar a abordagem CCA para utilizar estas novas evidências, viabilizando, dessa maneira, a comparação da abordagem CCA com outras abordagens de descoberta de funções de ordenação para a tarefa de busca na Web, como por exemplo a abordagem proposta por Fan *et al.* em [14] que utiliza informações da estrutura de documentos Web.

Variações no Arcabouço GP

Um outro possível trabalho futuro seria a investigação da influência de algumas variáveis presentes no arcabouço GP tais como o fator de mutação, a profundidade máxima da árvore que define um indivíduo, o tamanho da população e também o tamanho dos conjuntos de treino e validação. Um caminho para esta atividade seria o de executar novas baterias de experimentos variando-se o fator de mutação, a profundidade máxima da árvore e o tamanho da população, por exemplo.

Pode também ser avaliado o uso de programação genética orientada a gramática para restringir o espaço de busca e tentar direcionar a combinação dos terminais. Por exemplo, criar uma restrição para os indíviduos sejam formados por uma sub-árvore de cada componente de ponderação de termo — tfc, cfc e nc. Dessa forma, os terminais relativos a cada componente seriam combinados separadamente, isto é, terminais tfc só seriam combinados com outros semelhantes e assim por diante.

Impacto do Tamanho do Conjunto de Treino

Em função da análise dos terminais, um novo experimento também seria interessante: avaliar a abordagem CCA apenas com os terminais que mais ocorreram nas melhores funções e, além disso, verificar qual o impacto desta redução. Um outro possível próximo passo é o de avaliar o efeito do tamanho do conjunto de treino na qualidade das funções de ordenação geradas pelo arcabouço GP. Isto é, utilizar conjuntos de treino de diferentes tamanhos, contendo amostras aleatórias de consultas, para tentar identificar os tamanhos ideal e mínimo do conjunto de treino, mantendo a efetividade das funções geradas e verificando as possíveis perdas.

Funções de Ordenação Específicas para Grupos de Consultas

Como pôde ser observado nos histogramas de comparação consulta-a-consulta, Figura 4.4, como ocorre com qualquer função de ordenação, o desempenho das funções descobertas pela abordagem CCA varia de consulta para consulta. Por isso, uma sugestão de continuidade deste trabalho seria a de examinar detalhadamente um conjunto de consultas, procurando identificar características que distinguam uma consulta das demais, como por exemplo, o número de termos da consulta, o valor médio do *idf* destes termos e a natureza da consulta — informacional, navegacional etc. Uma vez identificadas, estas características podem facilitar a criação de agrupamentos de consultas, para os quais possa ser descoberta uma

função de ordenação diferente. A hipótese a ser investigada é a de que funções de ordenação mais específicas para um grupo de consultas, podem apresentar resultados mais efetivos do que uma única função descoberta para responder a qualquer consulta em uma coleção de documentos.

Análise dos Indivíduos Gerados

Finalmente, mas não menos importante, é necessário realizar uma análise mais detalhada das funções de ordenação descobertas para entender melhor como elas trabalham e quais as razões que levaram aos ganhos na efetividade da tarefa de recuperação de informação.

Assim como apresentado por Zhang em [49], um caminho possível para esta etapa de análise seria o exame detalhado das árvores de um conjunto de bons indivíduos, avaliando quais sub-árvores são comuns entre estes indivíduos. Estas porções comuns de funções de ordenação podem indicar uma possível explicação para o bom desempenho destes indivíduos.

Bibliografia

[1] J. Allan, J. P. Callan, F. Feng, and D. Malin. INQUERY and TREC-8. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, pages 637–644, Gaithersburg, MD, 1999. NIST Special Publication 500-246.

[2] H. M. Almeida, M. A. Gonçalves, M. Cristo, and P. Calado. A combined component approach for finding collection-adapted ranking functions based on genetic programming. In *Proceedings of the 30th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 399–406, 2007.

[3] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley-Longman, Boston, MA, 1999.

[4] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic programming: an introduction: on the automatic evolution of computer programs and its applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.

[5] B. T. Bartell, G. W. Cottrell, and R. K. Belew. Automatic combination of multiple ranked retrieval systems. In *Proceedings of the 17th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 173–181, Dublin, Ireland, 1994.

[6] J. Battelle. *A Busca*. Elsevier, Rio de Janeiro, 2005.

[7] C. Buckley, A. Singhal, and M. Mitra. New retrieval approaches using smart: TREC 4. In *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pages 25–48, Gaithersburg, MD, 1996. NIST Special Publication 500-236.

[8] C. Darwin. *The Origin of Species by Means of Natural Selection*. John Murray, London, 1859.

[9] W. Fan, E. A. Fox, P. Pathak, and H. Wu. The effects of fitness functions on genetic programming-based ranking discovery for web search. *Journal of the American Society for Information Science and Technology*, 55(7):628–636, 2004.

[10] W. Fan, M. Gordon, and P. Pathak. On linear mixture of expert approaches to information retrieval. *Decision Support Systems*, 42(2):975–987, 2006.

[11] W. Fan, M. D. Gordon, and P. Pathak. Personalization of search engine services for effective retrieval and knowledge management. In *Proceedings of the 21st International Conference on Information Systems*, pages 20–34, Brisbane, Australia, 2000.

[12] W. Fan, M. D. Gordon, and P. Pathak. Discovery of context-specific ranking functions for effective information retrieval using genetic programming. *IEEE Transactions on Knowledge and Data Engineering*, 16(4):523–527, 2004.

[13] W. Fan, M. D. Gordon, and P. Pathak. A generic ranking function discovery framework by genetic programming for information retrieval. *Information Processing and Management*, 40(4):587–602, 2004.

[14] W. Fan, M. D. Gordon, and P. Pathak. Genetic programming-based discovery of ranking functions for effective web search. *Journal of Management Information Systems*, 21(4):37–56, 2005.

[15] W. Fan, M. D. Gordon, P. Pathak, W. Xi, and E. A. Fox. Ranking function optimization for effective web search by genetic programming: An empirical study. In *Proceedings of 37th Hawaii International Conference on System Sciences*, pages 105–112, Hawaii, USA, 2004.

[16] G. W. Furnas, S. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, and K. E. Lochbaum. Information retrieval using a singular value decomposition model of latent semantic structure. In *Proceedings of the 11th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 465–480, Grenoble, France, 1988.

[17] H. Greisdorf. Relevance: An interdisciplinary and information science perspective. *Informing Science The International Journal of an Emerging Transdiscipline*, 3(2):67–72, 2000.

[18] F. Gruau. On using syntactic constraints with genetic programming. In *Advances in genetic programming*, volume 2, pages 377–394. MIT Press, Cambridge, MA, USA, 1996.

[19] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.

[20] J. R. Koza. Hierarquical genetic algorithms operating on populatins of computer programs. In *Proceedings of the 11th International Joint Conference on Artificial Intelligent (IJCAI-89)*, pages 768–774, Detroit, MI, 1989.

[21] J. R. Koza. *Genetic Programming: On the programming of computers by natural selection*. MIT Press, Cambridge, 1992.

[22] A. Lacerda, M. Cristo, M. A. Gonçalves, W. Fan, N. Ziviani, and B. Ribeiro-Neto. Learning to advertise. In *Proceedings of the 29th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 549–556, Seattle, Washington, USA, 2006.

[23] T. M. Mitchell. *Machine Learning*. McGraw Hill, New York, NY, 1997.

[24] N. Oren. Reexamining tf.idf based information retrieval with genetic programming. In *Proceedings of the South African Institute of Computer Scientists and Information Technologists (SAICSIT) 2002 Conference*, pages 224–234, Port Elizabeth, South Africa, 2002.

[25] P. Pathak, M. Gordon, and W. Fan. Effective information retrieval using genetic algorithms based matching functions adaptation. In *Proceedings of the 33rd Hawaii International Conference on System Sciences HICSS*, Maui, Hawaii, USA, 2000.

[26] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, Melbourne, Australia, 1998.

[27] B. Pôssas, N. Ziviani, W. Meira, Jr, and B. Ribeiro-Neto. Set-based model: a new approach for information retrieval. In *Proceedings of the 25th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 230–237, Tampere, Finland, 2002.

[28] B. Pôssas, N. Ziviani, W. Meira, Jr, and B. Ribeiro-Neto. Set-based vector model: An efficient approach for correlation-based ranking. *ACM Transactions on Information Systems TOIS*, 23(4):397–429, 2005.

[29] B. Ribeiro-Neto and R. Muntz. A belief network model for ir. In *Proceedings of the 19th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 253–260, Zurich, Switzerland, 1996.

[30] S. E. Robertson. The probability ranking principle in ir. *Journal of Documentation*, 33(4):294–304, 1977.

[31] S. E. Robertson and K. Sparck-Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976.

[32] S. E. Robertson and S. Walker. Okapi/keenbow at TREC-8. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, pages 151–162, Gaithersburg, MD, 1999. NIST Special Publication 500-246.

[33] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 109–126, Gaithersburg, MD, 1995. NIST Special Publication 500-226.

[34] E. L. M. Rodrigues. Evolução de funções em programação genética orientada a gramática. Master's thesis, Setor de Ciências Exatas, Universidade Federal do Paraná, Abril 2002.

[35] G. Salton. *The SMART retrieval system - Experiments in automatic document processing*. Prentice Hall Inc., Upper Saddle River, NJ, 1971.

[36] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[37] G. Salton and M. E. Lesk. Computer evaluation of indexing and text processing. *Journal of the ACM*, 15(1):8–36, 1968.

[38] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[39] T. Saracevic. Information science. *Journal of the American Society of Information Science*, 50(12):1051–1063, 1999.

[40] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *Proceedings of the 19th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29, Zurich, Switzerland, 1996.

[41] A. Trotman. Learning to rank. *Information Retrieval*, 8(3):359–381, 2005.

[42] H. Turtle and W. B. Croft. Inference networks for document retrieval. In *Proceedings of the 13th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1–24, Brussels, Belgium, 1990.

[43] C. C. Vogt and G. W. Cottrell. Fusion via a linear combination of scores. *Information Retrieval*, 1(3):151–173, 1999.

[44] E. M. Voorhees and D. Harman. Appendix: Evaluation techniques and measures. In *Proceedings of TREC-8*, pages 17–23, Gaithersburg, MD, 1999. NIST Special Publication 500-246.

[45] E. M. Voorhees and D. Harman. Overview of the eighth Text REtrieval Conference (TREC-8). In *Proceedings of TREC-8*, pages 1–24, Gaithersburg, MD, 1999. NIST Special Publication 500-246.

[46] P. A. Whigham. Grammatically-based genetic programming. In *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, pages 33–41, Tahoe City, California, USA, 9 1995.

[47] R. Wilkinson and P. Hingston. Using the cosine measure in a neural network for document retrieval. In *Proceedings of the 14th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 202–210, Chicago, USA, 1991.

[48] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers, San Francisco, CA, 1999.

[49] B. Zhang. *Intelligent Fusion of Evidence from Multiple Sources for Text Classification*. PhD thesis, Virginia Polytechnic Institute and State University, Junho 2006.

[50] J. Zobel and A. Moffat. Exploring the similarity space. *SIGIR Forum*, 32(1):453–490, 1998.